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

October 9, 1992

Owen Newnan

U S WEST Advanced Technologies
4001 Discovery Lane
Suite 190
Boulder, Colorado 80303
onewnan@uswest.com
303 541-6253 fax x6250

Status of this Memo

This memo provides information to the network and systems management community. This memo is not proposed to be a standard, but is intended as a contribution to ongoing work in the area of multi-protocol management coexistence and interworking. This memo is part of a package of ISO/CCITT and Internet Management Coexistence (IIMC) drafts; see also [IIMCIMIBTRANS], [IIMCMIB-II], [IIMCPARTY], and [IIMCPROXY].

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.

Distribution of this memo is unlimited. Comments on this memo should be sent to iimc@thumper.bellcore.com by November 20, 1992.

Abstract

This memo is intended to facilitate the use of ISO/CCITT CMI for integrated management of networks via proxy management of TCP/IP networks that are managed using SNMP. This memo 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). This approach accommodates both, protecting investment already made in management information specifications and minimizing cost to implement a second protocol where the market requires. The translation is designed to: 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; and support generic network models that span many computer platforms and network elements. It has been contributed the network and systems management community 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
Abstract ii
Table of Contents ii
1. Introduction 4
1.1 Background 4
1.2 Overview 5
1.3 Scope 8
1.4 Terms and Conventions 8
2. Translation Procedures 9
2.1 Relationship to RFC 1212 9
2.2 Mapping of Managed Object Classes and Attributes 10
2.3 Mapping to the SYNTAX clause 16
2.4 Generation of Internet MIB Identifiers 19
2.5 Mapping to the ACCESS clause 22
2.6 Mapping to the STATUS clause 22
2.7 Mapping to the DESCRIPTION clause 22
2.8 Mapping to the REFERENCE clause 23
2.9 Mapping to the INDEX clause 23
2.10 Mapping to the DEFVAL clause 23
2.11 Translation of Actions 23
2.12 Translation of Notifications 24
2.14 Translation of Create Operations 25
3 Constraints on SNMP Usage 26
3.1 Approach 26
3.2 Discussion 27
4 Summary 28

5 Acknowledgments 28
Appendix A: Abridged Example 31
A.1 Input: ISO/CCITT Management Information Base 31
A.2 Output: Internet MIB Macros 35
Appendix B:Abridged ISO/CCITT Compatibility MIB 41

1. Introduction

1.1 Background

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

- Translation of Internet MIBs to ISO/CCITT GDMO MIBs (LaBarre) [IIMCIMIBTRANS]
- Translation of Internet MIB-II (RFC1213) to ISO/CCITT GDMO MIB (LaBarre) [IIMCMIB-II]
- Translation of Internet Party MIB (RFC1353) to ISO/CCITT GDMO MIB (LaBarre) [IIMCPARTY]
- ISO/CCITT-Internet Management Proxy (Chang) [IIMCPROXY]

These memos together comprise a package aimed at integrating ISO/CCITT-based and Internet-based management systems. These memos are offered as input to coexistence and interworking efforts underway throughout the industry, including organizations such as:

- IETF OSI Internet Management (OIM),
- Network Management Forum Technology Convergence Team,
- X/Open Systems Management (SysMan),
- OIW Network Management Special Interest Group (NMSIG), and
- OSF Management Special Interest Group (MANSIG).

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 memos included in the IIMC package are intended as detailed specifications which implement several of the methodologies identified in this strategy.

1.2 Overview

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.

Standardisation 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 fora 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 (see Figure 1).

