ISO/CCITT and Internet Management Coexistence (IIMC):

Translation of ISO/CCITT GDMO MIBs to Internet MIBs

(IIMCOMIBTRANS)

Draft 2
May 26, 1993

Owen Newnan (Editor)

U S WEST Advanced Technologies
4001 Discovery Drive, Suite 190
Boulder, Colorado 80303
onewnan@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 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 the cost of dual protocol and proxy-based implementations; and support generic network models which span many computer platforms and network elements. This document is 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
Abstract..... ii
Table of Contents..... ii
Revision History..... iv
1 Introduction..... 1
1.1 Problem Statement..... 1
1.2 Overview of IIMC..... 2
1.3 MIB Translation Procedures..... 2
1.4 Native Management Model..... 3
1.5 Proxy Management Model..... 5
1.6 Scope of this Document..... 6
1.7 Terms and Conventions..... 7
2. SNMPv1 Translation Procedures..... 9
2.1 Relationship to RFC1212..... 9
2.2 Mapping of Managed Object Classes and Attributes..... 10
2.3 Mapping to the SYNTAX clause..... 18
2.4 Generation of Internet MIB Identifiers..... 22
2.5 Mapping to the ACCESS clause..... 27
2.6 Mapping to the STATUS clause..... 27
2.7 Mapping to the DESCRIPTION clause..... 27
2.8 Mapping to the REFERENCE clause..... 28
2.9 Mapping to the INDEX clause..... 28
2.10 Mapping to the DEFVAL clause..... 28
2.11 Translation of Actions..... 29
2.12 Translation of Notifications..... 30
2.13 Translation of Delete Operations..... 33
2.14 Translation of Create Operations..... 34
3. Constraints on SNMPv1 Protocol Usage..... 38
4. SNMPv2 Translation Procedures..... 39
4.1 General Approach..... 39

4.2 Object Definitions 40
4.3 Trap Definitions 43
5. Constraints on SNMPv2 Protocol Usage 44
5.1 Approach 44
5.2 Discussion 45
6. Summary 45
7. Acknowledgments 46
References 48
Appendix A (Informative): Abridged Example 50
A.1 Input ISO/CCITT Management Information Base 50
A.2 Output SNMPv1 Management Information Base 54
A.3 Output SNMPv2 Management Information Base 64
Appendix B (Normative): Definition of Management Information 76
B.1 Notes on ISO/CCITT DMI Translation 76
B.2 SNMPv1 Translation of DMI 80
B.3 SNMPv2 Translation of DMI 90
Appendix C (Normative): Support Objects For Use With Translated MIBs 101

Revision History

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

Draft 1 - March 26, 1993
Previous draft of this document (replaced Draft 0).

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

Major Changes Since Last Revision

1. Removed convergence MIB, and revised structure and content of the conceptual rows generated in translation:
   - Replaced StatusDelete objects with SNMPv2 RowStatus.
   - Moved containment information from the compatibility MIB into the class object groups.
   - Assigned column OIDs sequentially, beginning with one.
   - Revised name translation and table generation procedures.
2. Clarified enumerated usage.
3. Restructured document to accommodate SNMPv2:
   - Revised requirements to minimize differences between v1 and v2 mappings.
   - Redefined class entry instance to class InstancePointer, reflecting SNMPv2 InstancePointer conventions.
   - Added a table showing how common constructs (types and conventions) of the SNMPv2 are mapped to/from OSI usage. Where practical, adjusted the v1 output to anticipate/align with v2 usage.
   - Specified read-create for generated table entries.
   - Reflected ranges and permitted values from the input MIB.
   - Added discussion of constraints on v2 protocol usage
   - Both annexes revised with subsections providing SNMP v2 MIBs.
4. Clarified SET OF CHOICE usage.
5. Incorporated new introduction and clarified the distinction between MIB translation and supporting conventions, constraints and objects.
6. Better aligned style and use of terms with other IIMC documents (arcs vs. legs, labels versus identifiers) and NMF usage (normative and informative appendices).
7. Updated SNMPv2 references to the final RFCs.
8. Changed auto-registry and made other adjustments to accommodate common SNMP parsers.

Outstanding Issues

1. Further investigation to determine whether a more comprehensive and deterministic mapping of notifications is feasible and appropriate. Concurrently, a complete and normative translation of DMI [ISO10165-2] should be attempted to provide a more comprehensive example translated MIB.

2. Conformance and compliance, and registry of translated MIBs, are yet to be addressed. Use of the DESCRIPTION clause to define textual conventions and additional constraints should be considered.

3. The mapping for ANY and CHOICE syntaxes requires further study; comments are solicited during review.

1 Introduction

This section provides an overview of ISO/CCITT and Internet Management Coexistence (IIMC) activities, insight into the problem being addressed by IIMC, and a brief introduction to the strategy adopted by IIMC: use of translated MIBs in either a proxy or native implementation. The section concludes by describing the scope of this document, and terms and conventions used by this document.

1.1 Problem Statement

The need for enterprise network management has been addressed by development of network management standards within various communities, most notably the ISO/CCITT and Internet communities.

- The ISO/CCITT community developed the Common Management Information Protocol (CMIP) [ISO9596-1], and related SMI documents [ISO10165-1,2,4].

- The Internet community developed the Simple Network Management Protocol (SNMP) [RFC1157], and its successor, SNMPv2 [RFC1448]. The Internet SMI is defined in [RFC1155] and [RFC1442].

These standards share a nearly common management model, but diverge due to differing management philosophies. Although functionally similar, the Internet and ISO/CCITT protocols and SMIs differ in terms of their complexity and specific operations. Business requirements for end-to-end enterprise management include 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.

For example, many 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 enterprise management model on the ISO/CCITT management model. These organizations are particularly interested in integrated management of devices that use the Internet management. This interest is primarily due to the widespread commercial implementation and use of such devices, especially devices that use the Internet TCP/IP protocol suite.

1.2 Overview of IIMC

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

[IIMCIMIBTRANS]	Translation of Internet MIBs to ISO/CCITT GDMO MIBs
[IIMCOMIBTRANS]	Translation of ISO/CCITT GDMO MIBs to Internet MIBs
[IIMCMIB-II]	Translation of Internet MIB-II (RFC1213) to ISO/CCITT GDMO MIB
[IIMCPROXY]	ISO/CCITT to Internet Management Proxy
[IIMCSEC]	ISO/CCITT to Internet Management Security

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 (AIII) technical team.

