ISO/CCITT and Internet Management Coexistence (IIMC): Translation of ISO/CCITT GDMO MIBs to Internet MIBs (IIMCOMIBTRANS)

March 26, 1993

Owen Newnan (Editor)

U S WEST Advanced Technologies
4001 Discovery Lane, Suite 190
Boulder, Colorado 80303
onewnan@atqm.advtech.uswest.com

Status of this Memo

This document provides information to the network and systems management community. This document is intended as a contribution to ongoing work in the area of multi-protocol management coexistence and interworking. This document is part of a package; see also [IIMCIMIBTRANS] [IIMCMIB-II] [IIMCPROXY] and [IIMCSEC]. Distribution of this document is unlimited. Comments should be sent to the Network Management Forum IIMC working group (iimc@thumper.bellcore.com).

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.''

Please check the 1id-abstracts.txt listing contained in the internet-drafts Shadow Directories on nic.ddn.mil, nnsc.nsf.net, nic.nordu.net, ftp.nisc.sri.com, munnari.oz.au to learn the current status of any Internet Draft.

Abstract

This document is intended to facilitate the use of ISO/CCITT MIBs for integrated management of networks via proxy management of TCP/IP networks that are managed using SNMP. This document provides heuristic procedures that translate managed object specifications from ISO/CCITT Guidelines for Definition of Managed Object (GDMO) templates to Internet MIB macros. Currently, some market segments demand SNMP for customer network management, while others require the ISO/CCITT Common Management Information Protocol (CMIP). The approach defined in this document accommodates both, protecting investment already made in management information specifications and minimizing cost to implement a second protocol where the market requires. This translation is designed to: lose as little functionality as possible in translation; minimize need for human involvement during translation; minimize cost to implement dual protocol and proxy-based applications; and support generic network models that span many computer platforms and network elements. This document in intended to encourage standardization of such an approach and promote consistent usage of MIB definitions, independent of protocol.

Table of Contents

Status of this Memo i
Revision History iii
1. Introduction 1
1.1 Background 1
1.2 Overview 2
1.3 Scope 4
1.4 Terms and Conventions 6
2. Translation Procedures 6
2.1 Relationship to RFC 1212 6
2.2 Mapping of Managed Object Classes and Attributes 7
2.3 Mapping to the SYNTAX clause 15
2.4 Generation of Internet MIB Identifiers 18
2.5 Mapping to the ACCESS clause 21
2.6 Mapping to the STATUS clause 21
2.7 Mapping to the DESCRIPTION clause 22
2.8 Mapping to the REFERENCE clause 22
2.9 Mapping to the INDEX clause 23
2.10 Mapping to the DEFVAL clause 23
2.11 Mapping of Actions 23
2.12 Translation of Notifications 24
2.13 Mapping of Delete Operations 27
2.14 Translation of Create Operations 28
3. Constraints on SNMP Usage 29
4. Summary 31
5. Acknowledgments 32
Appendix A Abridged Example 36
Appendix B Abridged ISO/CCITT Compatibility MIB 46

Revision History

Draft 0 - October 9, 1992
Initial draft of this document.

Draft 1 - March 26, 1993
Current draft of this document (replaces Draft 0).

Major Changes Since Last Revision

At the IIMC meeting of February 8-9, 1993, the editor was instructed to make revisions and supply proposed text in certain areas for the next draft of this document. Those changes have been applied to Draft 0 in arriving at this draft (Draft 1). Following is a summary of the edits applied:

1. Auto-registry of generated ASN.1 Modules was added to translation of notifications.
2. Index Usage in Translated MIBs was clarified.
3. An algorithm is provided for mapping notifications to traps and Appendix B is largely rewritten to provide the example output, derived from ISO 10165-2 (DMI). In the process, several miscellaneous errors were corrected.
4. The "OSI Compatibility MIB" was renamed "ISO/CCITT Convergence MIB". (Alignment with the name of the ISO/CCITT Proxy MIB should also be considered).
5. Objectives have been clarified to indicate that SNMPv2 is in the scope of the first Issue of this document, and pragmas are not.
6. Machine Readable Conventions for REFERENCE Values have been specified to facilitate mechanized translation.
7. Translating the attributes of ISO/CCITT Top have been clarified. The NAME BINDING table provides the only attribute of Top that appears to be needed.

Action Item Proposals Contained In This Document

#19 Current Issues List

Outstanding Issues

The following actions were identified at the February IIMC meeting, but have not yet been addressed in this draft. Proposals in these areas would be most welcome.

#21 A More Extensive Example for Appendix A.
#20 Add support for SNMPv2.

1.Introduction

The past decade has witnessed the development of enterprise wide networks composed of a multi-vendor environment containing heterogeneous protocol and hardware suites. Organizations have become increasingly dependent on these enterprise networks for their daily operations. This dependence has focused attention on the need for operation, administration, maintenance, and provisioning (OAM&P) of the multi-vendor enterprise network on an end-to-end basis.

1.1 Background

This document is part of a package of ISO/CCITT and Internet Management Coexistence (IIMC) drafts. Other documents included in this package are:

[IIMCIMIBTRANS]	Translation of Internet MIBs to ISO/CCITT GDMO MIBs
[IIMCMIB-II]	Translation of Internet MIB-II(RFC 1213) to ISO/CCITT GDMO MIB
[IIMCSEC]	ISO/CCITT to Internet Management Security
[IIMCPROXY]	ISO/CCITT to Internet Management Proxy

These documents together comprise a package aimed at integrating ISO/CCITT-based and Internet-based management systems. These documents represent coexistence and interworking efforts underway within the IIMC working group, chartered under the auspices of the Network Management Forum Architecture Integration ISO/Internet technical team.

This work was initiated, in part, by NM Forum efforts to translate RFC 1214 for use with OMNIPoint 1 implementations. Through this effort, it became obvious that end-to-end management requires an integrated, unified view of the managed network, despite differences in management protocol and information structure. Integrated management can be facilitated by the development of "proxy" mechanisms which translate between functionally equivalent service, protocol, and SMI differences to create this unified view. MIB translation procedures can be used to support proxy management, as well as to take advantage of existing MIB definition and avoid duplication of effort. In this way, commercial investment in both ISO/CCITT and Internet-based management technologies can be preserved through deployment of common methods and tools which support integration.

This overall strategy was outlined in a joint publication developed by the NM Forum and X/Open entitled "ISO/CCITT and Internet Management: Coexistence and Interworking Strategy" [NMFMC92]. The documents included in the IIMC package are

the next level of detailed specifications which implement several of the methodologies identified in the strategy.

1.2 Overview

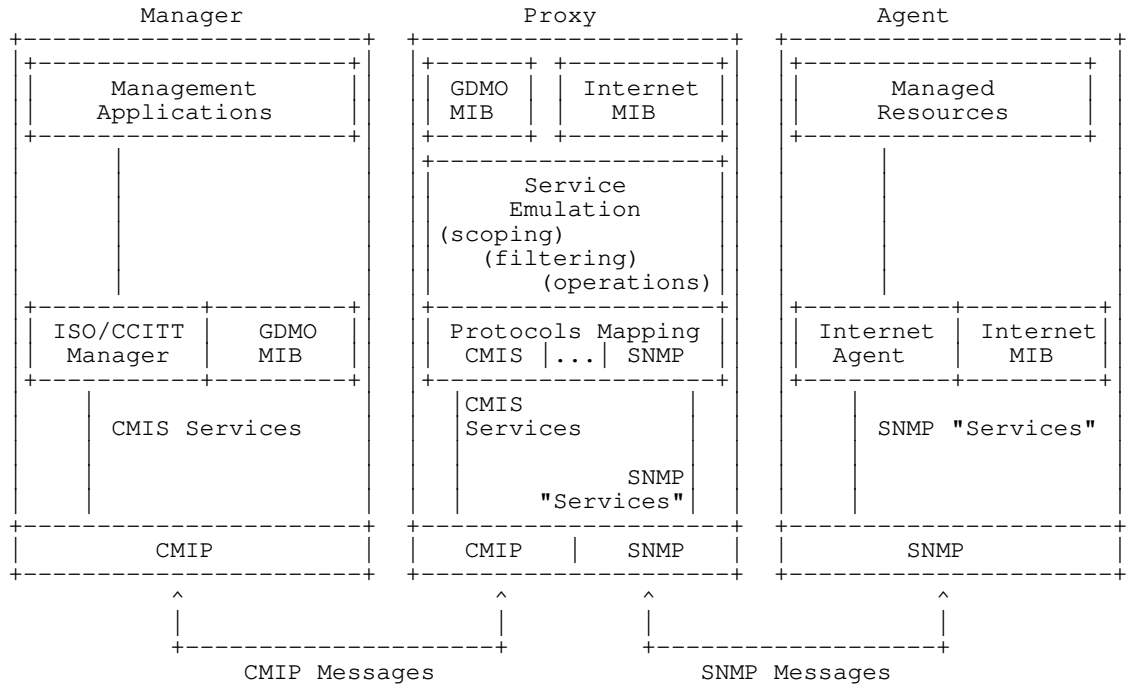
The response to the need for OAM&P of enterprise networks has been the development of network management standards within various networking communities - most notably the ISO/CCITT and Internet communities. However, coordination of standards activities between these two communities has not occurred. As a result, although they share a nearly common management model, differences in their management protocols and structures of management information (SMIs) have developed due to differing management philosophies.

The ISO/CCITT community has developed the Common Management Information Protocol (CMIP) [ISO9596-1], and related SMI documents [ISO10165-1,2,4]. The Internet community has developed the Simple Network Management Protocol (SNMP) [RFC1157], and its successor, SNMPv2 [SNMPv2PROT]. The Internet SMI is defined in [RFC1155] and [SNMPv2SMI]. Although functionally similar, the Internet and ISO/CCITT protocols and SMIs differ in terms of their complexity and specific operations.

The focus on the need for end-to-end enterprise management has indicated the need to integrate the management of components accessed by ISO/CCITT management, Internet management and proprietary management mechanisms in a manner which presents a unified view of the network, despite protocol and SMI differences. One way to integrate management is by the development of "proxy" mechanisms which translate between functionally equivalent services, protocol and SMI differences to create this unified view.

A body of telecommunications and computer vendors, represented by organizations such as the Network Management Forum (NMF), and the U.S. government, as specified in the Government Network Management Profile (GNMP) have based their integrated management model on the ISO/CCITT management model using CMIP and the ISO/CCITT SMI. These organizations are particularly interested in the development of proxies for devices that use the Internet management protocols and SMI. Their interest is primarily due to the widespread commercial implementation and use of such devices within their enterprises, especially devices that use the Internet TCP/IP protocol suite.

The basic model for ISO/CCITT-Internet proxy management is illustrated in the following diagram.



The proxy architecture provides emulation of CMIS services by mapping to the corresponding SNMP message(s) necessary to carry out the service request. The service emulation allows management of Internet objects by an ISO/CCITT manager. The left hand side of the proxy behaves like an ISO/CCITT agent, communicating with the ISO/CCITT manager using CMIP protocols. The right hand side of the proxy behaves like an Internet manager, communicating with the Internet agent using SNMP protocols.

The proxy relies on the existence of a pair of directly-related MIB definitions, where the Internet MIB has been translated into ISO/CCITT GDMO using the procedures specified in [IIMCIMIBTRANS]. The proxy defined in [IIMCPROXY] uses these MIB definitions and rules to provide run-time translation of management information carried in service requests and responses.

The proxy architecture is designed with a specified interface between the proxy and the underlying protocol stacks, and so deals primarily in terms of CMIS services and SNMP "services". The proxy emulates services such as CMIS scoping and filtering, processing of CMIS operations, and forwarding/logging of CMIS notifications by performing a mapping process which must be tailored for each protocol (for example, SNMP and SNMPv2 are variants of the same protocol mapping process).

In addition, this document specifies translation procedures for converting ISO/CCITT GDMO MIBs into Internet MIBs. MIBs generated by this translation process cannot be utilized by the Proxy defined in [IIMCPROXY], although another kind of Proxy could be defined for this purpose in the future.

Finally, note that MIBs translated by procedures such as those defined by [IIMCIMIBTRANS] and [IIMCOMIBTRANS] may also be used without a proxy. For example, a translated MIB may be used to take advantage of existing MIB definitions when business needs require deployment in a different management environment. Translated MIBs may also be used to provide uniformity when multiple management environments are supported by a single system (e.g., dual stack managers).

1.3 Scope

In recent years computer networks have experienced an explosion in the number and complexity of objects to be managed. Especially challenging have been environments where platforms from many vendors must interact and complex software and hardware configurations must be supported. A chronic concern for such environments is end-to-end problem determination and resolution.

Consequently there has been much effort toward standardizing the management of applications, networks, services and systems. Despite this major investment, consumers and standards participants have often found progress disturbingly slow — especially in standardizing management information.

To further complicate matters, different subcultures have developed differing philosophies and technologies for network management. Notable examples are the Internet and ISO/CCITT communities. Although MIB work in these communities has so far mostly been complementary there is increasing danger of duplicative, inconsistent and competitive MIB specification.