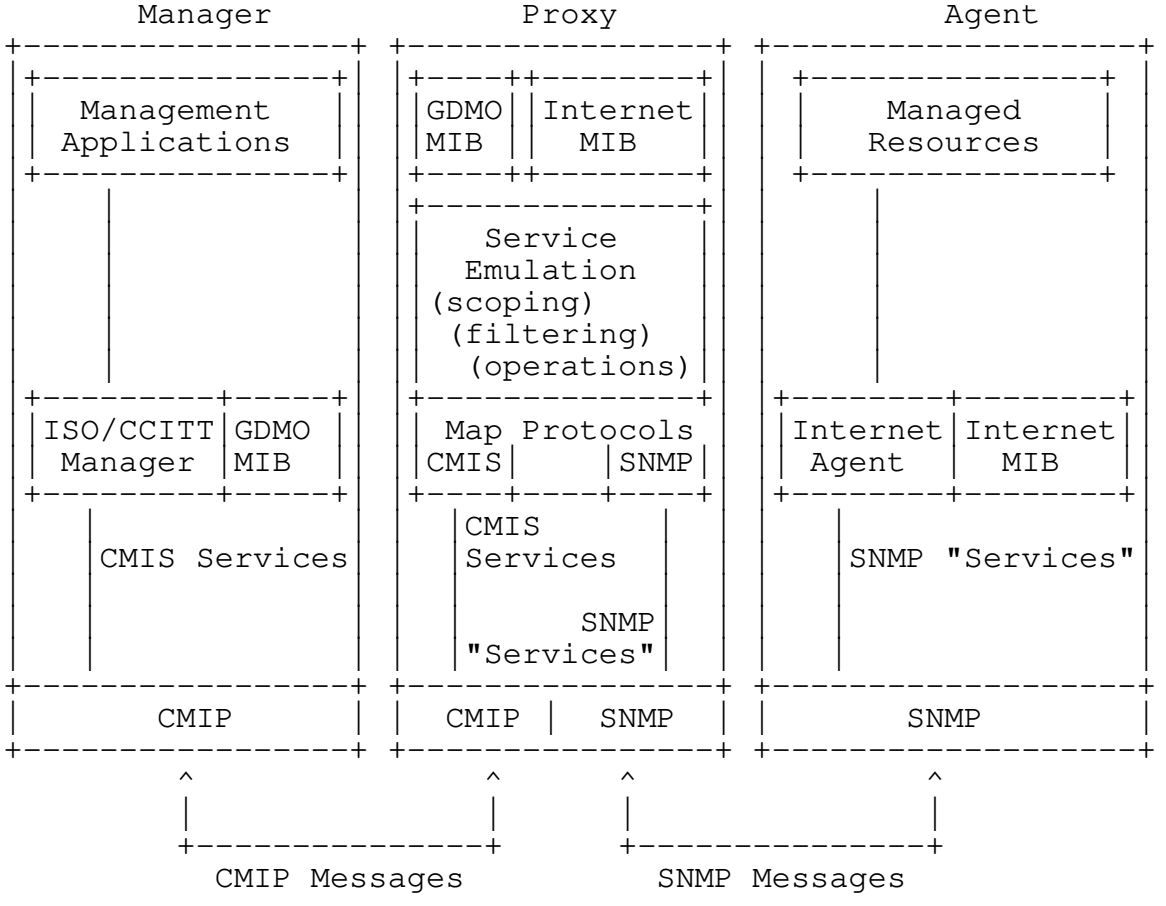

Figure 1. Proxy Architecture

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 MIBs have been translated using heuristic procedures. Currently, the proxy defined in [IIMCPROXY] is based on the reverse MIB translation procedures defined in [IIMCIMIBTRANS]. MIBs generated by this translation process described in this memo cannot be utilized by the Proxy defined in [IIMCPROXY], although

another kind of Proxy could be defined for this purpose in the future.

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 contribution.

1.3 Scope

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 memo 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.

A sample input and translated output 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 a future update of this memo.

- This procedure is based on current Internet SMI standards, but should be extended to also cover proposed SNMP-2 (SMP) SMI standards.
- This procedure currently provides only limited support for translation of notifications. Additional support should be provided, including one-time translation of all generic notifications commonly found in ISO/CCITT GDMO objects.
- An implementation of this translation procedure is now underway as proof of concept and validation. It is expected that during implementation, the need for certain input "pragmas" may be identified, in order to more fully automate the translation process by inclusion of machine-readable ASN.1 comments in the ISO/CCITT GDMO MIB input specification.

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 nodes having 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 RFC 1212

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 RFC 1212. 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 optionals.

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.[1]

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.[2]

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

---

[1] Assignment of naming prefixes and registry subtrees are required activities, however, procedures for these are outside the scope of this paper.

[2] Since all ATTRIBUTE's expand into class tables or side tables, class groups generated by the algorithm never contain scalar values.

For the entries of each class table, begin by allocating the following conceptual rows 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.[3] 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.[4]

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.[5]

2.2.2.2 Mapping of PACKAGE's

---

[3] As discussed later, no direct translation of Distinguished Names is attempted.