The IIMC intends to address the problem 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" [NMFTR107]. The documents included in the IIMC package are the next level of detailed specifications which implement several of the methodologies identified in the strategy.

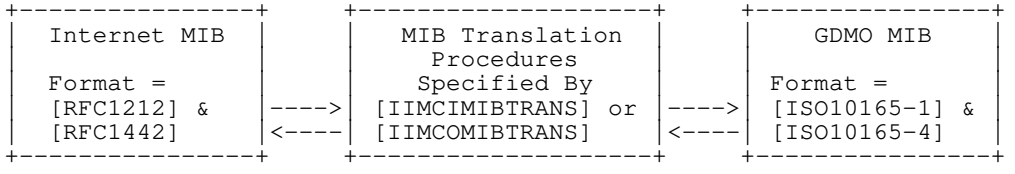
1.3 MIB Translation Procedures

The foundation of IIMC is provided by a pair of Management Information Base (MIB) translation procedures.

- [IIMCIMIBTRANS] specifies translation procedures for converting MIBs from Internet MIB macro format into ISO/CCITT GDMO template format.

- [IIMCOMIBTRANS] specifies translation procedures for converting MIBs from ISO/CCITT GDMO template format into Internet MIB macro format.

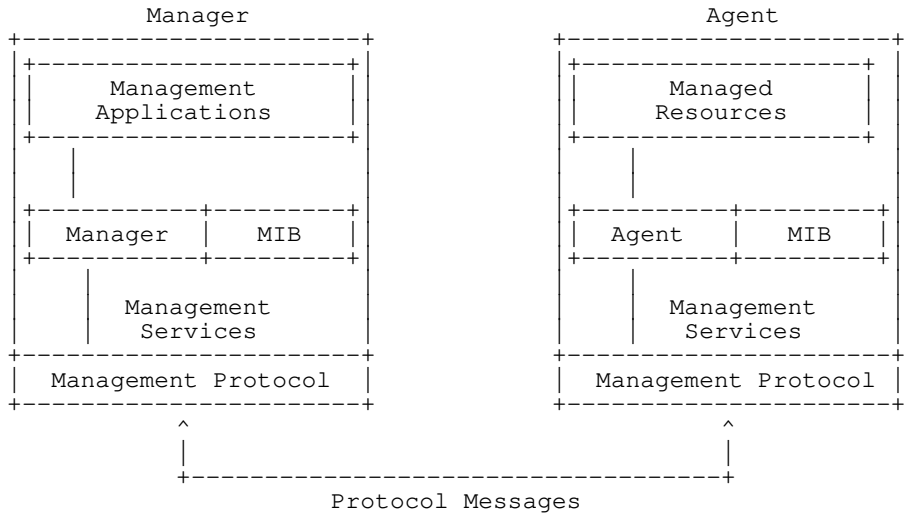
The IIMC approach is to specify direct translation procedures which yield a pair of functionally-equivalent MIBs, as shown in the following figure.



MIBs translated by these procedures 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). Finally, IIMC MIB translation procedures may be used to support service emulation by a proxy.

1.4 Native Management Model

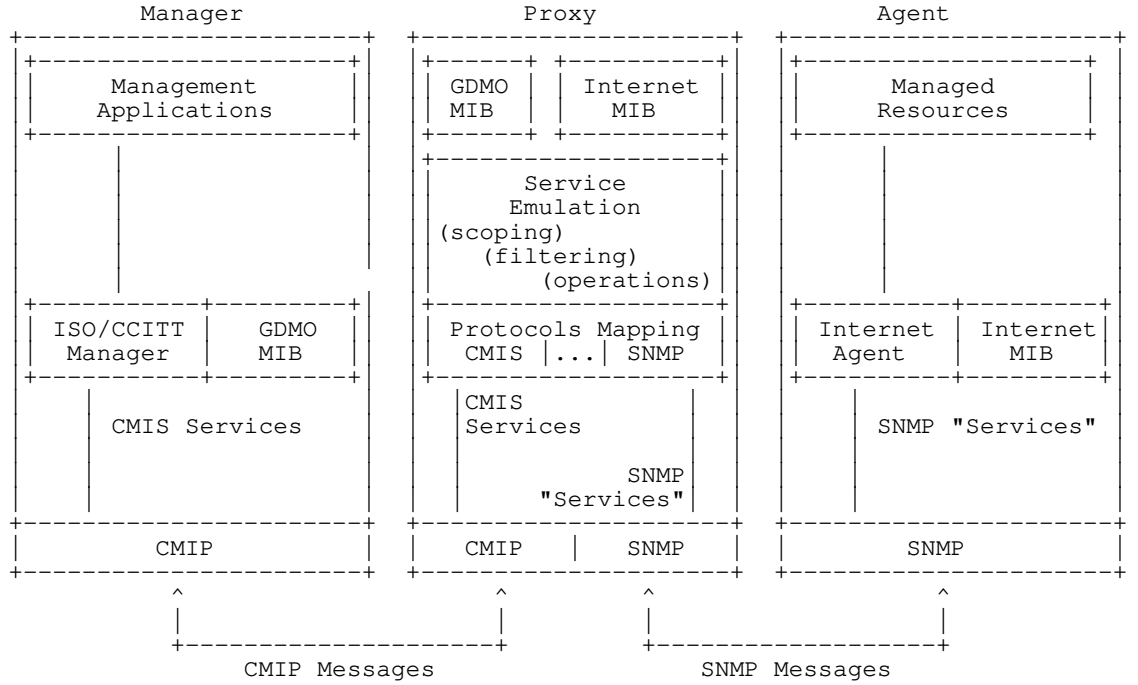
The basic model for ISO/CCITT and Internet management is illustrated in the following diagram.



Within IIMC documents, this model is referred to as the "native" management model. MIBs translated using IIMC procedures can be used by "native" agent implementations. For example, an ISO/CCITT agent can make visible TCP/IP managed resources using the translated GDMO version of the Internet MIB-II [RFC1213] specified by [IIMCMIB-II]. Dual-stack managers or agents may also be implemented which support both the original MIB and the translated MIB generated using IIMC-specified procedures.

1.5 Proxy Management Model

The basic model for ISO/CCITT to Internet proxy management is illustrated in the following diagram. This proxy is specified by [IIMCPROXY]. A similar approach could also be taken to specify an Internet to ISO/CCITT proxy, although no such IIMC document is currently specified.