Standardization is a political process and each philosophy of network management has its merits. However a network manger rarely has the luxury of being politician or philosopher; they must pragmatically determine problems in the network and rapidly resolve them. Typically, service must be assured in an end-to-end environment management by a wide range of technologies — Internet, ISO/CCITT, and otherwise.

If network management is to meet needs of such network operators, an "ecumenical" approach to MIBs is required that marries the work of these standards for a thus providing a comprehensive and consistent view of networked resources. An end-to-end approach is needed, one that conceals

differences of protocol and presents the consolidated view of distributed resources to network operators, system administrators and programmers.

This implies a common MIB independent of protocol. One way to arrive at this is to pursue comprehensive and mechanizable procedures to assist in translating MIB specifications. This should allow rapid sharing of MIB specifications while requiring minimal technical and political intervention by human beings.

What you are reading is a contribution toward this end, a heuristic procedure to translate from ISO/CCITT GDMO MIB specifications to the Internet MIB macro specifications. This translation procedure is written with four objectives in mind:

- Lose as little functionality as possible in translation.
- Minimize need for human involvement to translate.
- Minimize cost to implement dual protocol and proxy-based applications.
- Support generic network models that span many computer platforms and network elements.

While an entirely mechanized translation from an ISO/CCITT GDMO MIB to an Internet MIB is not always possible, the intent is to mechanize the process as much as possible and supply reasonable defaults that then may be tempered with human judgment.

In the longer term, MIBs translated in this manner might be used in conjunction with a proxy architecture that enables interworking between ISO/CCITT managers and Internet agents, or vice versa.

Following is a procedure for translating management information bases (MIB's) from ISO/CCITT Guidelines for Definition of Managed Objects (GDMO) format to that of the Internet MIB macro format [RFC1212]. The body of this document has five parts:

- This introduction, including motivation and objectives for the translation.
- The translation procedure itself.
- Constraints on use of the SNMP with MIBs translated by this procedure.
- Summary.

Sample inputs and translated outputs are also provided in an appendix. Examples used throughout the body of the text are taken from these samples.

The following outstanding issues have been identified for inclusion in future versions of this document.

- This procedure is based on current Internet SMI standards, but should be extended to also cover proposed SNMPv2 SMI standards. This is a definite requirement for Issue 1 of this algorithm.

- Certain input "pragmas" may be appropriate to document human overrides to algorithmic translation in a systematic and machine readable form. Among other things, this might facilitate generation of proxies. Pragmas, however, are beyond the scope of Issue 1.

This paper assumes the existence of appropriate mechanisms and procedures for registry of translated objects. What those procedures might be and where such objects should be registered, however, is beyond the scope of this document.

1.4 Terms and Conventions

The reader is assumed to be familiar with the vocabularies of Internet and ISO/CCITT management; in cases where there might be confusion between the two, words such as "ISO/CCITT", "GDMO" and "Internet" are inserted to avoid ambiguity.

The following conventions are used throughout the paper:

The terms "class" and "attribute" when expressed in lower case are generic, referring either to ISO/CCITT MANAGED OBJECT CLASS's and ATTRIBUTE's (respectively) or their translated Internet MIB counterparts.

The term "arc" means a single level of branching within an Abstract Syntax Notation One(ASN.1) registration tree.

The terms "enumerate" and "explode" are used synonymously to describe the process of translating ATTRIBUTE's and their values to OBJECT TYPE macros.

A "registry family" is defined to be a set of ASN.1 OBJECT IDENTIFIER arcs and node shaving a common immediate parent.

Footnotes explore aspects of the translation procedure where human judgment may be especially advisable, rather than accepting the raw output of a translator.

2. Translation Procedures

2.1 Relationship to RFC1212

While translation per se has not been widely investigated, [RFC1212] does provide advice for adopting MIB objects from ISO/CCITT GDMO to Internet MIB macros. However, RFC 1212 advises a minimalistic approach to MIB specification,

discouraging carryover of the complexities often found in ISO/CCITT GDMO specifications. Thus, it does not so much tell how to translate a MIB as provide advice for borrowing elements of a ISO/CCITT GDMO specification and constructing a MIB more in keeping with Internet philosophy.

The translation procedure provided here seeks instead to provide as faithful a translation as possible, in order to support the objectives identified in section 1.2 above.

As applicable, the subsections are divided into one or more of the following parts:

- Relevant advice quoted verbatim from RFC1212. Beware that the entire RFC is not quoted here. The reader is referred to the source for complete text.
- Additional constraints are identified to allow as comprehensive and mechanizable an approach as possible.
- Discussion of the translation procedure is provided as guidance to the reader.

2.2 Mapping of Managed Object Classes and Attributes

2.2.1 RFC 1212 Advice

... The next step is to categorize the objects into groups. For experimental MIB's, optional objects are permitted. However, when a MIB module is placed in the Internet-standard space, these optional objects are either removed, or placed in an optional group, which, if implemented, all objects in the group must be implemented. For the first pass, it is wisest to simply ignore any optional objects in the original MIB: experience shows it is better to define a core MIB module first, containing only essential objects; later, if experience demands, other objects can be added.

It must be emphasized that groups are "units of conformance" within a MIB: everything in a group is "mandatory" and implementations do either whole groups or none.

Next for each managed object class, determine whether there can exist multiple instances of that managed object class. If not, then for each of its attributes, use the OBJECT TYPE macro to make an equivalent definition.

Otherwise, if multiple instances of the managed object class can exist, then define a conceptual table having conceptual rows each containing a columnar object for each of the managed object class's attributes. If the managed object class is contained within the containment tree of another managed object class, then the assignment of an object type is normally required for each of the "distinguished attributes" of the containing managed object class. If they

do not already exist within the MIB module, then they can be added via the definition of additional columnar objects in the conceptual row corresponding to the contained managed object class.

In defining a conceptual row, it is useful to consider the optimization of network management operations which will act upon its columnar objects. In particular, it is wisest to avoid defining more columnar objects within a conceptual row, than can fit in a single PDU. As a rule of thumb, a conceptual row should contain no more than approximately 20 objects. Similarly, or as a way to abide by the "20 object guideline", columnar objects should be grouped into tables according to the expected grouping of network management operations upon them. As such, the content of conceptual rows should reflect typical access scenarios, e.g., they should be organized along functional lines such as one row for statistics and another row for parameters, or along usage lines such as commonly-needed objects versus rarely-needed objects.

On the other hand, the definition of conceptual rows where the number of columnar objects used as indexes outnumbers the number used to hold information, should also be avoided. In particular, the splitting of a managed object class's attributes into many conceptual tables should not be used as a way to obtain the same degree of flexibility/complexity as is often found in MIB's with a myriad of optionality.

2.2.2 Additional Constraints

This subsection addresses:

- Mapping of MANAGED OBJECT CLASSES and Distinguished Names.
- Mapping of PACKAGE's.

It deals only with the high level procedures for mapping ISO/CCITT GDMO MANAGED OBJECT CLASSES and ATTRIBUTE's into Internet MIB macros. Enumeration of individual ATTRIBUTE values is addressed in subsection 2.3.

2.2.2.1 Mapping of MANAGED OBJECT CLASSES and Distinguished Names

Translation of a registry family of MANAGED OBJECT CLASS specifications begins by

- Allocating the root of a registry subtree to contain the translated objects.
- Assigning a brief naming prefix that distinguishes contents of a corresponding ASN.1 module generated by the translation. The module itself is registered at the root of the registry subtree.

> Note: Assignment of naming prefixes and registry subtrees are required activities, however, procedures for these are outside the scope of this paper

For each MANAGED OBJECT CLASS of the input registry family, define a corresponding Internet MIB object group, its "class group" Reserve an arc for each of these, keeping the same relative arc number as is assigned to its equivalent MANAGED OBJECT CLASS in the input ISO/CCITT GDMO registry. Avoid registration of ISO/CCITT objects under arc zero(0) by using the value 16,384 instead.

For each class group define one Internet MIB table — a "class table" — that represents the class as a whole. Assign this table arc number 1 beneath the arc of the class group. As will be further discussed in subsection 2.3 below, "side tables" will also often be required for a class group to support multi-valued ATTRIBUTE's and ATTRIBUTE's with complex syntaxes. Assign arcs for side tables in ascending order beneath the arc of the class group beginning with arc number two.

> Note: Since all ATTRIBUTE's expand into class tables or side tables, class groups generated by the algorithm never contain scalar values.

For each table in the class group, define a table entry object and syntax consistent with RFC 1212 usage for table entries.

For the entries of each class table, begin by allocating the following conceptual columns and associated arcs:

- Leg 0 beneath the class entry arc is reserved (by the Internet Structure of Management Information (SMI)).

- Assign leg 1 the delete object, a write-only INTEGER for which 0 indicates deletion of the corresponding conceptual row.

- Leg 2 of the arc is assigned the "class entry index," an INTEGER valued object that provides the unique index for an entry to the class table. This distinguishes an instance of an object within its class.
  The value of the class entry index is a component of the "class entry instance." The latter identifies both an object's class and the unique instance within that class. This value is used in the translated MIB instead of Distinguished Name's and ObjectInstance's that represent relationships between managed objects. As discussed later, no direct translation of Distinguished Names is attempted. The class entry instance is arrived at by concatenating:

```
-- The OBJECT IDENTIFIER of the class table entry.
-- The value 2 (arc number for the class entry index)
-- The value of the class entry index for the instance in
question.
```

Note: Where "concatenating" means arriving at a simple combined sequence of arc values, without repeat counts or nesting.

Entries of side tables begin in similar fashion:

- Leg 0 is reserved.

- Leg 1 is assigned the delete object.

- Leg 2 of the arc is assigned the class entry index, which has the same value as the corresponding class entry index of the class table. This ties side table entries to their corresponding class table entry.

- Leg 3 of the arc is assigned the first (and typically only) "value index"; this distinguishes a particular value of a multi-valued attribute or syntax from the other values.

- Legs 4-19 of the arc can be assigned if necessary to provide additional entry value indices. These are needed only for nested multiply occurring syntaxes.

Note: The mapping of multiply occurring elements of syntaxes is discussed further in later sections. While supported for completeness, multi-dimensional values represent an extreme case. Caution should be used in adopting the raw output of the algorithm for complex syntaxes.

2.2.2.2 Mapping of PACKAGE's

In reading this section the reader may wish to refer to Figures 1 and 2, which illustrate the input and output subtrees of the sample MIB (Appendix A). Note that for this example, the Trouble Administration module is rooted beneath an (optional) version arc, to facilitate future releases.

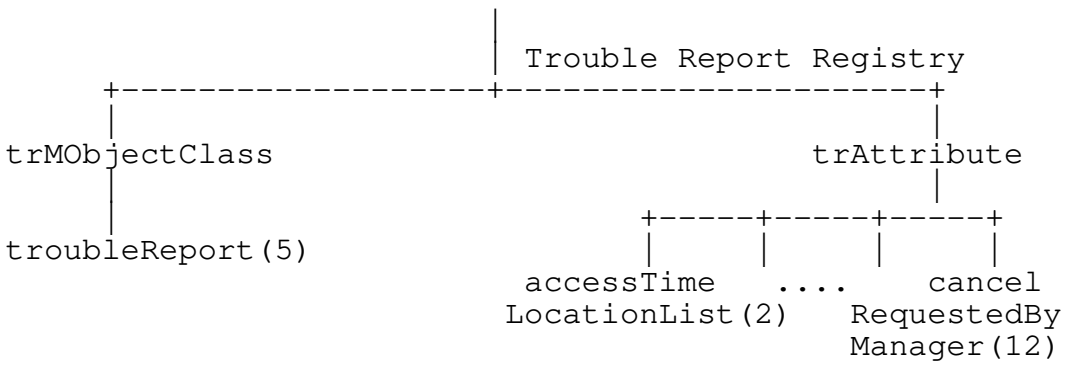


Figure 1. Sample Input Registration Tree

All else being equal, enumerate ATTRIBUTE's based upon a left-to-right scanning of the input ISO/CCITT GDMO specification. ATTRIBUTE's and their values may be enumerated multiple times in the course of translating a specification, once for every template in which they are referenced. For example, if an attribute is included in the PACKAGE's of two MANAGED OBJECT CLASS's, it will be enumerated twice in the output: once for each class group.

Enumerate ATTRIBUTE's and their values in the same sequence as declared in a PACKAGE. These translate to OBJECT TYPE's that, unless otherwise noted below, receive successive arcs in the output registry tree.

Enumerate all components of base classes before those of their derivative classes. Thus where a package is refined by a subclass, first enumerate all components of the base class, keeping the sequence of the base class PACKAGE's. Explode ATTRIBUTE's of a derivative class later, enumerating in the order of specification for that class.