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

[5] The mapping of multiply occuring 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.

In reading this section the reader may wish to refer to Figures 2 and 3, 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.

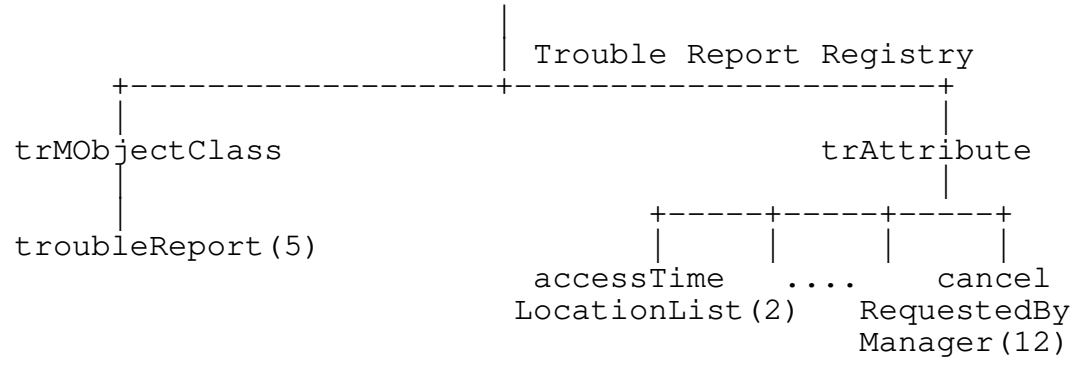

Figure 2. Sample Input Registration Tree

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.