This ISO/CCITT to Internet proxy 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 uses these MIB definitions and rules to provide run-time translation of management information carried in service requests and responses.

The proxy 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, SNMPv1 and SNMPv2 are variants of the same protocol mapping process).

1.6 Scope of this Document

This document (IIMCOMIBTRANS) specifies an heuristic method to translate from ISO/CCITT GDMO MIB specifications to Internet SNMPv1 and SNMPv2 MIB macro specifications. The method is intended to meet six objectives:

- Lose as little functionality as possible in translation.

- Minimize need for human involvement in translation.

- Minimize cost to implement multi-stack (e.g., CMIP, SNMPv1 and SNMPv2) and proxy-based applications.

- Support an end-to-end view (e.g., Generic Network Model) of distributed information resources that span many computing platforms and network elements.

- Support translation to both SNMPv1 and SNMPv2 while minimizing differences between the two mappings. An important corollary is to generate objects and traps/notifications that are syntactically and semantically equivalent and can therefore be registered the same.

- Within constraints of the other objectives, generate MIBs that are as consistent with customary Internet usage as possible. In particular, these should be compatible with common SNMP parsers.

While 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 may be tempered by human judgment.

This specification provides core methodology such that a GDMO template specification can be translated to Internet MIB macro notation. It thus supports native mode implementations.

Proxy implementations are beyond the scope of this document. However, in the longer term, MIBs translated using this method could be used in conjunction with a proxy architecture to enable interworking between ISO/CCITT managers and Internet agents.

The body of this document has six sections:

- Introduction

- SNMPv1 Translation Procedures
- Constraints on SNMPv1 Protocol Usage
- SNMPv2 Translation Procedures
- Constraints on SNMPv2 Protocol Usage
- Summary

Sample inputs and translated outputs (both normative and informative) are provided in appendices. Appendix C also provides some supporting Internet macro object specifications to assist in MIB translation and implementation.

Two outstanding issues have been identified that are outside the scope of the present version of this document, but may be addressed in future versions:

- definition of a Proxy for translated ISO/CCITT GDMO MIBs, and

- definition of "Pragmas" to document human overrides to algorithmic translation in a systematic and machine readable form. Among other things, this might facilitate generation of proxies.

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

1.7 Terms and Conventions

This document assumes the reader is familiar with the concepts and 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 term "SNMP" is used throughout this document generically to indicate both the SNMPv1 and SNMPv2 frameworks, unless a distinction needs to be made. The terms SNMPv1 and SNMPv2 are used to refer to the respective frameworks specifically.

Formal language constructs from the various MIB notations (ATTRIBUTE, OBJECT-TYPE, SYNTAX, etc.) are used verbatim in this text. Plurals of such keywords ending in upper case are expressed by appending "s" or "es" (e.g., NOTIFICATIONs) while those ending in lower case are suffixed with "'s" (e.g., TruthValue's).

The terms "class" and "attribute" when expressed in lower case are generic, referring either to ISO/CCITT MANAGED OBJECT CLASSes and ATTRIBUTEs (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 ATTRIBUTEs and their values to OBJECT-TYPE macros.

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

Identifiers of Internet MIB macro constructs are referred to as "labels".

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 label.

Internet conventions are observed throughout the text. Customary SNMP types and textual conventions [RFC1443] are produced in the translated output whenever there is a straightforward mapping. Such types and conventions are most developed in SNMPv2. Considering the need to minimize differences between SNMPv1 and SNMPv2 outputs, the SNMPv1 outputs are designed to anticipated SNMPv2 usage. The following table summarizes the SNMPv2 types and conventions mapped to (in alphabetic order of SNMPv2 construct).

ISO Construct	SNMPv2 Types And Conventions
counter ATTRIBUTE	Counter32 & 64 Types
GeneralizedTime & UTCTime	DateAndTime Convention
IA5String, NumericString, PrintableString, VisibleString	DisplayString Convention
GraphicString	GraphicString (see below)
gauge ATTRIBUTE	Gauge32 Type
ObjectInstance or DN	InstancePointer Convention
INTEGER Type	Integer32 or UInteger32
Create/Delete operations and administrative state	RowStatus Convention
BOOLEAN type	TruthValue Convention

All of these are native constructs for SNMPv2, except for GraphicString (which is an SNMPv2 textual convention defined by this specification - refer to Appendix C.2), corresponding to the commonly used OSI type of the same name. Usage of all these constructs is further developed in the sections below.

2. SNMPv1 Translation Procedures

2.1 Relationship to RFC1212

While *translation* per se has not previously been widely investigated, [RFC1212] does provide advice for adopting MIB objects from ISO/CCITT GDMO to Internet MIB macros. 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 MIBs, 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 that will act upon its columnar objects. In particular, it is wisest to avoid defining more columnar objects within a conceptual row, which 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 MIBs with a myriad of optionality.

2.2.2 Additional Constraints

The discussion that follows has six parts.

1. Framework (which introduces the general approach and such notions as class, side and child tables)
2. Registration of Modules and Tables
3. Structure of Class Tables
4. Structure of Side Tables
5. Structure of Child Tables
6. Enumeration of Attribute Values

2.2.2.1 Framework

The framework that follows establishes the general approach to translation. It also introduces associated vocabulary used throughout the remainder of the text.

Every MANAGED OBJECT CLASS input maps to a corresponding object group, the "class group." Each class group consists entirely of Internet MIB tables that are logically linked to each other, namely,

- A "class table" that corresponds to the class as a whole. It contains objects representing scalar ATTRIBUTEs and additional objects that reflect status and containment relationships.

- Zero or more "side tables" to accommodate multiply occurring attribute values. Subsection 2.3 below provides instruction about how side tables are generated.

- A "child table" that lists subordinate managed objects as mapped into SNMP macros.

For each managed object mapped, there is a single entry in the corresponding class table, the "class entry." Associated with this are zero or more "side entries" and "child entries", associated conceptual rows within the class group. A "class instance" is a class entry together with any associated side and child entries.

A class InstancePointer is an OBJECT IDENTIFIER that discriminates (points to) a class instance. It conforms to the SNMPv2 InstancePointer textual convention. Specifically, it points to the first column of the class entry.