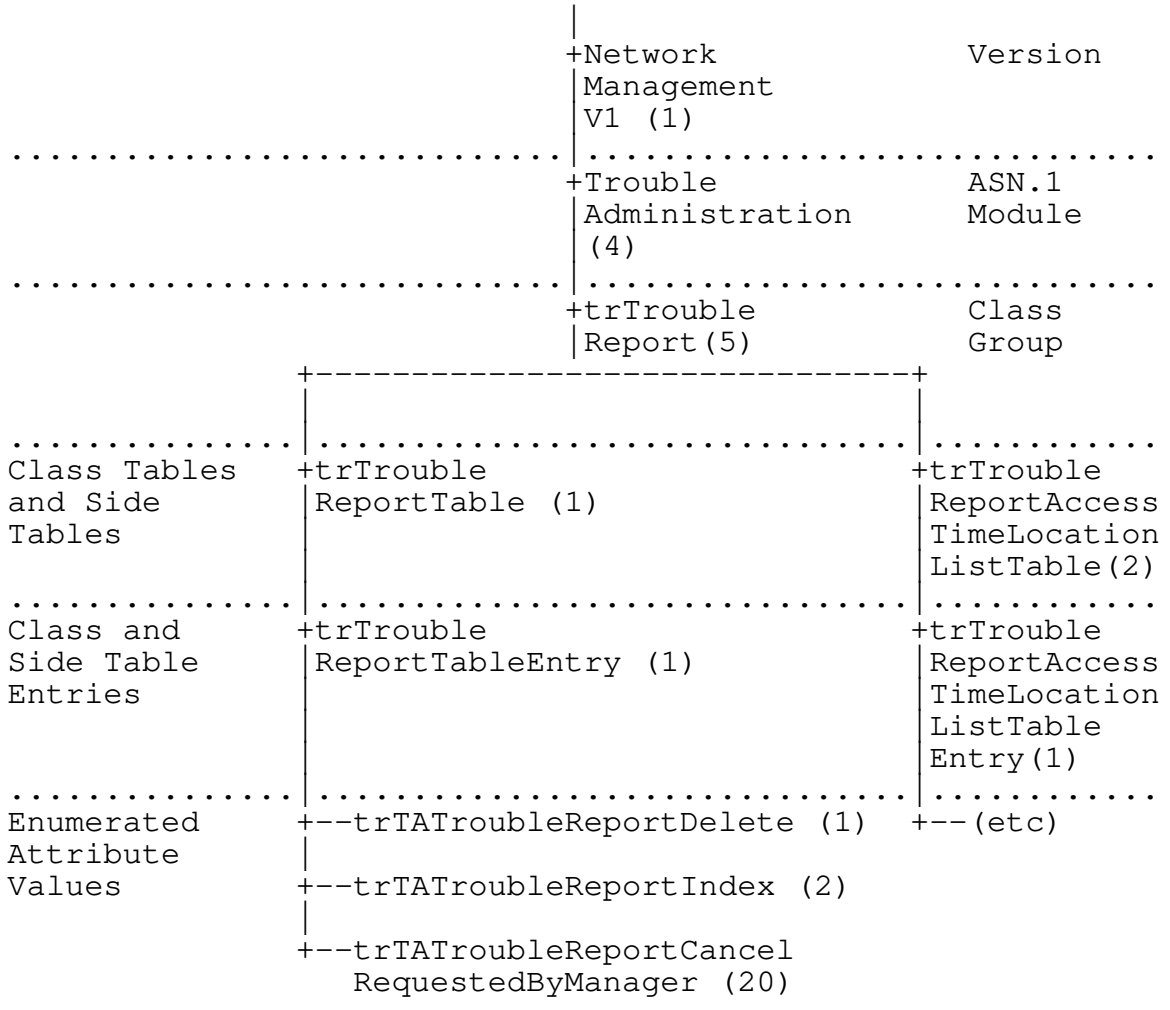


Figure 2. Sample Output Registration Tree

Reserve at least ten arcs for future growth for each level of derivation, i.e., take the highest arc number of the base class, add ten and round upwards to the nearest even multiple of ten, to determine the first arc number assigned to the derivative class.

For multiple inheritance, enumerate contents of base classes left-to-right and depth first, i.e., enumerate all components attributable to one immediate base class before proceeding to the next. In this case also reserve arcs for future growth by adding ten and rounding up to the nearest even multiple of ten.

2.2.3 Discussion

2.2.3.1 Mapping of MANAGED OBJECT CLASSES and Distinguished Names

RFC 1212 recommends defining a table for every object group that can be multiply occurring within an agent. It would be very unusual for a MANAGED OBJECT CLASS not to have potentially multiple occurrences, especially given a generic network model that spans systems. Thus to simplify translation, all object classes map to tables. This approach has several advantages:

- It supports default values for new object instances.

- It is easy to mechanize, since there is no need to recognize special cases that are not multiply occurring.

- It simplifies programming of proxy or dual protocol agents, since the programmer is dealing with basically the same syntax for scalar items, regardless of protocol.

The last point applies to both programming effort and processing overhead. To the extent the elements of syntax are mapped one-to-one, and underlying syntax is similar or identical for both protocols, they can be manipulated with common code and common data structures. This simplifies translation from both the programming and run-time perspectives.

The notion of a class entry instance is introduced since direct translation of Distinguished Name's appears impractical for the following reasons:

- A possible ambiguity arises since NAME BINDING's allow different object instances of the same MANAGED OBJECT CLASS to exist under parent objects of different classes. Likewise, different classes can have the same syntax for their distinguished attribute(s). Thus, a concatenation of attribute values is not sufficient to uniquely distinguish an object instance.

- NAME BINDING's allow different instances of the same class to be subordinate to different types of parent and even allow instances of a class to be contained recursively within others of the same class. Thus, in directly translating the DN one cannot assume a fixed sequence of parameters as required for by the INDEX clause (DN's for different instances of the same object class may be have different numbers of RDN's).

- Both problems could in theory be circumvented by fully translating the Distinguished Name and incorporating

Attribute Type's as well as Attribute Value's into the subidentifier OID (in which case the INDEX clause would only need to specify one index, an OBJECT IDENTIFIER).

While the latter is in theory possible, it would result in exceedingly verbose subidentifiers, on the order of 70 octets rather than 7. This is of serious concern due to PDU length restrictions for SNMP. RFC 1212 proposes a "rule of twenty," i.e., no more than twenty attributes per operation. That guideline was designed for relatively compact subidentifiers. When using an RDN for an INDEX, this would more likely amount to a "rule of three," which is why comprehensive translation of the ISO/CCITT DN to an INDEX appears practical.

The result of this approach is that Distinguished Names are NOT translated at all. Similar functionality (naming of objects) is instead provided by OBJECT IDENTIFIERS that identify the table entries which corresponding to MANAGED OBJECTs. The indexes of these tables are arbitrary integers that have no significance other than discriminate between the entries of a table. In general, all management information that is mapped by this specification to OBJECT-TYPEs is mapped to tables that have one or more arbitrary indexes of type INTEGER. Class table entries in particular have exactly one index.

For the following reasons, attributes of TOP are not directly translated into OBJECT TYPE's:

- Most of these notions will be unfamiliar to the Internet user and thus their presence would add questionable value.

- Multi-valued attributes would require additional side tables for all object class groups, which would be cumbersome.

- Managed object class is implicit in the class prefix thus does not require a special attribute.

- An allomorphs attribute is supported in ISO/CCITT to allow dynamic recognition of allomorphs which are supported by an instance. Since Internet MIB does not support allomorphs at all — much less dynamic ones — there is no reason to list them.

- There is no notion of NAME BINDING's for Internet MIB. NAME BINDING rules must still be enforced for the translated MIB, and should be documented via comments in the Internet MIB specifications. However, there seems to be no point in providing this attribute to the Internet user in the run time environment.

- Presence or absence of conditional packages can be detected using a GetNextRequest and determining whether the conceptual rows returned are the same as expected. No special attribute is needed for this purpose.

The ocNameBindingTable of the ISO/CCITT Convergence MIB provides a mechanism by which name bindings can be inspected, also provided upon object creation. This feature is included for completeness and to avoid ambiguity. It is unlikely to be used often, so it is placed in a table rather than enumerated in the class tables.

2.2.3.2 Mapping of PACKAGE's

Left-to-right sequential enumeration is the obvious approach for a mechanized translation procedure.

While ISO/CCITT GDMO allows ATTRIBUTE's and ASN.1 syntaxes to be referenced in multiple places and thus be shared, Internet MIB format does not. Thus one or more OBJECT TYPE's must be specified for each template in which they are referenced. The consequent explosion of enumerated ATTRIBUTE's and ASN.1 syntaxes is a major motivation for a mechanizable procedure and mechanized translation aids.

2.3 Mapping to the SYNTAX clause

2.3.1 RFC 1212 Advice

When mapping to the SYNTAX clause of the OBJECT TYPE macro:

(1) An object with BOOLEAN syntax becomes an INTEGER taking either of values true(1) or false(2).

(2) An object with ENUMERATED syntax becomes an INTEGER, taking any of the values given.

(3) An object with BIT STRING syntax containing no more than 32 bits becomes an INTEGER defined as a sum; otherwise if more than 32 bits are present, the object becomes an OCTETSTRING, with the bits numbered from left-to-right, in which the least significant bits of the last octet may be "reserved for future use."

(4) An object with a character string syntax becomes either an OCTETSTRING or a DisplayString, depending on the repertoire of the character string.

(5) An non-tabular object with a complex syntax, such as REAL or EXTERNAL, must be decomposed, usually into an OCTET STRING (if sensible). As a rule, any object with a complicated syntax should be avoided.

(6) Tabular objects must be decomposed into rows of columnar objects.

2.3.2 Additional Constraints

2.3.2.1 Simple Input Syntax

Following are rules for translating non-constructed (scalar)syntax.

For ENUMERATED types, transform to INTEGER, with values of (0) mapped to (32767).

Where ISO/CCITT management allows certain forms to be present or not present on a case by case basis (i.e., conditional packages and ASN.1 OPTIONAL and CHOICE syntaxes)enumerate all possibilities and allow corresponding conceptual columns to be conditionally present on a row-by-row basis.

For CHOICE types, provide an OBJECT TYPE for every alternative and populate exactly one of these alternatives per conceptual row.

For ASN.1 string types, use DisplayString whenever the character set actually expected in the element is a subset of DisplayString, else specify OCTET STRING.

In general, treat a constructed type that contains no more than one scalar (e.g., various forms of string specialization) as if it were the contained scalar.

ANY's and ANY DEFINED BY's map to OCTET STRING's that contain the BER encoded values of the corresponding ASN.1 types.

2.3.2.2.Complex Input Syntax

Map compound constructors (those that may contain multiple scalars) to SEQUENCES — including SET syntaxes.

Expansion of compound constructors sometimes requires definition of "side tables," ancillary tables within the object group that supplement the main table representing the ISO/CCITT GDMO managed object class. The rules for making side tables are applied recursively and are as follows:

- For a given level of nesting, if the ISO/CCITT GDMO ASN.1 syntax is SEQUENCE (not SEQUENCE OF) or SET (not SET OF) enumerate its scalar elements "in-line" without constructing a new side table, and otherwise treat them as if scalars.

- Otherwise (SEQUENCE OF or SET OF) enumerate the scalar elements of the SEQUENCE or SET in a new side table that is a "child" of the current table. Since this may happen recursively, a side table may be a child of another side table.

- The INDEX of a child table is that of its parent concatenated with a value index that uniquely distinguishes instances within the child table.

2.3.3 Discussion

2.3.3.1 Simple Input Syntax

Internet SMI precludes a value of zero and some compilers won't take it. 32767 is a number that practically any machine architecture can support and is large enough so it should not conflict with any enumerated actually specified.

Allowing optional conceptual columns within rows is not consistent with the philosophy of RFC 1212, but neither are the MIB's the procedure seeks to translate. However, optional columns can be accommodated by SNMP using the GetNext request. In that case protocol returns inconsistent object ID prefixes for any non-present objects, rather than an error condition.

2.3.3.2 Complex Input Syntax

This is the messiest part of the translation process but cannot be avoided given that the ISO/CCITT GDMO information model is to be carried over. One way of looking at this is that constructed types are put in "third normal form," i.e., broken up into a set of flat tables each of which has a unique key. In EVERY case that key is comprised of one or more ARBITRARY indexes of type INTEGER. The class table entries have exactly one such index. Multi-valued attributes are represented as side tables that typically have two indexes: the first BEING the index of the corresponding class table entry and the second an arbitrary integer that distinguishes one attribute value from another for the multi-valued case. If a set valued ATTRIBUTE contained a multiply occurring syntax (e.g., SEQUENCE OF) that would map to a side table with three indexes of type INTEGER: first, the index of the class table entry, second, the index of the attribute value, and third, the index of the multiply occurrence of the syntax.

There is no need for explicit pointers from class table entries to side tables or vice versa. Given the index of a side table entry, one can find the corresponding class table entry since the class arc is known and the subidentifier is

also known, i.e., the first index of the side table. Likewise, knowing the arc for a side table and the index of a corresponding class entry, one can use the side table arc suffixed by the index in a GetNextRequest to discover the first entry in the side table.

There is a convention in the Internet world that the key of a table references only attributes contained within that table. The translation procedure honors that practice by defining distinct OBJECT TYPE's for all indices of side tables, although though a child table only has only one index with a different value from its parent's.

There may very well be a "natural key" for multi-valued syntax, e.g., an address or name. In this case, an artificial index may be inappropriate. Human judgment must weigh whether there is a "natural" key and whether the length of the associated subidentifier would be permissible for purposes of indexing.

> Note: It is not recommended that natural keys be used for the INDEX parameter of a class table as that will result in very long subidentifiers and complicate allocation of indexes for new object creation. Human judgment can be used to supplement class entry indices with side tables that provide secondary indices that support access based on natural keys.