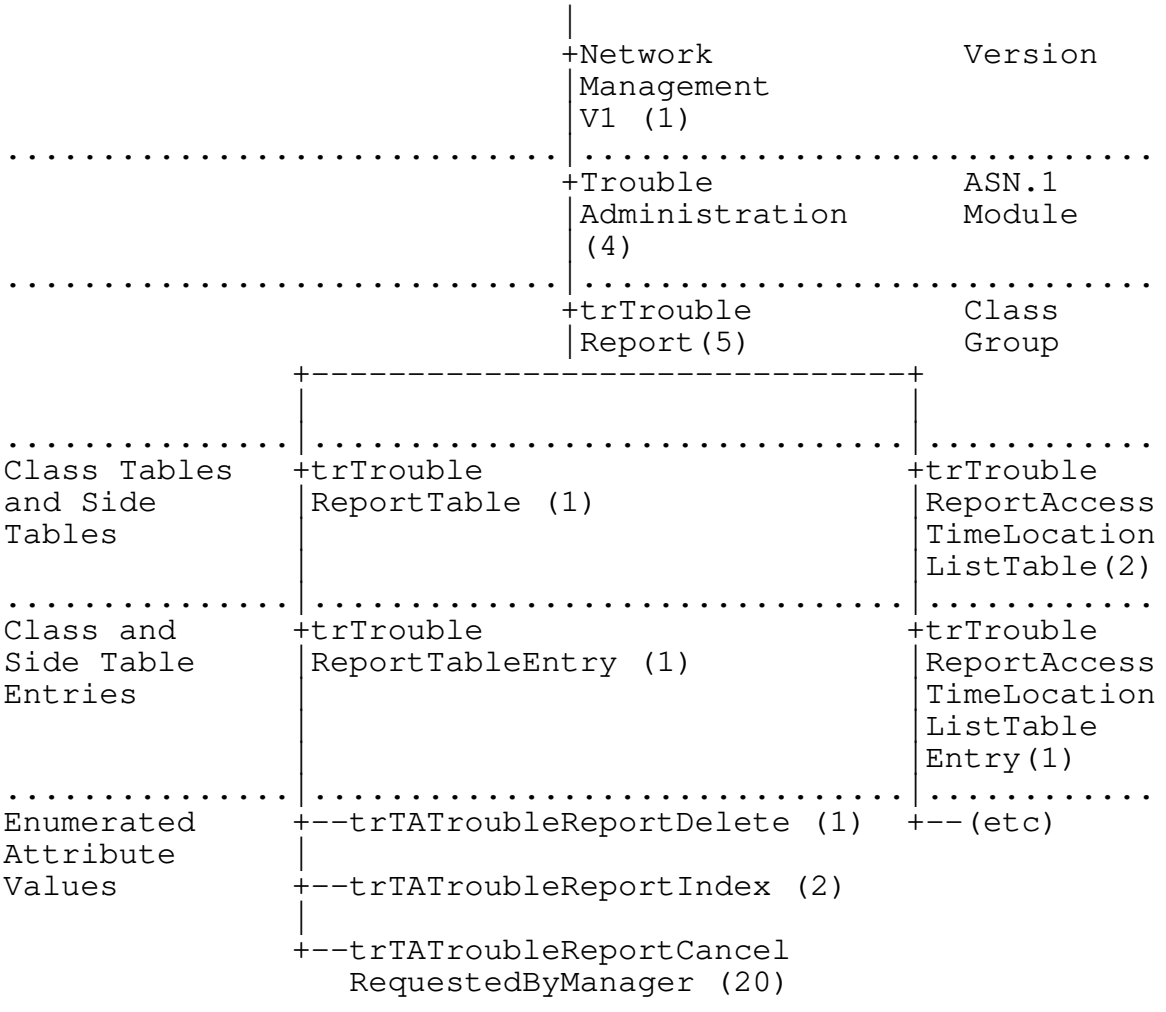

Figure 3. Sample Output Registration Tree

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 AttributeType's as well as AttributeValue'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.

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 request and determining whether the conceptual rows returned are the same as expected. No special attribute is needed for this purpose.

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 OCTET STRING, 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 OCTET STRING 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 enum 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.

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.[6]

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."
<MANAGED OBJECT 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."

[6] 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 judgement can be used to supplement class entry indices with side tables that provide secondary indices that support access based on natural keys.

<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.[7]

Figure 4. 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 4 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 CLASS id>||"TableEntry" e.g., t1TATroubleReportTableEntry

[7] This option is included for completeness. This is a good example of a case where human judgement should be used before carrying over full functionality between MIB's.

class entry delete flag	<module prefix> || <MANAGED OBJECT CLASS id> || "Delete" e.g., t1TATroubleReportDelete
class entry index of class table	<module prefix> || <MANAGED OBJECT CLASS 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>8 || "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	<module prefix> || <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 5. Mapping Rules for Generating Identifiers

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."

---

[8] 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.

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 (automatically) provides the name and object identifier of the attribute.

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

Notifications are not translated algorithmically. However, traps corresponding to standard notifications, are provided in the ISO/CCITT Compatibility MIB defined in Appendix B. These are only used for a managed object if the notification is specified for its object class in the base standard, and then if there is a clear need (since there is no event filtering mechanism in SNMP). Objects corresponding to mandatory parameters of the ISO/CCITT notifications are also included in the ISO/CCITT Compatibility MIB. These objects are referenced by the VARIABLES clauses of traps. An additional object in this group, ocReferencedInstance, gives the class entry instance of the object the notification is about.

2.12.2 Discussion

These are not translated automatically but are dealt with on a case by case basis. The ISO/CCITT Compatibility MIB provides mapped subsets of the most common types of notification.

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 or syntax, 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 Compatibility 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.

3.1 Approach.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 the value of either the ocNextLocallyUniqueIndex object or ocNextGloballyUniqueIndex object in the "ISO/CCITT Compatibility MIB" (listed in the appendices). Both objects return a unique subidentifier when read (i.e., a different value each time). The subidentifier returned by ocNextLocallyUniqueIndex is guaranteed to be unique

within an agent, while a subidentifier returned by the ocNextGloballyUniqueIndex is guaranteed unique across the network.

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

  -- a valid preallocated 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 Compatibility 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 Compatibility 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 author wishes to express gratitude to Keith McCloghrie for his extremely timely and expert assistance with the 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. In addition, thanks are due to the following individuals who participated in the generation of the IIMC package:

Lisa Phifer - Bellcore
April Chang - NetLabs
Jock Embry - Opening Technologies
Steve Ng - MPR Teltech
Lee LaBarre - Mitre

References

[ISO8824] ISO/IEC IS 8824: Information Technology - Open System Interconnection - Specification of Abstract Syntax NotationOne (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 SystemInterconnection - CommonManagement Information Service Definition, 1991.