The class InstancePointer is used in lieu of the OSI Distinguished Name (DN). It is analogous to the DN in that it names a particular instance of a managed object class, also supplying the type of the object pointed to. However, it is more compact and less mnemonic than the DN.

2.2.2.2 Registration of Modules and Tables

Begin translation of a registry family of MANAGED OBJECT CLASS specifications by allocating the root of a registry subtree to contain the translated objects, the "output subtree."

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

Assign a brief naming prefix to assist in labeling the contents of a corresponding ASN.1 module to be generated.

Define an OBJECT IDENTIFIER of the form

```
<naming prefix>   || " OBJECT IDENTIFIER ::= "
                  || <output subtree>
```

Register the ASN.1 module that is produced as:

```
"{ " || <output subtree> || " 0 1 }"
```

Name the resulting module:

```
"MIB" || <naming prefix> || "SNMPv1",
 e.g., MIBt1taSNMPv1.
```

For each MANAGED OBJECT CLASS of the input registry family, define a corresponding Internet MIB object group — the class group. Assign an arc to each, keeping the same relative arc number in the output registry family as 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 substituting the value 16,384.

Assign arcs to the various tables of a class group.

- The class table registry is the same as that of the class group arc.

- Assign arcs for side tables in ascending order beneath the class group arc beginning with arc number two.

- Assign the child table its own arc underneath the class group, its arc number immediately following that of the highest numbered side table.

    Note: Since all ATTRIBUTEs 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 (naming of objects and entries is addressed below).

2.2.2.3 Structure of Class Tables

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

Reserve arc 0 beneath the class entry arc per the Internet Structure of Management Information (SMI).

Assign arc 1 to the "class entry index," an INTEGER valued object that provides the unique index of a class entry. This may be an arbitrary value without particular meaning. The class InstancePointer is arrived at by concatenating:

- The OBJECT IDENTIFIER of the class entry.

- The value 1 (arc number for the class entry index)

- The value of the class entry index for the instance in question. This index is not-accessible.

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

    Also Note: The class InstancePointer value is used in the translated MIB in lieu of the OSI Distinguished Name's and ObjectInstance's that represent relationships between managed objects. As discussed in section 2.2.3, no direct translation of Distinguished Names is provide.

Assign arcs 2 through <n+1> sequentially to the <n> conceptual columns that correspond to attribute values.

  Note: instructions for mapping of ATTRIBUTE syntax to conceptual columns of the class and side tables are provided in subsection 2.3.

Assign arc <n+2> to the parent object, a class InstancePointer that points to the superior (parent) class entry.

Assign arc <n+3> to the NameBinding object, which holds an OBJECT IDENTIFIER that distinguishes the NAME BINDING in effect for the class instance.

Assign arc <n+4> to the RowStatus object, which conforms to a subset of the SNMPv2 RowStatus textual convention, allowing creation and deletion of class instances.

2.2.2.4 Structure of Side Tables

Side entries have similar structure.

Reserve arc 0 beneath the class entry arc per the Internet Structure of Management Information (SMI).

Assign arc 1 to the "class index," an INTEGER valued object that provides the same value as the corresponding class entry index. This may be an arbitrary value without particular meaning and is not-accessible.

Assign arcs 2 through <m+1> sequentially to the <m> "value indexes" that collectively qualify the class index to uniquely identify a particular entry in the side table. These also are INTEGER valued objects which may have arbitrary values and are not-accessible.

Assign arcs <m+2> through <m+n+1> sequentially to the <n> conceptual columns that correspond to attribute values.

Assign arc <m+n+2> to the RowStatus object, which conforms to a subset of the SNMPv2 RowStatus textual convention, allowing creation and deletion of values for multivalued attribute syntax.

2.2.2.5 Structure of Child Tables

Assign child tables entries the following structure.

Reserve arc 0 beneath the class entry arc per the Internet Structure of Management Information (SMI).

Assign arc 1 to the class index, a not-accessible OBJECT IDENTIFIER valued object that identifies the class of child for which pointers are desired.

Assign arc 2 to the value index, a not-accessible INTEGER valued object that identifies the instance of child within its class

Assign arc 3 to the CHILD OBJECT, a read-only OBJECT IDENTIFIER valued object that provides an InstancePointer to a particular subordinate class entry, i.e., an object contained beneath the class instance in question.

2.2.2.6 Enumeration of Attribute Values

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

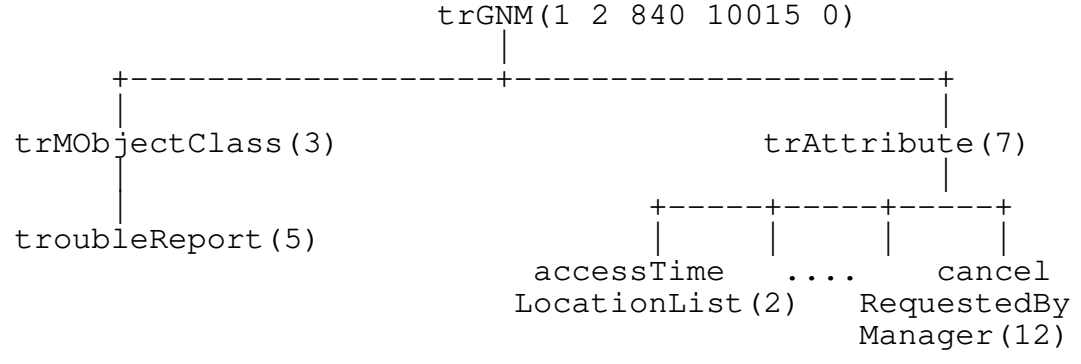


Figure 1. Sample Input Registration Tree

All else being equal, enumerate ATTRIBUTEs based upon a left-to-right scanning of the input ISO/CCITT GDMO specification. ATTRIBUTEs 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 PACKAGEs of two MANAGED OBJECT CLASSes, it will be enumerated twice in the output: once for each class group.

Enumerate ATTRIBUTEs and their values in the same sequence as declared in a PACKAGE. These translate to OBJECT-TYPEs 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 PACKAGEs. Explode ATTRIBUTEs of a derivative class later, enumerating in the order of specification for that class.

In the case of 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.