There is no need to actually access OBJECT TYPES that correspond to table indices — you would have to know them first to read them, and they can't be changed. Therefore, their ACCESS clauses specify not-accessible.

2.4 Generation of Internet MIB Identifiers

2.4.1 Translation Procedure

This discussion has two parts:

- Definition of notation for mapping rules.
- Rules for name mapping, with examples.

2.4.1.1 Notation

<ASN.1 id>	Identifier of a production rule specified using Abstract Syntax Notation One (ASN.1), e.g., "AccessTimeLocationList."
<ATTRIBUTE id>	Identifier used for ATTRIBUTE template, e.g.,"Trouble ReportCancelRequestedByManager."

<MANAGEDOBJECT CLASS id>	Identifier used for MANAGED OBJECT CLASS template, e.g.,"TroubleReport."
<module prefix>	An arbitrary literal assigned to the ASN.1 module to be generated, e.g., "t1TA."
<n>	A decimal string literal indicating the dimension represented by a value index of a side table. The first dimension corresponds to the instance of the managed object (i.e., class index) and <n> is not concatenated in its name. <n> is null valued for the second dimension, which is usually the greatest dimension, i.e., the standard multi-valued attribute. Values of <n> for higher dimensions are decimal literals assigned in ascending sequence starting with "3," i.e., "3," "4," etc. Note: This option is included for completeness. This is a good example of a case where human judgment should be used before carrying over full functionality between MIB's.

Figure 3. Variables for Generating Identifiers

The following notation is used to specify mapping rules:

- Symbols enclosed in quotes are literals.

- Symbols enclosed in angle brackets ("<>") are variables that have different string values depending on instance or context. Figure 3 describes the variables.

- Double upended bars ("||") signify the concatenation operator. For this operator, literals are concatenated without modification. When concatenating a variable however, its first character of its value string is promoted to upper case. Strings are concatenated without intervening punctuation or white space to arrive at the resulting identifier.

2.4.1.2. Mapping Rules

The following table provides mapping rules for generating Internet MIB identifiers. An example is provided for each rule, based on the sample inputs and outputs of Appendix A.

Identifier	Syntax and Example

class table	<module prefix>||<MANAGED OBJECT CLASS id>||"Table" e.g., t1TATroubleReportTable
class (table)entry	<module prefix>||<MANAGED OBJECT CLASSid>||"TableEntry" e.g., t1TATroubleReportTableEntry
class entry delete flag	<module prefix> || <MANAGED OBJECT CLASSid> || "Delete" e.g., t1TATroubleReportDelete
class entry index of class table	<module prefix> || <MANAGED OBJECTCLASS id> || "Index" e.g., t1TATroubleReportIndex
conceptual row of class table	<module prefix> ||<MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || <ASN.1 id> e.g., t1TATroubleReport
side table	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id>|| <ASN.1 id>* || "Table" e.g.,t1TATroubleReportAccessTimeLocationList Table
side (table) entry	<module prefix> ||<MANAGED OBJECT CLASS id> ||<ATTRIBUTE id> || "TableEntry" e.g., t1TATroubleReportAccessTimeLocationList TableEntry
side entry delete flag	<module prefix>|| <MANAGED OBJECT CLASS id> || "Delete" e.g., t1TATroubleReportAccessTimeLocationList Delete
class entry index of side table	<moduleprefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || "ClassIndex" e.g., t1TATroubleReportAccessTimeLocationList ClassIndex
value index of side table	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || "ValueIndex" || <n> e.g., t1TATroubleReportAccessTimeLocationList ValueIndex

Figure 4. Mapping Rules for Generating Identifiers

* Note: The <ASN.1 id> is only present in the names of side table objects when its presence is necessary to unambiguously distinguish the table in question.

The identifier for the syntax for a (class or side) table entry is the same as that of the table entry itself except the first character is promoted to upper case, e.g., "T1TATroubleReportTableEntry."

2.4.2 Discussion

This approach is verbose but can be mechanized and ambiguities and collisions should be rare. It has the further advantage that names can be used for C language program variables without further manipulation.

Separating constituent ids with hyphens would increase readability and decrease likelihood of ambiguity. Unfortunately, many Internet MIB compilers do not allow this.

In cases where the same ATTRIBUTE appears in more than one PACKAGE included in a MANAGED OBJECT CLASS, manual intervention is necessary to assign distinct identifiers for the corresponding OBJECT TYPE's.

2.5 Mapping to the ACCESS clause

2.5.1 RFC 1212 Advice

This is straight-forward.

2.5.2 Discussion

Note that ADD-REMOVE and REPLACE map to "write," while GET maps to "read." There is no direct mapping to SET-TO-DEFAULT, since SNMP requires values be explicitly set for existing objects. PERMITTED VALUES are not directly mapped in the Internet MIB but need to be understood by the management station.

2.6 Mapping to the STATUS clause

2.6.1 RFC 1212 Advice

This is usually straight-forward; however, some osified-MIBs use the term "recommended." In this case, a choice must be made between "mandatory" and "optional."

2.6.2 Additional Constraints

The translation procedure always uses mandatory.

2.6.3 Discussion

Human judgment can qualify this as necessary.

2.7 Mapping to the DESCRIPTION clause

2.7.1 RFC 1212 Advice

This is straight-forward: simply copy the text, making sure that any embedded double quotation marks are sanitized (i.e., replaced with single-quotes or removed).

2.8 Mapping to the REFERENCE clause

2.8.1 RFC 1212 Advice

This is straight-forward: simply include a textual reference to the object being mapped, the document which defines the object, and perhaps a page number in the document.

2.8.2 Additional Constraints

The translation procedure transcribes the registry of the ATTRIBUTE to the corresponding OBJECT-TYPE REFERENCE clause, which ensures that proper ASN.1 syntax for the values of OBJECT IDENTIFIERs is retained. If any additional information is to be provided in this value, it must be provided to the left of the machine readable ASN.1 syntax and separated from it by a colon, e.g., "ISO 10165-2: joint-iso-ccitt ms (9) smi(3) part2(2)attribute(7)". This facilitates generation of proxies.

2.9 Mapping to the INDEX clause

2.9.1 RFC 1212 Advice

Decide how instance-identifiers for columnar objects are to be formed and define this clause accordingly.

2.9.2 Additional Constraints

Use the class entry index's for the table and any containing table, as discussed previously. This keeps the index for any particular kind of table constant and predictable.

2.10 Mapping to the DEFVAL clause

2.10.1 RFC 1212 Advice

Decide if a meaningful default value can be assigned to the object being mapped, and if so, define the DEFVAL clause accordingly.

2.10.2 Additional Constraints

Please see the previous sections on mapping of managed objects and syntaxes.

2.11 Translation of Actions

2.11.1 RFC 1212 Advice

2.11.1.1 General Advice

Actions are modeled as read-write objects, in which writing a particular value results in the action taking place.

Usually an INTEGER syntax is used with a distinguished value provided for each action that the object provides access to. In addition, there is usually one other distinguished value, which is the one returned when the object is read.

2.11.1.2 Mapping to the ACCESS clause

Always use read-write.

2.11.1.3 Mapping to the STATUS clause

This is straight-forward.

2.11.1.4 Mapping to the DESCRIPTION clause

This is straight-forward: simply copy the text, making sure that any embedded double quotation marks are sanitized (i.e., replaced with single-quotes or removed).

2.11.1.5 Mapping to the REFERENCE clause

This is straight-forward: simply include a textual reference to the action being mapped, the document which defines the action, and perhaps a page number in the document.

2.11.2 Discussion

This is one of the areas where mechanization can at best be an aid, rather than an automatic solution, to translation. The RFC 1212 advice provides a point of departure in this regard.

2.12 Translation of Notifications

2.12.1 Approach

This subsection provides a method whereby notifications are translated to traps. This method provides the basis for translation of standard Definition of Management Information (DMI) [ISO10165-2] NOTIFICATIONs to traps as reflected in the ISO/CCITT Convergence MIB of Appendix B. Since use of traps is strongly discouraged in the Internet community and there is no filtering mechanism in SNMP, such translation should be done very sparingly. In particular,

- Mapping of notifications to traps should always have a documented justification.

- Wherever such mapping is deemed necessary, the standard traps provided in Appendix B of this specification should be used, if applicable.

Where translation of NOTIFICATIONs is necessary, the following method can be used:

(1) For each input registry subtree in which there are NOTIFICATIONs or ATTRIBUTEs to be translated, establish a corresponding output registry subtree to hold the translated MIB (in the example of Appendix B, the output subtrees are onSMI2toInternetNotification and onSMI2toInternetAttributeID respectively).

A separate ASN.1 module is generated for each subtree in which NOTIFICATIONs to be translated are registered. Mappings of corresponding ATTRIBUTEs are also incorporated in each such output module, which may lead to objects (VARIABLES) of the same registry appearing in different ASN.1 modules. The registration of the resulting ASN.1 module is the same as that of the root of the output registry subtree assigned for notifications. Mnemonic naming of the output module is encouraged but outside the scope of this specification.

Assign a naming prefix for translated ATTRIBUTEs and NOTIFICATIONs. This must start with a lower case letter ("on" is the prefix for translated notifications in Appendix B).

Editor's Note: [Should we consider auto registry of the MIB module]

(2) For each notification to be mapped list its: identifier; arc within its input registry subtree; and mandatory ATTRIBUTEs. For mandatory ATTRIBUTEs, also list the corresponding syntaxes resolved to.

(3) For each syntax identified in (2), determine a mapping to the Concise MIB domain. The mapping rules are the same as for subsection 2.3 (Mapping to the SYNTAX Clause) above, but with added constraints that resulting syntax must be scalar and of fixed type (not a CHOICE). Where the rules of subsection 2.3 do not yield such a form, the syntax of DisplayString is used (as a default). This is an area in which human judgment will often be required following deterministic translation.

(4) Define an OBJECT-TYPE (variable) for each ATTRIBUTE identified in (2):

- The identifier of the resulting object is arrived at by promoting the first character of the input identifier to upper case and prepending the prefix of (1). Thus for example eventTime becomes onEventTime given the prefix "on."

- The value for the ACCESS clause is always not-accessible.

- The value of the STATUS clause is always mandatory.

- The value of the REFERENCE clause reflects the registration of the input ATTRIBUTE. The value for this clause also follows the convention for machine readability described in subsection 2.7.

- The value of the DESCRIPTION clause is taken from the BEHAVIOUR DEFINED AS parameter, if any. This value is otherwise empty. This clause will often need to be supplemented by comments, and should be manually edited if the full semantics cannot be carried over due to syntactical or other restrictions.

- The applicable syntax selected in step (3) is used for the value of the SYNTAX clause.

- The (relative) leg number for the output registry is the same as for the input registry.

(5) For each input notification, define a corresponding TRAP-TYPE:

- The identifier of the resulting trap is the same as for the notification except that its first letter is promoted to upper case and the prefix provided in (1) is prepended. For

example, "objectCreation" maps to "onObjectCreation" given the prefix "on".

- The VARIABLES parameter reflects all mandatory ATTRIBUTEs as identified in(2) and translated in (4), plus two variables that are always present: onManagedObjectInstance and onAdditionalText.

- The text of the description is the same as for the BEHAVIOUR DEFINED AS parameter of the corresponding NOTIFICATION, if any.

- The REFERENCE clause reflects the registry of the input NOTIFICATION, using the conventions as for machine readability established in subsection 2.7 above.

- The relative leg number for the output registry (ENTERPRISE clause) is the same as for the input registry.

2.12.2 Discussion

Limitations of this procedure reflect basic functional differences between the CMIP and SNMP, with much necessarily lost in translation.

In particular, SNMP Version 1 provides no mechanism for filtering traps at the source, and the Internet community frowns on the usage of traps in any case. Thus anyone translating a MIB according to this procedure should avoid translating NOTIFICATIONs without good reason. Where this cannot be avoided, only the minimum necessary functionality should be carried over — and the justification for this should be documented. Such decisions should be made on a class by class, NOTIFICATION by NOTIFICATION and even ATTRIBUTE by ATTRIBUTE basis. While translated DMI NOTIFICATIONs are provided here for the sake of completeness, it may never be appropriate to use some of these traps.

The method described in the preceding subsection can be mechanized. However, at best this will provide the human translator with default options and a point of departure for making hard choices.

The mapping of all ATTRIBUTE syntaxes to scalar types simplifies mapping of identifiers and facilitates auto registry. Notwithstanding, this approach can in certain cases be ugly and even unworkable. However, that seems to be the case for any deterministic procedure. Human judgment and intervention will often be required.

Consider for example the following. One could deal with optional syntax by defining different variables for reporting all optional forms. At the same time one could

define "null" values for each variable. Variables for all options could then be passed in a trap message, with exactly one of them non-empty.

However, specification of null values is messy and does not lend itself to automation. This also complicates assignment of identifiers and arcs, since there is a one-to-many mapping. Furthermore, passing of empty parameters is inefficient and complicates the work of a manager, which must determine which variable is "real." In any case, this approach does not address multi-valued ATTRIBUTEs — only CHOICEs.