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

[ISO10165-1] ISO/IEC IS 10165-1: Information Technology - Open Systems Interconnection - Structureof 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] RFC1155, M. Rose and K. McCloghrie, Structure and Identification of ManagementInformation for TCP/IP based internets, May 1990.

[RFC1157] RFC 1157, J.D. Case, M.S. Fedor, M.L. Schoffstall, C. Davin, Simple NetworkManagement 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 usewith the SNMP, March 1991.

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

[SMPCOEX] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, Coexistance between the Internet-standard

Network Management Framework and the Simple Management Protocol (SMP)Framework, Internet-draft, October 1992.

[SMPPROT] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, ProtocolOperations for the Simple Management Protocol (SMP) Framework, Internet-draft, October 1992.

[SMPSMI] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, ProtocolStructure of Management Information for the Simple Management Protocol (SMP) Framework, Internet-draft, October 1992.

[SMPMIB] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, ManagementInformation Base for the Simple Management Protocol (SMP) Framework, Internet-draft, October1992.

[SMPTC] J.D. Case, K. McCloghrie, M.T. Rose, S.L. Waldbusser, TextualConventions for the Simple Management Protocol (SMP) Framework, Internet-draft, October1992.

[IIMCIMIBTRANS] L. LaBarre, ISO/CCITT Integrated Management (OIM): Translation of Internet MIBs to ISO/CCITT GDMO MIBs, October, 1992.

[IIMCPARTY] L. LaBarre, ISO/CCITT and Internet Management Coexistence:Translation of Internet Party MIB (RFC1353) to ISO/CCITT GDMO MIB, October 1992.

[IIMCMIB-II] L. LaBarre, ISO/CCITT and Internet Management Coexistence:Translation of Internet MIB-II (RFC1213) to ISO/CCITT GDMO MIB, October 1992.

[IIMCPROXY] A. Chang, ISO/CCITT and Internet Management Coexistence: ISO/CCITT to Internet Management Proxy, October 1992.

[NMFMC92] Network Management Forum: Forum TRxxx, ISO/CCITT and Internet Management Coexistence, NM Forum, Issue 1.0, Draft 6, 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,[9]* Issue A, May 1992.

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 specfications 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.

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};
```

---

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

```
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 hasinitiated 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 classinstance or the GNM
            telecommunications network resource instance
            associated with aparticular 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 ::= { tbd }
-- T1M1 has not defined actual OID
trAttribute OBJECT IDENTIFIER ::= { tbd }
-- T1M1 has not defined actual OID
```

```
trNotification OBJECT IDENTIFIER ::= { tbd }
-- T1M1 has not defined actual OID

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 the
              t1TATroubleReportTable 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 concepual 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 willbe 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 time location 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
            listpremises address."
::= { t1TATroubleReportAccessTimeLocationListTableEntry 21 }
```