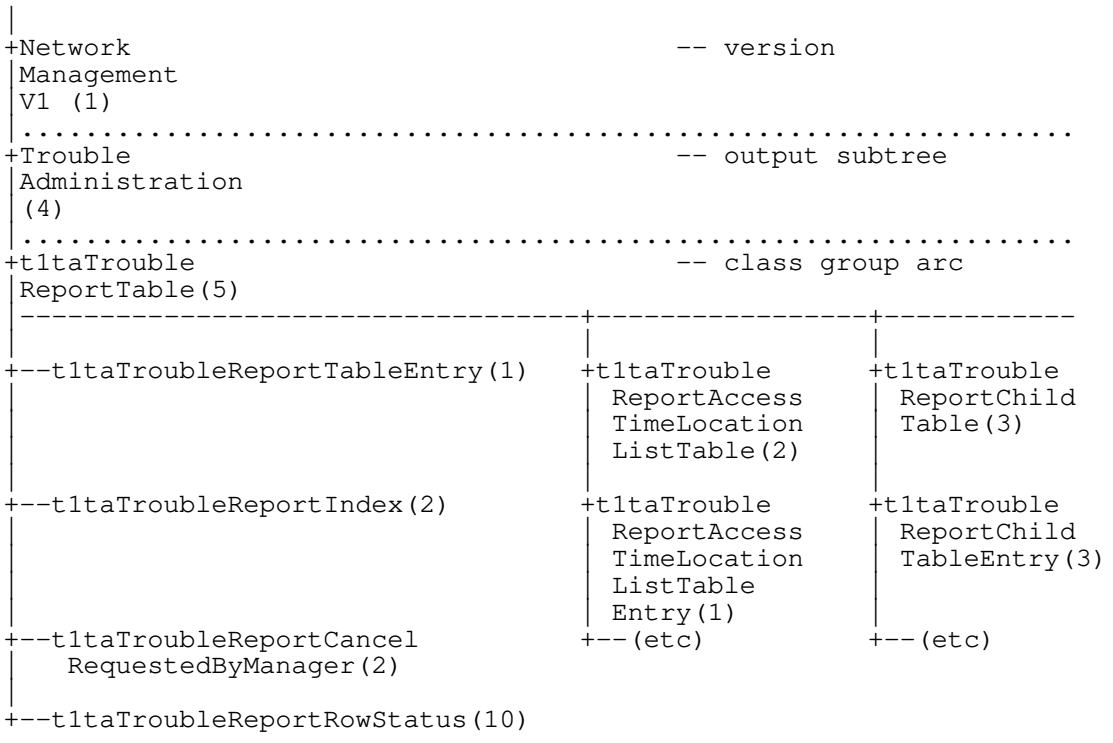


Figure 2. Sample Output Registration Tree

2.2.3 Discussion

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. To simplify translation, all object classes map to tables. This tabular 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 InstancePointer is introduced since direct translation of Distinguished Name's appears impractical, i.e.,

- A possible ambiguity arises since NAME BINDINGs 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 simple sequence of attribute values is not sufficient to uniquely distinguish an object instance.

- NAME BINDINGs 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 an INDEX clause (DNs for different instances of the same object class may be have different numbers of RDNs).

- 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 InstancePointers that identify the table entries corresponding to MANAGED OBJECTs. The indexes of these tables are arbitrary integers that have no significance other than to discriminate between entries of a table. In general, all management information mapped by this specification to OBJECT-TYPEs is mapped to tables that have one or more arbitrary indexes of type INTEGER. Class entries in particular have exactly one index.

For the following reasons, attributes of [ISO10165-2] Top are not directly translated into OBJECT-TYPEs.

- Most of these notions would 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 and 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.

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

The name binding column of the class table provides a mechanism by which name bindings can be provided and inspected. This feature is included for completeness and to avoid ambiguity. As regards order of enumeration, a left-to-right sequential approach is the obvious choice for mechanized translation.

While ISO/CCITT GDMO allows ATTRIBUTEs and ASN.1 syntaxes to be referenced in multiple places and thus shared, Internet MIB format does not. Thus one or more OBJECT-TYPEs must be specified for each template in which they are referenced. The consequent explosion of enumerated ATTRIBUTEs 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) A 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.

BOOLEANs map to the SNMPv2 TruthValue textual convention.

For CHOICE types, provide an OBJECT-TYPE for every alternative and populate exactly one of these alternatives per conceptual row. Document in the DESCRIPTION clause of the generated columns that exactly one of the choices will be present at any one time, also, that setting a value for a new type of choice causes the value for the previously existing option to be automatically removed from the conceptual row.

For SET OF CHOICE types, a conceptual column is likewise allocated for every option. However, multiple such options within a conceptual row may be present at once.

Editor's Note: [Comments are solicited on the correct mapping for CHOICE and SET OF CHOICE.]

For ASN.1 string types that are subsets of ASCII (PrintableString, NumericString, IA5String, VisibleString, etc.) use DisplayString if at all possible. Note that this imposes a length restriction. Map other string constructs (e.g., GraphicString) to OCTET STRING. Implementations should by convention still stick to the NVT ASCII subset where practical.

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.

2.3.2.2 Complex Input Syntax

Expansion of compound constructors sometimes requires definition of side tables, ancillary tables within the object group that supplement the class table corresponding to 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 dependent of the current table. Since this may happen recursively, a side table may be a dependent of another side table.

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

Editor's Note: [Comment on appropriate treatment of ANY is requested. Should we use the deprecated form of SNMP? Should we expand the ANYs for known outcomes? Any other suggestions?]

2.3.3 Discussion

2.3.3.1 Simple Input Syntax

Internet SMI precludes a named INTEGER value of zero and some compilers will not accept 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. Clearly, collisions can occur, in which case human judgment is necessary.

Allowing optional conceptual columns within rows is not consistent with the philosophy of RFC 1212, but neither are the MIBs 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 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 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) it would map to a side table with three indexes of type INTEGER: first, the index of the class 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 entries to side/child tables or vice versa. Given the index of a side table entry, one can find the corresponding class 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.

A common practice with SNMPv1 is to require the INDEX clause of a table to reference only attributes contained within that table. The translation procedure honors that practice by defining distinct OBJECT-TYPEs 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 cannot 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, and

- rules for name mapping, with examples.

2.4.1.1 Notation

The following variables are used to generate labels.

<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."

<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 case where human judgment should be used before carrying over full functionality between MIBs.