For the translated DMI NOTIFICATIONs of Appendix B, multi-valued ATTRIBUTEs are dealt with(as necessary) by issuing separate traps for each attribute value. This is perhaps not unreasonable for a default approach. For example, it might be appropriate for reporting certain kinds of ATTRIBUTE change such as state changes. However, this approach should be used VERY, VERY SPARINGLY if at all.

The onAdditionalInformation field provides a place to put additional information otherwise lost in translation (e.g., non-mandatory or multi-valued ATTRIBUTE values). The implementor should populate this field with self-defining information that can easily be understood by operations personnel.

The onManagedObjectInstance variable is used in lieu of the ManagedObjectClass and ManagedObjectInstance parameters provided by CMIP.

2.13 Translation of Delete Operations

2.13.1 RFC 1212 Advice

Nonetheless, it is highly useful to provide a means whereby a conceptual row may be removed from a table. In MIB-II, this was achieved by defining, for each conceptual row, an integer-value columnar object. If a management station sets the value of this object to some value, usually termed "invalid," then the effect is one of invalidating the corresponding row in the table. However, it is an implementation-specific matter as to whether an agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the columnar object indicating the in-use status.

2.13.2 Discussion

To simplify mechanized translation, the DELETE operation is provided for all tables, rather than trying to determine which ones support manager-initiated DELETE operations.

2.14 Translation of Create Operations

2.14.1 RFC 1212 Advice

It is also highly useful to have a clear understanding of how a conceptual row may be added to a table. In Internet, at the protocol level, a management station issues an SNMP set operation containing an arbitrary set of variable bindings. In the case that an agent detects that one or more of those variable bindings refers to an object instance not currently available in that agent, it may, according to the rules of the SNMP, behave according to any of the following paradigms:

(1) It may reject the SNMP set operation as referring to non-existent object instances by returning a response with the error-status field set to "noSuchName" and the error-index field set to refer to the first vacuous reference.

(2) It may accept the SNMP set operation as requesting the creation of new object instances corresponding to each of the object instances named in the variable bindings. The value of each (potentially) newly created object instance is specified by the "value" component of the relevant variable binding. In this case, if the request specifies a value for a newly (or previously) created object that it deems inappropriate by reason of value orsyntax, then it rejects the SNMP set operation by responding with the error-status field set to badValue and the error-index field set to refer to the first offending variable binding.

(3) It may accept the SNMP set operation and create new object instances as described in (2) above and, in addition, at its discretion, create supplemental object instances to complete a row in a conceptual table of which the new object instances specified in the request may be a part.

It should be emphasized that all three of the above behaviors are fully conformant to the SNMP specification and are fully acceptable, subject to any restrictions which may be imposed by access control and/or the definitions of the MIB objects themselves.

2.14.2 Additional Constraints

Be very sparing in allowing Internet manager initiated object creation. A table of parents and a table of name

bindings are provided in the ISO/CCITT Convergence MIB so that a parent can be specified when creating an object.

2.14.3 Discussion

To create an object mapping into the ISO/CCITT world requires that its parent be known, hence the parent table. A child table is also provided to allow general navigation of the MIB tree. The name binding table is necessary to determine the object identifier associated with each parent/child binding.

An SNMP SetRequest needs to contain enough information to create an internally consistent object from the ISO/CCITT perspective. The SNMP PDU size restriction could be a problem here.

3 Constraints on SNMP Usage

The following constraints apply when using SNMP with a MIB translated by this procedure.

Editor's Note: [In this draft, the term "SNMP" implies SNMPv1. Support for SNMPv2 is to be added in the next draft.]

3.1 Approach

The following assumptions about use of the SNMP are made to facilitate MIB translation:

- The management station will expect that conditional attributes may not be present on a per conceptual row basis and will act appropriately, e.g., use GetNextRequest to test for presence.

- The management station will expect that access actually granted may be less than stated in the MIB specification due to dynamic access controls; in such cases it may receive error-status of readOnly — even for an SNMP GetRequest.

- A management station creating a new entry in a class or side table must first acquire an appropriate index for doing so. This is accomplished by reading a value from the appropriate row of the ocNextUniqueIndexTable provided in the "ISO/CCITT Convergence MIB" (see Appendix B subsection 2). The index of this table is an object ID, i.e., the ID of one of the tables generated by this algorithm. The output (reading the ocNextUniqueIndex conceptual column of the indicated conceptual row) is a unique integer subidentifier to be used for creating a new conceptual row in the table of interest. Typically, a different value is returned each time an ocNextUniqueIndex column is read.

The subidentifier returned by ocNextUniqueIndex is guaranteed to be unique within its table within an agent.

- Creation of a class or side table entry requires that the associated SNMP SetRequest PDU include:

  * a valid pre-allocated subidentifier for that table,

  * initial values for those attributes that must be present (which however may be allowed to default). and

  * in the case of a class table entry, class entry instance of a valid parent object to be inserted in the parent table.

  If any of these conditions are not met, noSuchName error-status is returned.

- If a management station attempts to delete an object or attribute value for which deletion is not permitted (for any reason) error-status of readOnly is returned.

- A management station must be prepared to receive badValue error-status when a SetRequest operation attempts to set an attribute to a value inconsistent with other attribute values according to the object's BEHAVIOR or PERMITTED VALUES specifications.

3.2 Discussion

To support create operations requires that the manager somehow supply a unique subidentifier. Rather than sub-allocate index ranges to different managers or administer pools on a per-table basis, it seems simplest to have a generic pool administered by the agent on behalf of all managers.

As regards error-status values, a "bending" of the rules of SNMP is necessary to map functionality not really supported in the protocol. Thus an off-the-shelf manager should be able to interoperate with an agent that implements a translated MIB but usage of the PDUs will not be entirely conventional. This is particularly true for usage of error-status.

4 Summary

Given certain assumptions about the use of SNMP (section 3) and the ancillary ISO/CCITT Convergence MIB (appendix B), this procedure allows mechanized translation of most functionality found in ISO/CCITT MIBs to the world of SNMP.

The algorithm preserves basic capability to navigate ISO/CCITT object relationships, i.e.,

- Location of parents (immediately containing objects),

- Location of children (immediately contained objects), and

- Location of referenced objects.

This approach preserves the capability to create new object instances and attribute values, even for generic network models that subsume multiple computer systems and network elements.

Areas in which significant functionality is lost in translation include:

- Notification: a minimalistic set of capabilities are provided for basic notifications in the ISO/CCITT Convergence MIB. No attempt is made to provide for filtering or logging of notifications.

- Actions: these must be dealt with manually, on a case by case basis.

- Scoping and filtering: only rudimentary tools are provided for navigating the translated MIB's using SNMP.

5 Acknowledgments

The editor wishes to express gratitude to Keith McCloghrie for his extremely timely and expert assistance with the U S West Trouble Administration translation effort, also, to Ken Hunter of Hewlett-Packard Company and Al Vincent of U S WEST Communications, Inc. who translated the MIB and made it work. These efforts served as input to the original contribution on which this document is based.

In addition, the following individuals have contributed to this effort.

Bob Aronoff	- NIST
Jon Biggar	- NetLabs
Mary Brady	- NIST
April Chang	- NetLabs
Jock Embry	- Opening Technologies
Paul Golick	- IBM
Pramod Kalyanas	- University of Delaware
Lee LaBarre	- The MITRE Corporation
David Liu	- Northern Telecom, Inc
Owen Newnan	- U S West Advanced Technologies
Steve Ng	- MPR Teltech
Yasuhiro Ohara	- NTT
George Pavlou	- UCL
Lisa Phifer	- Bellcore
Tom Rutt	- AT&T
Mark Smith	- Hewlett-Packard
Einar Stefferud	- Network Management Associates, Inc.
Dean Voiss	- NetLabs
Yoshi Yamashita	- NKK Corporation

References

[ISO8824] ISO/IEC IS 8824: Information Technology- Open System Interconnection - Specification of Abstract Syntax Notation One(ASN.1),1990.

[ISO8825] ISO/IEC IS 8825: Information Technology - Open System Interconnection-Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1),1990.

[ISO7498-4] ISO/IEC IS 7498-4, Information Processing Systems - Open Systems Interconnection -Basic Reference Model Part 4 - Management Framework, 1989.

[ISO9595] ISO/IEC IS 9595, Information Technology - Open System Interconnection- Common Management Information Service Definition, 1991.

[ISO9596-1] ISO/IEC IS 9596-1, Information Technology - Open Systems Interconnection- Common Management Information Protocol - Part 1: Specification, 1991.

[ISO10165-1] ISO/IEC IS 10165-1: Information Technology - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model, 1991.

[ISO10165-2] ISO/IEC IS 10165-2: Information Technology - Open Systems Interconnection -Structure of Management Information - Part 2: Definition of Management Information, 1992.

[ISO10165-4] ISO/IEC IS 10165-4: Information Technology - Open Systems Interconnection -Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects, 1991.

[RFC1052] RFC 1052, Cerf, V., IAB Recommendations for the Development of Internet Network Management Standards, April 1988.

[RFC1109] RFC 1109, Cerf, V., Report of the Second Ad Hoc Network Management Review Group, August 1989.

[RFC1155] RFC 1155, M. Rose and K. McCloghrie, Structure and Identification of Management Information for TCP/IP based internets, May 1990.

[RFC1157] RFC 1157, J.D. Case, M.S. Fedor, M.L. Schoffstall, C. Davin, Simple Network Management Protocol (SNMP), May 1990.

[RFC1212] RFC1212, M. Rose, K. McCloghrie - Editors, Concise MIB Definitions, March 1991.

[RFC1213] Network Management of TCP/IP-based internets: MIB-II, March 1991.

[RFC1214] RFC1214, L. LaBarre - editor, OSI Internet Management: Management Information Base, April 1991.

[RFC1215] RFC1215, M. Rose - Editor, Management A convention for Defining Traps for use with the SNMP, March 1991.

[RFC1353] RFC1353, K. McCloghrie, J.R. Davin, J.M. Galvin, Definitions of Managed Objects for SNMP Parties, July 1992.

[SNMPv2COEX] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Coexistence between version 1 and version 2 of the Internet Network Management Framework, Internet-draft, December 1992.

[SNMPv2PROT] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-draft, January 1992.

[SNMPv2SMI] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-draft, December 1992.

[SNMPv2MIB] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-draft, December 1992.

[SNMPv2TC] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-draft, December 1992.

[SNMPv2ADMIN] J.R. Davin, J.M. Galvin, K.McCloghrie, Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-Draft, January 1993.

[SNMPv2SEC] J.M. Galvin, K. McCloghrie, J.R. Davin, Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-Draft, January 1993.

[SNMPv2TM] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-Draft, January 1993.

[SNMPv2PARTY] J.D. Case, K. McCloghrie, M.T. Rose,S.L. Waldbusser, Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2), Internet-Draft, January 1993.

[IIMCIMIBTRANS] ISO/CCITT and Internet Management Coexistence (IIMC): Translation of Internet MIBs to ISO/CCITT GDMO MIBs, Draft 1 March 26,1993.

[IIMCMIB-II] ISO/CCITT and Internet Management Coexistence (IIMC): Translation of Internet MIB-II (RFC1213) to ISO/CCITT GDMO MIB, Draft 1, March 26, 1993.

[IIMCPROXY] ISO/CCITT and Internet Management Coexistence (IIMC): ISO/CCITT to Internet Management Proxy, Draft 1, March, 1993 [to be distributed].

[IIMCSEC] ISO/CCITT and Internet Management Coexistence (IIMC): ISO/CCITT to Internet Management Security, Draft 1, March 26, 1993.

[NMFMC92] NM Forum and X/Open, ISO/CCITT and Internet Management: Coexistence and Interworking Strategy, October, 1992.

[T1M192] ANSI T1M1.5, Operations, Administration and Provisioning (OAM&P)— Services for Interfaces between Operations Systems across Jurisdictional Boundaries to support Fault Management — Trouble Administration, T1LB.262R3-1991, January 13, 1992.

[USWE92] U S WEST Communications, Inc., U S WEST Network Interface Specification— MEDIACC Trouble Administration (TA), Document Number 77302,* Issue A, May 1992.

* U S WEST Business Resources, Inc., Manager—Information Release, 1801 California St., Room 1340, Denver CO 80202; 303 298 0117.

Appendix A Abridged Example

Following is a fairly brief example that illustrates some but not all aspects of the translation procedure.

The reader can find a more comprehensive example of MIB translation in [T1M192] and [USWE92], from which specifications this abridged example is adapted. The reader will note that this example is highly abridged and differs in some other respects from those two specifications.

Editor's Note: [This example is intended to be updated with an example that more fully illustrates these translation procedures (possibly another OMNIPoint 1 MIB, or, at minimum, the OMNIPoint 1 version of the ANSI T1M1 Trouble MIB).]