```
END
```

Appendix B Abridged ISO/CCITT Compatibility MIB

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

-- Following is an abridged ISO/CCITT Compatibility MIB.
-- The purpose of the ISO/CCITT compatibility MIB is to
-- facilitate ISO/CCITT GDMO to Internet MIB translation.
-- It has two primary features:

-- Parent and child tables that navigation of containment
-- relationships.
-- Mapping of common notifications to traps.

-- The following should be self-explanatory with sufficient
-- comments added:

ocNextLocallyUniqueIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "Provides a unique integer sub-identifier 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 object
            return different values that are unique
            within the scope of the agent.  Such values may be
            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 }

ocNextGloballyUniqueIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
            "Like ocNextLocallyUniqueIndex, provides a unique
            integer sub-identifier for purposes
            of manager-initiated object creation.  However,
            the  number assigned is guaranteed to
            be unique within a (private) administrative
            domain, permitting creation of objects
            that are visible across multiple Internet agents
            in that domain."
```

```
::= { ocObjectCreation 2 }

-- Following is the parent table; given the SNMP-DN of an
-- object, it yields the SNMP-DN 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.,  SNMP
            Distinguished Name."
::= { ocObjectCreation 3 }

ocParentTableEntry OBJECT-TYPE
     SYNTAX OcParentTableEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Entry of parent table."
     INDEX { ocParent  }
::= { ocParentTable 1 }

OcParentTableEntry ::=
     SEQUENCE { ocParent OBJECT IDENTIFIER }

ocParent OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-write
     STATUS mandatory
     DESCRIPTION
            "Provides the SNMP Distinguished Name of the
            parent.  An 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."
::= { ocParentTableEntry 1 }

-- Following is the child table; given the SNMP-DN of an
-- object, it yields a list of SNMP-DNs
-- 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 hierearchy to be determined."
```

```
::= { ocObjectCreation 4 }

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 SNMP-DN 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 SNMP-DNs 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 SNMP-DN, the SNMP-DN 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  { OBJECT IDENTIFIER } -- class entry instance
::= { ocParentTable 1 }

OcParentTableEntry ::= SEQUENCE
       { ocNameBindingRegistry OBJECT IDENTIFIER }

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 1 }

-- A limited mapping of notifications to traps is provided
-- using the objects defined below.   These should be
-- registered in a subtree administered by some neutral
-- organization, if not in subtrees administered by the
-- originating organization which owns the base standard
-- being translated.  This is an abridged set which continues
-- the TroubleReport example.


-- The following traps correspond to joint ISO-CCITT
-- notifications, i.e.,as defined in ISO 10165-2.  Integers
-- in the range of 1-100 are reserved for traps corresponding
-- to such notifications.  These should be defined in
-- a subtree administered by some neutral organization, if
-- not in subtrees administered by the originating
-- organization

onObjectCreation TRAP-TYPE
     ENTERPRISE neutralForum
     VARIABLES { onReportedInstance }
     DESCRIPTION "Per ISO IS 10165-2"
     REFERENCE "ISO 10165-2 smi2Notification 6"
::=     6

onObjectDeletion TRAP-TYPE
     ENTERPRISE neutralForum
     VARIABLES { onReportedInstance }
     DESCRIPTION "Per ISO IS 10165-2"
     REFERENCE "ISO 10165-2 smi2Notification 7"
::=     7
```

```
-- ISO Only Series, range 100-199

-- CCITT Only Series, range 200-299

-- ANSI Committee T1 Series, range 300-399

onTroubleHistoryEventNotification TRAP-TYPE
ENTERPRISE neutralForum
VARIABLES {
     onReportedInstance,
     onReceivedTime,
     -- exactly one of the following two
     -- is meaningful for a particular
     -- onTroubleHistoryEventNotification trap:
     onTroubleFoundNumber,
     onTroubleFoundIdentifier
}
DESCRIPTION "Per ANSI GNMTA."
::=     300

-- The following objects are not accessible and exist only
-- for purposes of specifying variables for traps. With the
-- exception of reportedInstance, they correspond to
-- ISO/CCITT notification parameters of the same name which
-- are mandatory for one of more of the notification types
-- mentioned above.

-- In all cases, the reportedInstance object is used to
-- identify the SNMP Distinguished Name corresponding to the
-- ISO/CCITT managed object instance and Internet class table
-- entry for which an event is being reported.

onReportedInstance OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Reports the SNMP Distinguished Name for which a
          notification trap is being reported."
::= {onObjectNotification 100}

onProbableCause OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          "Per ISO 10165-2.  Value assignments for probable
          cause are as specified in Attribute-ASN1Module
          {joint-iso-ccitt ms(9) smi(3)
          part2(2) asn1Module(2) 1}"
     REFERENCE "ISO 10165-2 smi2AttributeID 53"
::= {onObjectNotification 101}
```

```
onReceivedTime OBJECT-TYPE
     SYNTAX OCTET STRING -- GeneralizedTime
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "As described in ANSI GNMTA"
     REFERENCE "ANSI GNMTA trAttribute 33"
::= {onObjectNotification 102}

onTroubleFoundNumber OBJECT-TYPE
SYNTAX INTEGER -- (enumerated)

      -- value of zero indicates OID being used instead
      -- pending(32767),
      -- 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)
      -- OBJECT IDENTIFIER choice not supported
      -- in Internet MIB translation
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "As described in ANSI GNMTA"
     REFERENCE "ANSI GNMTA trAttribute 45"
::= {onObjectNotification 103}

onTroubleFoundIdentifier OBJECT-TYPE
-- empty object id indicates INTEGER choice being used
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
     "As described in ANSI GNMTA"
     REFERENCE "ANSI GNMTA trAttribute 45"
::= {onObjectNotification 104}
```

```
neutralForum OBJECT IDENTIFIER ::= { experimental 1 }

END
```