Figure 3. Variables for Generating Identifiers

2.4.1.2 Mapping Rules

The following table provides mapping rules for generating Internet MIB labels. 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 entry	<module prefix> || <MANAGED OBJECT CLASSid> | "TableEntry" e.g., t1taTroubleReportTableEntry
class entry index	<module prefix> || <MANAGED OBJECT CLASS id> || "Index" e.g., t1taTroubleReportIndex
class entry column for attribute value	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || <ASN.1 id> e.g., t1taTroubleReportTroubleFoundIdentifier
class entry parent	<module prefix> || <MANAGED OBJECT CLASS id> || "Parent" e.g., t1taTroubleReportParent
class entry nam binding	<module prefix> || <MANAGED OBJECT CLASS id> || "NameBinding" e.g., t1taTroubleReportNameBinding
class entry RowStatus	<module prefix> || <MANAGED OBJECT CLASS id> || "RowStatus" e.g., t1taTroubleReportRowStatus

Identifier	Syntax and Example
side table	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id>|| <ASN.1 id>* || "Table" e.g.,t1taTroubleReportAccessTimeLocationList Table
side entry	<module prefix> ||<MANAGED OBJECT CLASS id> ||<ATTRIBUTE id> || "TableEntry" e.g., t1taTroubleReportAccessTimeLocation ListTable-Entry
side entry clas index	<moduleprefix> || <MANAGED OBJECT CLASS id> | <ATTRIBUTE id> || "ClassIndex" e.g., t1taTroubleReportAccessTimeLocation ListClass-Index
side entry valu index	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || "ValueIndex" || <n> e.g., t1taTroubleReportAccessTimeLocation List-ValueIndex
side entry column for attribute value	<module prefix> || <MANAGED OBJECT CLASS id> || <ATTRIBUTE id> || <ASN.1 id> e.g., t1taTroubleReportAccessTimeLocation List-PremisesName
side entry RowStatus	<module prefix> || <MANAGED OBJECT CLASS id> || "RowStatus" e.g., t1taTroubleReportAccessTimeLocation List-RowStatus

Identifier	Syntax and Example
child table	<module prefix> || <MANAGED OBJECT CLASS id> || "ChildTable"
	e.g.,t1taTroubleReportChildTable
child entry	<module prefix> ||<MANAGED OBJECT CLASS id> | "ChildTableEntry"
	e.g., t1taTroubleReportChildTableEntry
child entry class index	<moduleprefix> || <MANAGED OBJECT CLASS id> "ChildClassIndex"
	e.g., t1taTroubleReportChildClassIndex
child entry value index	<module prefix> || <MANAGED OBJECT CLASS id> || "ChildValueIndex"
	e.g., t1taTroubleReportChildValueIndex
child entry child column	<module prefix> || <MANAGED OBJECT CLASS id> || "Child"
	e.g., t1taTroubleReportChild

Figure 4a-c. 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 label for the syntax of a (class, child 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 additional advantage that names can be used for C and 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 labels for the corresponding OBJECT-TYPEs.

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).

2.7.2 Additional Constraints

Text in any DEFINED AS specification is ordinarily carried over verbatim, although some post-processing may be needed to clarify context. If this information is lacking, the output DESCRIPTION must at least say so (this may also require post-processing).

In cases where generated (translated) SNMPv1 objects comply to SNMPv2 textual conventions, that fact is documented in the

DESCRIPTION clause (typically, following the general description of the object).

In cases where there are additional constraints established by this specification, a statement to that effect referencing this document is also inserted in the DESCRIPTION clause.

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.

This usage applies to ALL clauses that are REFERENCE clauses, e.g., SNMPv1 TRAP-TYPES and SNMPv2 NOTIFICATION-TYPES.

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 indexes 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 to which the object provides access. 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 Appendix B. Since use of traps is strongly discouraged in the Internet community and there is no filtering mechanism in SNMPv1, such translation should be done sparingly.

The SNMPv2 Party MIB does provide for views which might be considered a form of event discrimination. Also, the omibtransSptTrapSuppressionTable defined in Appendix C.1 provides an optional mechanism that can selectively suppress traps of different types.

In any case:

- mapping of notifications to traps should always have a documented justification, and

- 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) Naming and registration are similar to the procedure for translation of MANAGED OBJECTS. For each input registry subtree in which there are NOTIFICATIONs or ATTRIBUTEs to be translated, establish corresponding output registry subtrees to hold the translated MIBs (in the example of Appendix B, the output subtrees are osidmiNotifications and osidmiAttributes respectively).

Generate a separate ASN.1 module 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.

Assign a module naming prefix for translated ATTRIBUTEs and NOTIFICATIONs. This must start with a lower case letter ("osidmi" is the prefix for the translated notifications of Appendix B). Assign the name of the resulting module "MIB" || <module prefix> || "SNMPv1". Assign it the registration "{ " || <module prefix> || " 0 1 }". Define an OBJECT IDENTIFIER

with the label of <module prefix>, its value equaling the value of the output subtree.

(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 Internet 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 label 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 "additionalText" becomes "osidmiAdditionalText" given the prefix "osidmi."

- 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 2.8.

- The value of the DESCRIPTION clause is taken from the BEHAVIOUR DEFINED AS clause, 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) arc number for the output registry is the same as for the input registry.

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

- The label 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 "osidmiObjectCreation" given the prefix "osidmi".

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

- 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 arc number for the output registry (ENTERPRISE clause) is the same as for the input registry.

Editor's Note: [Comment is requested on how protocol length restrictions should be dealt with for mapped notifications.]

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 labels 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.

Editor's Note: [Comment is solicited whether use of the optional VarBind values in the protocol is a better way to deal with this problem, or whether some other approach is appropriate.]

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 Additional Constraints

To simplify mechanized translation, RowStatus is provided for all class tables (and side tables). This is simpler than trying to determine which classes support manager-initiated create and delete operations.

Deletion is accomplished by setting RowStatus to "destroy". Successful deletion of a class instance automatically leads to deletion of associated side and child table rows. The operation succeeds or fails as a whole (e.g., if there are dependents and automatic deletion is not supported, then nothing is deleted). Instances of side table entries can be deleted if and only if that is allowed for the associated set valued ATTRIBUTEs.

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 conforming 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

2.14.1 Overview

The row creation (and deletion) rules for conceptual rows created according to this procedure conform to a subset (refinement) of the SNMPv2 Row Status textual convention. That subset is required to enforce consistency with the containment relationships, services and valid object states of OSI management.