A.1 Input ISO/CCITT Management Information Base

```
troubleReport MANAGED OBJECT CLASS
DERIVED FROM "T1M1":top;   -- ANSI T1M1 variant of top
CHARACTERIZED BY
troubleReportPkg PACKAGE
        ATTRIBUTES
        cancelRequestedByManager GET-REPLACE
          DEFAULT VALUE

TroubleModule.troubleReportCancelRequestedByManagerDefault,
        managedObjectInstance GET,
        receivedTime GET,
        troubleFound GET;

NOTIFICATIONS
        "Rec. X.721 | ISO/IEC 10165-2 :
1992":objectCreation,
        "Rec. X.721 | ISO/IEC 10165-2 :
1992":objectDeletion,
        "T1LB-91-263R1":troubleHistoryEventNotification;;;

CONDITIONAL PACKAGES
troubleReportaccessTimeLocationListPkg PACKAGE
        ATTRIBUTES
        accessTimeLocationList GET-REPLACE;;
        PRESENT IF "an instance supports it.,",

troubleReportperceivedTroubleSeverityPkg PACKAGE
        ATTRIBUTES
        perceivedTroubleSeverity GET-REPLACE;;
        PRESENT IF "an instance supports it.";

REGISTERED AS { trMObjectClass 5};
```

```
accessTimeLocationList ATTRIBUTE
        WITH ATTRIBUTE SYNTAX
          TroubleModule.ServiceLocationList;
        BEHAVIOUR
          accessTimeLocationListBehaviour BEHAVIOUR
            DEFINED AS
"The Access Time Location list attribute identifies the set
or subset of service locations for which the Location Access
Hours attribute values are valid.";
            ;
        REGISTERED AS { trAttribute 2};

cancelRequestedByManager ATTRIBUTE
        WITH ATTRIBUTE SYNTAX
          TroubleModule.CancelRequestedByManager;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
          cancelRequestedByManagerBehaviour BEHAVIOUR
            DEFINED AS
"The Cancel Requested By Manager attribute indicates whether
the manager has initiated the process to cancel a Trouble
Report.";
            ;
        REGISTERED AS {trAttribute 12};

managedObjectInstance ATTRIBUTE
        WITH ATTRIBUTE SYNTAX
          TroubleModule.ManagedObjectInstance;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
          managedObjectInstanceBehaviour BEHAVIOUR
DEFINED AS
"The Managed Object Instance attribute indicates the CNM
Service object class instance or the GNM telecommunications
network resource instance associated with a particular
trouble report instance.";
            ;
        REGISTERED AS {trAttribute 29};

perceivedTroubleSeverity ATTRIBUTE
        WITH ATTRIBUTE SYNTAX
          TroubleModule.PerceivedTroubleSeverity;
        MATCHES FOR EQUALITY;
        BEHAVIOUR
          perceivedTroubleSeverityBehaviour BEHAVIOUR
            DEFINED AS
"The Perceived Trouble Severity attribute allows the manager
to indicate the effect of the trouble on the managed object
being reported.";
            ;
        REGISTERED AS {trAttribute 32};

receivedTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX TroubleModule.ReceivedTime;
```

```
        MATCHES FOR ORDERING;
        BEHAVIOUR
          receivedTimeBehaviour BEHAVIOUR
            DEFINED AS
"The Received Time attribute indicates the date and time
when a trouble report was entered.";
            ;
        REGISTERED AS {trAttribute 33};

troubleFound ATTRIBUTE
        WITH ATTRIBUTE SYNTAX TroubleModule.TroubleFound;
        BEHAVIOUR
          troubleFoundBehaviour BEHAVIOUR
DEFINED AS
"The Trouble Found attribute specifies an enumerated code
value, which identifies the problem resolved.  This field
will be copied into the trouble history information.";
            ;
        REGISTERED AS {trAttribute 45};



troubleHistoryEventNotification NOTIFICATION
WITH  INFORMATION SYNTAX
         TroubleModule.TroubleHistoryInfo;
REGISTERED AS                   {trNotification 1};



TroubleModule DEFINITIONS ::=
-- TroubleModule {...troubleModule(x)}
BEGIN

IMPORTS

AdditionalTroubleInfo,
CancelRequestedByManger,
CloseoutVerification,
CommitmentTime,
PerceivedTroubleSeverity,
TroubleFound,
TroubleReportNumberList,
TroubleType              FROM GNMTA

ObjectInstance           FROM CMIP-1  {joint-iso-ccitt
ms(9) cmip(1)  modules(0)  protocol(3)};

trMObjectClass OBJECT IDENTIFIER ::= { iso(1) member-body(2)
usa(840) ansi-t1-227-1992(10015) trGNM(0) objectClass(3) }

trAttribute OBJECT IDENTIFIER ::=  { iso(1) member-body(2)
usa(840) ansi-t1-227-1992(10015) trGNM(0) attribute(7) }
```

```
trNotification OBJECT IDENTIFIER ::= { iso(1) member-body(2)
usa(840) ansi-t1-228-1992(10016) trGNM(0) notification(10) }

CancelRequestedByManager ::= BOOLEAN
ManagedObjectInstance ::= ObjectInstance
PerceivedTroubleSeverity ::= INTEGER {
        outOfService(0),
        backInService(1),
        serviceAffectingTrouble(2),
        nonServiceAffectingTrouble(3)
}
PremisesAddress ::= PrintableString(SIZE(128))
PremisesName ::= PrintableString(SIZE(64))
ReceivedTime ::= GeneralizedTime
ServiceLocationList ::= SET OF SEQUENCE{
        PremisesName,
        PremisesAddress
}
TroubleFound ::= CHOICE{
     number INTEGER {
     pending(0),
     cameClear(1),
     centralOffice(2),
     customerPremisesEquipment(3),
     facility(4),
     interexchangeCarrier(5),
     information(6),
     nonplanClassified(7),
     noTroubleFound(8),
     station(9),
     servingBureau(10),
     testOK(11),
     publicServicesCoinSet(12),
     otherStationEquipment(13),
     stationWiring(14),
     centralOfficeFacility(15),
     customerOperatingInstructions(16),
     testedOKVerifiedOK(17),
     coFacilityTestedFoundOK(18),
     outsideFacilityTestedFoundOK(19),
     referredOutToOtherDept(20),
     protectiveConnectingArrang(21),
     cpeCustomerResponsibility(22)
},
identifier OBJECT IDENTIFIER
}
troubleReportCancelRequestedByManagerDefault BOOLEAN ::=
FALSE

-- Supporting Productions

TroubleAdministrationFunctionalUnits ::=  BIT STRING
{
fm-ta-kernel                    (0),
```

```
fm-ta-req-trb-rpt-format        (1),
fm-ta-trb-hist-evt- notif       (2),
fm-ta-rev-trb-hist-recd         (3),
fm-ta-add-trb-info              (4),
fm-ta-trb-rpt-up-notif          (5),
fm-ta-enrol-deenrol-notif       (6),
fm-ta-ver-trb-rep-comp          (7),
fm-ta-mod-trb-adm-info          (8)
}

TroubleHistoryInfo ::=  SEQUENCE{
     managedObjectInstance    [0]  ObjectInstance,
     receivedTime             [1]  GeneralizedTime,
     troubleFound             [2]  TroubleFound,
     additionalTroubleInfo    [3]  AdditionalTroubleInfo
                                   OPTIONAL,
     cancelRequestedByManager [4]  CancelRequestedByManager
                                   OPTIONAL,
     closeOutNarr             [5]  PrintableString OPTIONAL,
     closeoutVerification     [6]  CloseoutVerification
                                   OPTIONAL,
     commitmentTime           [7]  CommitmentTime OPTIONAL,
     custTroubleTicketNumber  [8]  PrintableString
                                   OPTIONAL,
     perceivedTroubleSeverity [9]  PerceivedTroubleSeverity
                                   OPTIONAL,
     restoredTime             [10] GeneralizedTime OPTIONAL,
     troubleReportNumberList  [11] TroubleReportNumberList
                                   OPTIONAL,
     troubleType              [12] TroubleType OPTIONAL
}

END
```

A.2 Output Internet MIB Macros

```
T1MODULE DEFINITIONS ::= BEGIN
IMPORTS OBJECT-TYPE, Counter FROM RFC1155-SMI;


t1TATroubleReportTable  OBJECT-TYPE
--   class table
               SYNTAX SEQUENCE OF
T1TATroubleReportTableEntry
               ACCESS  not-accessible
               STATUS  mandatory
               DESCRIPTION
                    "t1TATroubleReport class table."
               REFERENCE "trMObjectClass 5"
                         ::= { t1TA 5 }

t1TATroubleReportTableEntry     OBJECT-TYPE
```

```
--   class (table) entry
               SYNTAX   T1TATroubleReportTableEntry
               ACCESS  not-accessible
               STATUS  mandatory
               DESCRIPTION
                    "t1TATroubleReportTable  instance"
                            INDEX  { t1TATroubleReportIndex
}
               ::= { t1TATroubleReportTable 1 }

 T1TATroubleReportTableEntry   ::= SEQUENCE {
          t1TATroubleReportDelete
                          INTEGER,
          t1TATroubleReportIndex
                          INTEGER,
          t1TATroubleReportCancelRequestedByManager
                          INTEGER,
          t1TATroubleReportManagedObjectInstance
                          OBJECT IDENTIFIER,
          t1TATroubleReportReceivedTime
                          DisplayString,
          t1TATroubleReportTroubleFoundNumber
                          INTEGER,
          t1TATroubleReportTroubleFoundIdentifier
                          OBJECT IDENTIFIER,
          t1TATroubleReportPerceivedTroubleSeverity
                          INTEGER
                         }

t1TATroubleReportDelete   OBJECT-TYPE
--   class entry delete indicator
               SYNTAX  INTEGER
               ACCESS  write-only
               STATUS  mandatory
DESCRIPTION
"When set to zero, the corresponding entry of
thet1TATroubleReportTable is deleted."
                        ::= { t1TATroubleReportTableEntry 1
}

t1TATroubleReportIndex      OBJECT-TYPE
--   class entry index
                         SYNTAX  INTEGER
               ACCESS  not-accessible
               STATUS  mandatory
               DESCRIPTION
                    "Distinguishes unique entries of
                    t1TATroubleReportTable"
                        ::= { t1TATroubleReportTableEntry 2
}

-- Consistent with the current ANSI GNM, there are no
-- attributes associated with TOP.
-- there is one level of derivation for TroubleReport, arc
```

```
-- numbering commences at 2+10= 12 rounded up to the
-- nearest multiple of ten, i.e., with arc number 20:

t1TATroubleReportCancelRequestedByManager  OBJECT-TYPE
--   class table conceptual column
                        SYNTAX  INTEGER
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
"The Cancel Requested By Manager attribute indicates whether
the manager has initiated the process to cancel a Trouble
Report."
               REFERENCE "trAttribute 12"
               --  the following corresponds to a DEFAULT
               --  VALUE OF
               --
troubleReportCancelRequestedByManagerDefault
               -- BOOLEAN ::=  FALSE DEFVAL { 0 }
            ::= { t1TATroubleReportTableEntry 20 }

t1TATroubleReportManagedObjectInstance OBJECT-TYPE
-- recall that ObjectInstance maps to class entry instance

               SYNTAX  OBJECT IDENTIFIER
               ACCESS  read-only
               STATUS  mandatory
               DESCRIPTION
"The Managed Object Instance attribute indicates the CNM
Service object class instance or the GNM telecommunications
network resource instance associated with a particular
trouble report instance."
               REFERENCE "trAttribute 29"
                      ::= { t1TATroubleReportTableEntry 21 }

t1TATroubleReportReceivedTime  OBJECT-TYPE
                SYNTAX  DisplayString
               ACCESS  read-only
               STATUS  mandatory
               DESCRIPTION
"The Received Time attribute indicates the date and time
when a trouble report was entered."
               REFERENCE "trAttribute 33"
                      ::= { t1TATroubleReportTableEntry 22 }

-- Note that t1TATroubleReportAccessTimeLocationList is not
-- assigned arc 23 because it is implemented as a side table
-- due to its multivalued, complex syntax; see below.

-- Exactly one of the following two choices will be present
-- for a given table entry. These are enumerated in-line (in
-- the class table) rather than in a side table because the
-- syntax cannot be multi-valued.

t1TATroubleReportTroubleFoundNumber OBJECT-TYPE
```

```
                SYNTAX  INTEGER {
                pending(32767),-- value of zero not permitted
                cameClear(1),
                centralOffice(2),
                customerPremisesEquipment(3),
                facility(4),
                interexchangeCarrier(5),
                information(6),
                nonplanClassified(7),
                noTroubleFound(8),
                station(9),
                servingBureau(10),
                testOK(11),
                publicServicesCoinSet(12),
                otherStationEquipment(13),
                stationWiring(14),
                centralOfficeFacility(15),
                customerOperatingInstructions(16),
                testedOKVerifiedOK(17),
                coFacilityTestedFoundOK(18),
                outsideFacilityTestedFoundOK(19),
                referredOutToOtherDept(20),
                protectiveConnectingArrang(21),
                cpeCustomerResponsibility(22)
                }
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
"The Trouble Found attribute specifies an enumerated code
value, which identifies the problem resolved.  This field
will be copied into the trouble history information."
                REFERENCE "trAttribute 45"
                        ::= { t1TATroubleReportTableEntry 23}

t1TATroubleReportTroubleFoundIdentifier  OBJECT-TYPE
                SYNTAX  OBJECT IDENTIFIER
                ACCESS  read-only
                STATUS  mandatory
     DESCRIPTION
"The Trouble Found attribute specifies an enumerated code
value, which identifies the problem resolved.  This field
will be copied into the trouble history information."
                 REFERENCE "trAttribute 45"
                        ::= { t1TATroubleReportTableEntry 24}

t1TATroubleReportPerceivedTroubleSeverity  OBJECT-TYPE
                         SYNTAX INTEGER {
                        outOfService(32767),
                        backInService(1),
                        serviceAffectingTrouble(2),
                        nonServiceAffectingTrouble(3)
                        }
                ACCESS  read-write
                STATUS  mandatory
```

```
               DESCRIPTION
"The Perceived Trouble Severity attribute allows the manager
to indicate the effect of the trouble on the managed object
being reported"
               REFERENCE "trAttribute 32"
                       ::= { t1TATroubleReportTableEntry 25}

-- the following is a side table because it is translated
-- from a multi-valued attribute:

t1TATroubleReportAccessTimeLocationListTable   OBJECT-TYPE
--   side table
               SYNTAX  SEQUENCE OF

     T1TATroubleReportAccessTimeLocationListTableEntry
                ACCESS  not-accessible
               STATUS  mandatory

               -- this attribute is in a conditional package
               -- thus the table may be empty

DESCRIPTION
"The Access Time Location list attribute identifies the set
or subset of service locations for which the Location
Access Hours attribute values are valid."

               REFERENCE "trAttribute 2"
                        ::= { t1TATroubleReport 2 }

t1TATroubleReportAccessTimeLocationListTableEntry OBJECT-
TYPE
--   side table entry

     SYNTAX
T1TATroubleReportAccessTimeLocationListTableEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION

     "t1TATroubleReportAccessTimeLocationListTable entry."
     INDEX  {
          t1TATroubleReportAccessTimeLocationListClassIndex,
          t1TATroubleReportAccessTimeLocationListValueIndex
          }
     ::= { t1TATroubleReportAccessTimeLocationListTable 1 }

T1TATroubleReportAccessTimeLocationListTableEntry
     ::=    SEQUENCE {
     t1TATroubleReportAccessTimeLocationListDelete
               INTEGER,
     t1TATroubleReportAccessTimeLocationListClassIndex
               INTEGER,
     t1TATroubleReportAccessTimeLocationListValueIndex
               INTEGER,
```

```
     t1TATroubleReportAccessTimeLocationListPremisesName
               DisplayString,
     t1TATroubleReportAccessTimeLocationListPremisesAddress
               DisplayString
                        }

t1TATroubleReportAccessTimeLocationListDelete OBJECT-TYPE
--   side table delete indication
               SYNTAX  INTEGER
               ACCESS  write-only
               STATUS  mandatory
               DESCRIPTION
"When set to zero, the corresponding entry of the access
timelocation list table is deleted."
::= { t1TATroubleReportAccessTimeLocationListTableEntry 1 }

t1TATroubleReportAccessTimeLocationListClassIndex OBJECT-
TYPE
--   side table class index
               SYNTAX  INTEGER
               ACCESS  not-accessible
               STATUS  mandatory
               DESCRIPTION
"Has same value as class entry index of managed object."
::= { t1TATroubleReportAccessTimeLocationListTableEntry 2 }

t1TATroubleReportAccessTimeLocationListValueIndex OBJECT-
TYPE
--   side table value index
               SYNTAX  INTEGER
               ACCESS  not-accessible
               STATUS  mandatory
               DESCRIPTION
"Uniquely discriminates a value of
t1TATroubleReportAccessTimeLocationList within a managed
object"
::= { t1TATroubleReportAccessTimeLocationListTableEntry 3 }

t1TATroubleReportAccessTimeLocationListPremisesName
               OBJECT-TYPE
               SYNTAX  DisplayString
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
"The access time for a service location list premises name."
::= { t1TATroubleReportAccessTimeLocationListTableEntry 20 }

t1TATroubleReportAccessTimeLocationListPremisesAddress
               OBJECT-TYPE
               SYNTAX  DisplayString
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
```

```
"The access time for a service location list premises
address."
::= { t1TATroubleReportAccessTimeLocationListTableEntry 21 }

END
```