The RowStatus convention defines six possible status values (states) for a conceptual row: active, notInService, notReady, createAndGo, createAndWait and destroy. Three of these are visible to get operations: active and notInService, which map to existing OSI managed objects, and notReady, which is a preparatory state during creation of a managed object. Statuses of createAndGo and createAndWait are transitory values used by set operations, discussed further below.

The fundamental principle for this subset of RowStatus usage is that all values of an instance of a class entry remain in a state consistent with OSI usage so long as Row Status is active or notInService. This consistency constraint applies subsequent to *every* set operation, although the operation may partially succeed (best efforts). If a set operation is attempted that in any way violates OSI usage (NAME BINDINGs, containment, conditional packages, etc.) error-status other than noError is returned (see below for further discussion of error-status usage).

While OSI structural notions are translated to Internet MIB macro format where there is an analog, in many cases there is no equivalent, e.g., NAME BINDINGs and PACKAGEs. No attempt is made to define alternative notation in such cases. The source

GDMO specification is the ultimate authority determining valid state of translated objects, their attributes and relationships.

For constructs that do not carry over, no attempt is made to generate corresponding text in DESCRIPTION clauses of the generated tables and traps/notifications. However, DESCRIPTION clause use for RowStatus objects generated in translation cite this specification explicitly.

2.14.1.1 Status of notInService

Status of notInService is only supported if the corresponding OSI resource is governed by an Administrative state ATTRIBUTE that supports a value of "locked". The notInService status is functionally equivalent to locked, while active status equates to unlocked (or existing without Administrative state). An instance of a class entry can have its administrative state locked or unlocked either by setting the administrative state column directly or by switching RowStatus between active and notInService.

The transitory Administrative state of "shutting down" is reflected in a RowStatus of active until it completes, at which point RowStatus becomes notInService. In this case, a response to the set operation is not returned until the resource is shut down.

2.14.1.2 Status of notReady

A conceptual row with row status of notReady does not (yet) exist with respect to consistency with OSI usage. It is impossible for an inconsistentValue error-status to arise when manipulating a row with this status, since there is no enforcement of integrity constraints.

However, once a row achieves active or notInService status, it cannot return to notReady unless first destroyed and recreated.

2.14.1.3 Row Creation

A basic requirement for creating an instance of a class entry is that the conceptual row's parent column points to an existing instance of a superior class entry for which there is a valid NAME BINDING. In the event there is ambiguity about which NAME BINDING to use, the OBJECT IDENTIFIER of the NAME BINDING to be used must be present in the NAME BINDING column of the class entry being created. If the row is successfully created, an empty child table is automatically generated for the new class InstancePointer. The child table of the new

instance's superior is also automatically updated to reflect the new subordinate.

A class InstancePointer must be created (first set to active or notInService) in a single operation resulting in a state consistent with OSI usage. This is accomplished using a single set operation without setup, or by staging information in notReady status and then creating the instance. Creation is ordinarily accomplished via set of the class InstancePointer RowStatus to createAndGo, resulting in active status. For a class having administrative state, this can alternatively be done by setting RowStatus to createAndWait, resulting in a new instance in locked state.

In either case, values for conditional packages that need to be present must be populated when the instance is activated. Side table instances cannot be created before corresponding class InstancePointers are created, although they may be staged in notReady status.

The omibtransSptNextUniqueIndexTable (see Appendix C) provides a convenient mechanism for agent to supply manager with an index to create a new conceptual row. The index returned is unique to the table in question and allows the agent (optionally) to implement index assignment policies (e.g., FIFO or small number) on a per table basis. Implementation of that table is discretionary. Any valid option for RowStatus may be used.

2.14.2 Discussion

To support creation of class instances complicates matters considerably, since the manager and its users must understand the underlying OSI constructs. This may not always be appropriate for native SNMP implementations, in which case a subset can be implemented that does not support create/delete.

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 potentially be a problem, in which case human judgment is needed.

3. Constraints on SNMPv1 Protocol Usage

3.1 Approach

The following assumptions about use of SNMPv1 protocol 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 that stated in the ACCESS clause due to dynamic access controls.

- As discussed above, set operations on RowStaus and associated objects are constrained to leave the object in a state consistent with OSI usage. Among other things, that means that multiple columns having consistency dependencies must jointly be updated in a single operation when the class InstancePointer has active or outOfService status.

Mapping of OSI functionality also requires refinements to use of error-status as documented as follows.

noError	No constraints.
tooBig	No constraints.
noSuchName	Customary usage, but is also returned when an attempt is made to access an object for which read access is denied due to access control. I this case, the object is "invisible."
badValue	Customary usage, but may be returned due to a set request that is contrary to the translated object's BEHAVIOR or PERMITTED VALUES specifications, or would violate integrity of the class InstancePointer overall.
readOnly	Customary usage, but may be returned due to access control as well as violated ACCESS clause.
genErr	No constraints.

3.2 Discussion

A "bending" of the rules of SNMPv1 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. SNMPv2 Translation Procedures

4.1 General Approach

Consistent with the section 2 which defines translation in terms of constraints on existing Internet usage, this section builds upon the subsections 2.1 and 2.2 of [RFC1452] that respectively describe conversion from SNMPv1 OBJECT-TYPEs and TRAP-TYPEs to the SNMPv2 framework.

However, this format is an editorial convenience. The purpose of this section is to describe the *differences* between translated SNMPv1 and SNMPv2 in a succinct and familiar form. Note however that in practice a translator is unlikely to translate from GDMO to SNMPv1 then translate that output to SNMPv2. Translation to either SNMP framework requires deep knowledge of the original GDMO MIB; translation is best conceived of as a single input, dual output process.

The philosophy guiding the procedures in this section is as follows.

(1) Build upon the work of [RFC1452].

(2) Bring SNMPv1 mappings previously defined for IIMCOMIBTRANS in line with anticipated SNMPv2 usage (e.g., RowStatus). This includes mapping into common conventions as described in section 1 above. Relevant textual conventions are cited directly in their associated SYNTAX clauses rather than as comments in DESCRIPTION clauses.

(3) Make all other adjustments necessary to compile with SNMPv2 syntax.

(4) Make optional adjustments sparingly, only when there is a natural mapping and where mechanized translation is not difficult to implement.