Appendix B ISO/CCITT Convergence MIB

This appendix has two subsections (parts):

(1) Translation of DMI Notifications to Traps. An informative part that explains how (to what extent) standard Definition of Management Information (ISO/IEC IS 10165-2) notifications are mapped to SNMP traps. This example provides guidance as to how other traps might be mapped, although that should be avoided where possible.

(2) ISO/CCITT Convergence— ASN.1 Definitions. A normative part that provides the translated MIB developed per part (1).

B.1. Translation of DMI Notifications to Traps

This subsection documents the translation of DMI notifications to traps per steps described in subsection 2.12.

The decisions per step (1) are to select the output subtrees onSMI2toInternetNotification and onSMI2toInternetAttributeID, and the prefix "on".

Table B-1 reflects the listing of notifications per the procedures of step (2). The abbreviations used for mandatory attributes in this table are defined in Table B-2.

Table B-2 itself provides the mapping of ISO/CCITT syntaxes to fixed and scalar types per step (3). This illustrates the complexity of the problem and need for human judgment.

leg	notification	syntax	mandatory attributes
1	attributeValue-Change	AttributeValue-ChangeInfo	avcd
2	communicationsAlarm	AlarmInfo	pc,ps
3	environmentalAlarm	AlarmInfo	pc,ps
4	equipmentAlarm	AlarmInfo	pc,ps
5	integrityViolation	SecurityAlarmInfo	sac,sas,sad, su,spv
6	objectCreation	ObjectInfo	(none)
7	objectDeletion	ObjectInfo	(none)
8	operational-Violation	SecurityAlarmInfo	sac,sas,sad, su,spv
9	physicalViolation	SecurityAlarmInfo	sac,sas,sad, su,spv
10	processingErrorAlarm	AlarmInfo	pc,ps
11	qualityOfServiceAlarm	AlarmInfo	pc,ps
12	relationshipChange	Relationship-ChangeInfo	rcd
13	securityServiceOr-MechanismViolation	SecurityAlarmInfo	sac,sas,sad, su,spv
14	stateChange	StateChangeInfo	scd
15	timeDomainViolation	SecurityAlarmInfo	sac,sas,sad, su,spv

Figure B-1. DMI Notifications

leg	abbreviation attribute identifier/syntax	resolves to
10	avci	attributeValueChangeDefinition/AttributeValueChangeDefinition
18	pc	probableCause/ProbableCause
17	ps	perceivedSeverity/PerceivedSeverity
20	rcd	relationshipChangeDefinition/AttributeValueChangeDefinition
21	sac	securityAlarmCause/ProbableCause
22	sad	securityAlarmDetector/SecurityAlarmDetector
23	sas	securityAlarmSeverity/SecurityAlarmSeverity
28	scd	stateChangeDefinition/AttributeValueChangeDefinitiion
24	spv	serviceProvider/ServiceUser
25	su	serviceUser/ServiceUser

Figure B-2. Mandatory Attributes

The latter half of subsection B.2 provides the output of steps (4) and (5).

```
-------------------------------------------------------
The format of this table is as follows:
< identifier > ::= < OSI Syntax >
=>   < resulting SNMP Syntax >
[Comments]
-------------------------------------------------------
AttributeValueChangeDefinition::=SET OF SEQUENCE {
     attributeID      AttributeId,
     oldAttributeValue
          [1] ANY DEFINED BY attributeID OPTIONAL,
     newAttributeValue  [2] ANY DEFINED BY attributeID}
=>   OBJECT IDENTIFIER
[Maps to OBJECT IDENTIFIER for conceptual row of ATTRIBUTE
within class table or of side table.  The oldAttributeValue
is optional thus not supported. The new value can be gotten
through polling.]
-------------------------------------------------------
 PerceivedSeverity ::= ENUMERATED{ indeterminate(0),
     critical (1), major (2), minor (3), warning (4),
     cleared (5)  }
=>   INTEGER
[Enumerated values are carried over except that zero maps to
32767.]
-------------------------------------------------------
ProbableCause  ::=  CHOICE  {
     globalValue       OBJECT IDENTIFIER,
     localValue        INTEGER}
=>   OBJECT IDENTIFIER
[The localValue is option is supported by suffixing the
integer value to the special OBJECT IDENTIFIER,
onSMI2toInternetIntegerForm.]
-------------------------------------------------------
SecurityAlarmDetector::=   CHOICE {
     mechanism      [0]   OBJECT IDENTIFIER,
     object         [1]  ObjectInstance,
     application    [2]   AE-title}
=>   OBJECT IDENTIFIER
[ObjectInstance maps to OID (class entry instance ) and
mechanism OID is passed through unaltered; there is no
ambiguity between the two.  AE Title is an alien construct
to the internet community.  This option should be supported
via an empty OID and descriptive text in the
ocAdditionalInformation variable.]
-------------------------------------------------------
ServiceUser ::= SEQUENCE {
        identifier       OBJECT IDENTIFIER,
        details          ANY DEFINED BY identifier }
=>   DisplayString
[ANYs are deprecated for SNMP.  A text string provides a
reasonable way to map this general notion.]
```

---Figure B-3. Mapping of ISO/CCITT Syntaxes to Scalars

B.2 ISO/CCITT Convergence— ASN.1 Definitions

```
ISOCCITTCONVERGENCE DEFINITIONS ::= BEGIN
IMPORTS OBJECT-TYPE, Counter FROM RFC1155-SMI;

-- Following is an ISO/CCITT Convergence MIB.
-- The purpose of this MIB is to
-- facilitate ISO/CCITT GDMO to Internet MIB translation.
-- It has two primary features:

--   (1) Relationship Management Convergence Group:
--   Parent, child and other tables that facilitate
--   navigation of containment relationships.  These
--   objects belong to one managed object group
--   that must be supported by any agent for which
--   conformance to this specification is claimed.

--   (2) Mapping of common notifications to traps. Each trap
--   listed is a separate unit of conformance and thus its
--   own object group.

-- ********************************************************
-- Relationship Management Convergence Group
-- ********************************************************

-- The following table enables manager creation of
-- class entry instances:

-- Editor's Note: [The object ocNextLocallyUniqueIndex
-- and ocNextLocallyUniqueIndex have been replaced in
-- this draft with the ocNextUniqueIndexTable, which
-- allows the agent to allocate indices on a per-class
-- basis.  Thus the manager need not determine whether
-- a class is implemented locally or globally.
-- Reviewer comment on this approach would be
-- appreciated.]

ocNextUniqueIndexTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OcNextUniqueIndexEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Provides a class table or side table index for purposes of
manager-initiated creation of rows in tables (i.e., new
managed object instances or new values for multi-valued
attributes).  Successive reads to this table return
different values that are unique within the scope of the
table within the agent.  Such values are assigned
arbitrarily by the agent, so a manager should make no
assumption about the
sequence or magnitude of values which will be returned."
::= { ocObjectCreation 1 }

ocNextUniqueIndexEntry OBJECT-TYPE
```

```
     SYNTAX OcNextUniqueIndexEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Entry of ocNextUniqueIndexTable."
     INDEX { ocNextUniqueIndexIndex }
::= { ocNextUniqueIndexTable 1 }

OcNextUniqueIndexEntry ::=
     SEQUENCE {
          ocNextUniqueIndexIndex   OBJECT IDENTIFIER,
          ocNextUniqueIndex        INTEGER
     }

ocNextUniqueIndexIndex OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Provides the class or side table ID of the table for which
a conceptual row is to be created."
::= { ocNextUniqueIndexEntry 1 }

ocNextUniqueIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
"Returns the next index to be used for creation of a
conceptual row in the indicated table.  This read-only
object
returns different values each time it is read."
::= { ocNextUniqueIndexEntry 2 }


-- Following is the parent table;
-- given the class table entry of an
-- object, it yields the class table entry for that
-- object's parent (immediately containing) object.

ocParentTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OcParentTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Allows parents in the ISO/CCITT Management naming hierarchy
to be determined.
INDEX  is OBJECT IDENTIFIER,  i.e., class entry instance ."
::= { ocObjectCreation 2 }

ocParentTableEntry OBJECT-TYPE
     SYNTAX OcParentTableEntry
     ACCESS not-accessible
     STATUS mandatory
```

```
     DESCRIPTION
          "Entry of parent table."
     INDEX { ocParent  }
::= { ocParentTable 1 }

OcParentTableEntry ::=
     SEQUENCE {
          ocParentTableIndex  OBJECT IDENTIFIER,
          ocParent            OBJECT IDENTIFIER
     }

ocParentTableIndex OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Provides the class table entry of the object in question."
::= { ocParentTableEntry 1 }

ocParent OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-write
     STATUS mandatory
     DESCRIPTION
"Provides the class table entry of the parent.  A manager
can only set this value when creating a new instance of a
managed object class, and only for those object classes for
which manager-initiated instance creation is allowed. An
empty value is returned for TOP."
::= { ocParentTableEntry 2 }

-- Following is the child table; given the class table
-- instance of an object, it yields a list of class table
-- instances for objects immediately subordinate to that
-- object.

ocChildTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OcChildTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Allows children in the ISO/CCITT Management naming
hierarchy to be determined."
::= { ocObjectCreation 3 }

ocChildTableEntry OBJECT-TYPE
     SYNTAX OcChildTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Entry of Child table."
     INDEX  { ocParentOfChild, ocChild }
::= { ocChildTable 1 }
```

```
OcChildTableEntry ::= SEQUENCE {
     ocParentOfChild OBJECT IDENTIFIER,
     ocChild INTEGER
}

ocParentOfChild OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
          "Provides the class table entry of the parent."
::= { ocChildTableEntry 1 }

ocChild OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
"This parameter is typically used in conjunction with a get
next operation to acquire class table entries for successive
child (contained) objects, given the parent.  If this value
is zero, the first child in the list is returned.  If it
is a class prefix, the first child in the given class is
returned.  If it is the full class table entry, the class
table entry of the next higher child is returned. "
::= {ocChildTableEntry 2}

-- Following is the NAME BINDING table.  It allows the  NAME
-- BINDING OBJECT IDENTIFIER of an object instance to be
-- gotten or set (can be set only on object creation).

ocNameBindingTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OcNameBindingTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Allows the NAME BINDING registration to be specified on
object creation, or fetched for an existing class entry
instance.."
::= { ocObjectCreation 4 }

ocNameBindingTableEntry OBJECT-TYPE
     SYNTAX OcNameBindingTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Entry of name binding table."
     INDEX  { ocObjectBound  } -- class entry instance
::= { ocNameBindingTable 1 }

OcNameBindingTableEntry::= SEQUENCE{
          ocObjectBound            OBJECT IDENTIFIER,
          ocNameBindingRegistry    OBJECT IDENTIFIER
     }
```

```
ocObjectBound OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
"Provides the class entry instance  of the object in
question."
::= { ocNameBindingTableEntry 1 }

ocNameBindingRegistry OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-write
     STATUS mandatory
     DESCRIPTION
"Provides the Name binding registration on object creation
and can also be fetched.  A manager can only set this value
when creating a new instance of a managed object class, and
only for those object classes for which manager-initiated
instance creation is allowed."
::= { ocNameBindingTableEntry 2 }



-- ********************************************************
-- Managed Object Groups and Trap for DMI Notifications
-- ********************************************************

-- Following are SNMP mappings of the standard DMI
-- NOTIFICATIONS.  Each of these traps and OBJECT-TYPES
-- may be implemented independent of the others,
-- thus constituting its own managed object group.

-- It is recommended these mappings to traps be used
-- very sparingly.  Where their use
-- is necessary, the following "standard"
-- traps be used where applicable:

onAttributeValueChange TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onAttributeValueChangeDefinition
          }
     DESCRIPTION
          "This notification type is used to report changes
          to the attribute such as addition or deletion of
          members to one or more set valued attributes,
          replacement of the value of one or more attributes
          and setting attribute values to their defaults."
     REFERENCE "ISO 10165-2: smi2Notification 1"
::=     1
```

```
onCommunicationsAlarm TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onProbableCause,
          onPerceivedSeverity
          }
     DESCRIPTION
          "This notification type is used to report when the
          object detects a communications error."
     REFERENCE "ISO 10165-2: smi2Notification 2"
::=     2

onEnvironmentalAlarm TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onProbableCause,
          onPerceivedSeverity
          }
     DESCRIPTION
          "This notification type is used to report  a
          problem in the environment."
     REFERENCE "ISO 10165-2: smi2Notification 3"
::=     3

onEquipmentAlarm TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onProbableCause,
          onPerceivedSeverity
          }
     DESCRIPTION
          "This notification type is used to report  a
          failure in the equipment."
     REFERENCE "ISO 10165-2: smi2Notification 4"
::=     4

onIntegrityViolation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onSecurityAlarmCause,
          onSecurityAlarmDetector,
          onSecurityAlarmSeverity,
          onServiceProvider,
          onServiceUser
          }
     DESCRIPTION
```

```
     "This notification is used to report that a
     potential interruption in information flow
     has occurred such that information may have been
     illegally modified, inserted or deleted"
     REFERENCE "ISO 10165-2: smi2Notification 5"
::=     5

onObjectCreation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES{
          onManagedObjectInstance,
          onAdditionalText
          }
     DESCRIPTION
          "This notification type is used to report the
          creation of a managed object to another
          open system."
     REFERENCE "ISO 10165-2: smi2Notification 6"
::=     6

onObjectDeletion TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES{
          onManagedObjectInstance,
          onAdditionalText
          }
     DESCRIPTION
          "This notification type is used to report the
          deletion of a managed object to another
          open system."
     REFERENCE "ISO 10165-2: smi2Notification 7"
::=     7

onOperationalViolation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onSecurityAlarmCause,
          onSecurityAlarmDetector,
          onSecurityAlarmSeverity,
          onServiceProvider,
          onServiceUser
          }
     DESCRIPTION
          "This notification is used to report that the
          provision of the requested service was not
          possible due to the unavailability, malfunction or
          incorrect invocation of the service"
     REFERENCE "ISO 10165-2: smi2Notification 8"
::=     8

onPhysicalViolation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
```

```
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onSecurityAlarmCause,
          onSecurityAlarmDetector,
          onSecurityAlarmSeverity,
          onServiceProvider,
          onServiceUser
          }
     DESCRIPTION
          "This notification is used to report that a
          physical resource has been violated in a way
          that indicates a potential security attack"
     REFERENCE "ISO 10165-2: smi2Notification 9"
::=     9

onProcessingErrorAlarm TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onProbableCause,
          onPerceivedSeverity
          }
     DESCRIPTION
     "This notification type is used to report processing
     failure in a managed object."
     REFERENCE "ISO 10165-2: smi2Notification 10"
::=    10

onQualityOfServiceAlarm TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onProbableCause,
          onPerceivedSeverity
          }
     DESCRIPTION
     "This notification type is used to report a failure
     in the quality of service of the managed object."
     REFERENCE "ISO 10165-2:
smi2Notification 11"
::=     11

onRelationshipChange TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onRelationshipChangeDefinition
          }
     DESCRIPTION
     "This notification type is used to report
```

```
     the change in the value of a relationship
     attributes of a managed object, that result
     through either internal operation of the managed object
     or via management operation."
     -- This is a subset of ISO/CCITT functionality and
     -- has been edited accordingly.
     REFERENCE "ISO 10165-2: smi2Notification 12"
::=    12

onSecurityServiceOrMechanismViolation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onSecurityAlarmCause,
          onSecurityAlarmDetector,
          onSecurityAlarmSeverity,
          onServiceProvider,
          onServiceUser
          }
     DESCRIPTION
     "This notification is used to report that a
     security attack has been detected by a security service
     or mechanism"
     REFERENCE "ISO 10165-2: smi2Notification 13"
::=    13

onStateChange TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onStateChangeDefinition
          }
     DESCRIPTION
     "This notification type is used to report the change in
     the value of a state attribute of a managed object,
     that result through either internal operation of
     the managed object or via management operation."
     -- This is a subset of ISO/CCITT functionality and
     -- the description accordingly has been manually
     -- edited.
     REFERENCE "ISO 10165-2: smi2Notification 14"
::=     14

onTimeDomainViolation TRAP-TYPE
     ENTERPRISE onSMI2toInternetNotification
     VARIABLES {
          onManagedObjectInstance,
          onAdditionalText,
          onSecurityAlarmCause,
          onSecurityAlarmDetector,
          onSecurityAlarmSeverity,
          onServiceProvider,
```

```
          onServiceUser
          }
     DESCRIPTION
     "This notification is used to report that an
     event has occurred at an unexpected or prohibited time"
     REFERENCE "ISO 10165-2: smi2Notification 15"
::=     15


-- The following objects are not accessible and exist only
-- for purposes of mapping ISO/CCITT
-- DMI attributes to the VARIABLES of traps.

-- These objects map several kinds of DMI ATTRIBUTEs
-- relating to NOTIFICATIONs:

--        Every DMI ATTRIBUTE that is mandatory for some
--        NOTIFICATION.

--        ATTRIBUTEs eventTime and managedObjectInstance,
--        which are always provided by CMIP.

--        ATTRIBUTE additionalText, which is listed in the
--        VARIABLES parameter for all translated
--        NOTIFICATIONs, even though it is generally
--        optional in ISO/CCITT usage. This allows
--        additional information to be conveyed that might
--        otherwise be lost when translating to SNMP

-- In the DESCRIPTION clauses of these objects, the term
-- "attribute" is used synonomously with corresponding
-- SNMP trap VARIABLEs.

onAdditionalText  OBJECT-TYPE
     SYNTAX DisplayString
     -- Editor's Note: [Should this be OCTET STRING?
     -- The ISO form is GraphicString.]
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute is used to specify  additional
     textual information in NOTIFICATIONs "
     REFERENCE "ISO 10165-2: smi2AttributeID 7"
::= {onSMI2toInternetAttributeID 7}

onAttributeValueChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute contains the OBJECT IDENTIFIER of
     an ATTRIBUTE whose change has been detected."
     -- This is a subset of ISO/CCITT functionality, as
     -- discussed in the informational part of this annex.
```

```
     -- The description clause is therefore entered manually
     -- rather than copied verbatim.
     REFERENCE "ISO 10165-2: smi2AttributeID 10"
::= {onSMI2toInternetAttributeID 10}

onPerceivedSeverity  OBJECT-TYPE
     SYNTAX INTEGER
     -- indeterminate (32767), critical (1), major (2),
     -- minor (3), warning (4), cleared (5)
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION ""
     -- no ISO/CCITT BEHAVIOR description exists
     REFERENCE "ISO 10165-2: smi2AttributeID 17"
::= {onSMI2toInternetAttributeID 17}

onProbableCause  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION ""
     -- no ISO/CCITT BEHAVIOR description exists
     -- defined in ISO 10164-4
     REFERENCE "ISO 10165-2: smi2AttributeID 18"
::= {onSMI2toInternetAttributeID 18}

onRelationshipChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute provides the OBJECT IDENTIFIER for
     a relationship ATTRIBUTE that has changed."
     -- This is a subset of ISO/CCITT functionality, as
     -- discussed in the informational part of this annex.
     -- The description clause is therefore entered manually
     -- rather than copied verbatim.
     REFERENCE "ISO 10165-2: smi2AttributeID 20"
::= {onSMI2toInternetAttributeID 20}

onSecurityAlarmCause  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute specifies the cause of the
     security alarm"
     REFERENCE "ISO 10165-2: smi2AttributeID 21"
::= {onSMI2toInternetAttributeID 21}

onSecurityAlarmDetector  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
```

```
     DESCRIPTION
     "This attribute identifies the entity that
     detected the security alarm"
     REFERENCE "ISO 10165-2: smi2AttributeID 22"
::= {onSMI2toInternetAttributeID 22}

onSecurityAlarmSeverity  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute indicates the severity of the
     security alarm"
     REFERENCE "ISO 10165-2: smi2AttributeID 23"
::= {onSMI2toInternetAttributeID 23}

onServiceProvider OBJECT-TYPE
     SYNTAX DisplayString
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute contains information about the
     service provider associated with the service request
     that caused the security alarm"
     REFERENCE "ISO 10165-2: smi2AttributeID 24"
::= {onSMI2toInternetAttributeID 24}

onServiceUser OBJECT-TYPE
     SYNTAX DisplayString
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute contains information about the
     service user associated with the  service request that
     caused the security alarm"
     REFERENCE "ISO 10165-2: smi2AttributeID 25"
::= {onSMI2toInternetAttributeID 25}

onStateChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "This attribute contains the OBJECT IDENTIFIER of
     a state ATTRIBUTE that has changed."
     -- This is a subset of ISO/CCITT functionality, as
     -- discussed in the informational part of this annex.
     -- The description clause is therefore entered manually
     -- rather than copied verbatim.
     REFERENCE "ISO 10165-2: smi2AttributeID 28"
::= {onSMI2toInternetAttributeID 28}

onManagedObjectInstance    OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER -- class entry instance
```

```
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION ""
     -- Provides class entry instance  (managed object
     -- instance) of the object issuing the notification.
     -- This is used in lieu of the CMIP ManagedObjectClass
     -- and ManagedObjectInstance parameters.
     REFERENCE "ISO 10165-2: smi2AttributeID 61"
::= {onSMI2toInternetAttributeID 61}


--   This document is allocated the following
--   registration identifier for purposes of referencing
--   material contained herein.

     iimcOMIBTRANS OBJECT IDENTIFIER ::=
                                    {iimcManagementDocMan 5}

--   Editor's Note: [The iimcManagementDocMan will be
--   resolved before the final publication of this
--   document.]

--   Editor's Note: [The following arcs assignments
--   have been fabricated in order to check syntax of
--   this module.  It is anticipated the NMF will
--   register appropriate arcs upon approval of this
--   specification.]

onSMI2toInternetAttributeID OBJECT IDENTIFIER  ::=
                                    { iimcOMIBTRANS 1 }
onSMI2toInternetNotification OBJECT IDENTIFIER ::=
                                    { iimcOMIBTRANS 2 }
onSMI2toInternetIntegerForm OBJECT IDENTIFIER      ::=
                                    { iimcOMIBTRANS 3 }
ocObjectCreation OBJECT IDENTIFIER            ::=
                                    { iimcOMIBTRANS 4 }

-- supplies integer form via object ID, the integer
-- provided as the leg immediately subordinate to
-- this subtree.

END
```