4.2 Object Definitions

4.2.1 RFC 1452 Advice

In general, conversion of a MIB module does not require the deprecation of the objects contained therein. Only if the semantics of an object truly changes should deprecation be performed.

(1) The IMPORTS statement must reference SNMPv2-SMI, instead of RFC1155-SMI and RFC-1212.

(2) The MODULE-IDENTITY macro must be invoked immediately after any IMPORTs or EXPORTs statement.

(3) For any descriptor which contains the hyphen character, the hyphen character is removed.

(4) For any object with an integer-valued SYNTAX clause, in which the corresponding INTEGER does not have a range restriction (i.e., the INTEGER has neither a defined set of named-number enumerations nor an assignment of lower- and upper-bounds on its value), the object must have the value of its SYNTAX clause changed to Integer32.

(5) For any object with a SYNTAX clause value of an enumerated INTEGER, the hyphen character is removed from any named-number labels which contain the hyphen character.

(6) For any object with a SYNTAX clause value of Counter, the object must have the value of its SYNTAX clause changed to Counter32.

(7) For any object with a SYNTAX clause value of Gauge, the object must have the value of its SYNTAX clause changed to Gauge32.

(8) For all objects, the ACCESS clause must be replaced by a MAX-ACCESS clause. The value of the MAX-ACCESS clause is the same as that of the ACCESS clause unless some other value makes "protocol sense" as the maximal level of access for the object. In particular, object types for which instances can be explicitly created by a protocol set operation, will have a MAX-ACCESS clause of "read-create". If the value of the ACCESS clause is "write-only", then the value of the MAX-ACCESS clause is "read-write", and the DESCRIPTION clause notes that reading this object will result implementation-specific results.

(9) For any columnar object which is used solely for instance identification in a conceptual row, the object must have the value of its MAX-ACCESS clause set to "not

accessible", unless all columnar objects of the conceptual row are used for instance identification, in which case, the MAX-ACCESS clause for one of them must be something other than "not-accessible".

(10) For all objects, if the value of the STATUS clause is "mandatory", the value must be replaced with "current".

(11) For all objects, if the value of the STATUS clause is "optional", the value must be replaced with "obsolete".

(12) For any object not containing a DESCRIPTION clause, the object must have a DESCRIPTION clause defined.

(13) For any object corresponding to a conceptual row which does not have an INDEX CLAUSE, The object must have either an INDEX clause or an AUGMENTS clause defined.

(14) For any object with an INDEX clause that references an object with a syntax of NetworkAddress, the value of the STATUS clause of the both objects is changed to "obsolete".

(15) For any object containing a DEFVAL clause with an OBJECT IDENTIFIER value which is expressed as a collection of sub-identifiers, change the value to reference a single ASN.1 identifier.

Other changes are desirable, but not necessary.

(1) Creation and deletion of conceptual rows is inconsistent using the current framework. The SNMPv2 framework corrects this. As such, if the MIB module undergoes review early in its lifetime, and it contains conceptual tables which allow creation and deletion of conceptual rows, then it may be worthwhile to deprecate the objects relating to those tables and replacing them with objects defined using the new approach.

(2) For any object with a string-valued SYNTAX clause, in which the corresponding OCTET STRING does not have a size restriction (i.e., the OCTET STRING has no assignment of lower- and upper-bounds on its length), one might consider defining the bounds for the size of the object.

(3) For all textual conventions informally defined in the MIB module, one might consider redefining those conventions using the TEXTUAL-CONVENTION macro. Such a change would not necessitate deprecating objects previously defined using an informal textual convention.

(4) For any object which represents a measurement in some kind of units, one might consider adding a UNITS clause to the definition of t

(5) For any conceptual row which is an extension of another conceptual row, i.e., for which subordinate columnar objects both exist and are identified via the same semantics as the other conceptual row, one might consider using an AUGMENTS clause in place of the INDEX clause for the object corresponding to the conceptual row which is an extension.

Finally, when encountering common errors in SNMPv1 MIB modules:

(1) For any object with a SYNTAX clause value of an enumerated INTEGER, if a named-number enumeration is present with a value of zero, the value of the STATUS clause of that object is changed to "obsolete".

(2) For any non-columnar object that is instanced as if it were immediately subordinate to a conceptual row, the value of the STATUS clause of that object is changed to "obsolete".

(3) For any conceptual row object that is not contained immediately subordinate to a conceptual table, the value of the STATUS clause of that object (and all subordinate objects) is changed to "obsolete".

4.2.2 Additional Constraints

Translated MIBs must also reference SNMPv2-TC and SNMPv2-CONF.

While translation procedures are designed so objects and traps/notifications need not be deprecated, two different ASN.1 output modules are needed to support the same input MIB, given the incompatible MIB specification macros of the two SNMP frameworks. Generate the following names for the SNMPv2 module as follows.

- The identifier of the generated ASN.1 module is: "MIB" || <module prefix> || "SNMPv2".

- The ASN.1 registry of the generated module is { <output subtree> 0 2 }.

- The identifier of the MODULE-IDENTITY macro is <module prefix> || "SNMPv2".

Most values for the MODULE-IDENTITY clauses must be provided manually. However, a mechanized translator should generate

skeletal text sufficient to compile. The value of the LAST-UPDATED clause, the macro's identifier and OBJECT-IDENTIFIER can be automatically generated. The value of that macro production, which equals the registry of the generated SNMPv2 ASN.1 module, i.e., { <module prefix> 0 2 }.

Map OSI counters and gauges to SNMP equivalents, using the smallest type that can accommodate the range of values. Generate an INTEGER type if there are named values (map enumerates into named values) or if subtyping information is available, carrying over the values / subtypes. Otherwise, use Integer32.

Textual conventions are observed as identified in Section 1.

Generated objects showing read-write ACCESS for SNMPv1 reflect read-create MAX-ACCESS for SNMPv2.

4.2.3 Comments

While at any point there may not be a known NAME BINDING that would allow a create operation, it is always possible that a future binding will so allow. Any translated object that can be written can potentially be created.

Some SNMPv1 compilers take exception if INDEXes are not conceptual columns within the conceptual rows they index. This is accommodated in the SNMPv1 mappings and carried over to SNMPv2.

Information about UNITS can optionally be picked up in a manual post-processing phase, by examining BEHAVIOR clauses of mapped ATTRIBUTES and PACKAGEs

4.3 Trap Definitions

If a MIB module is changed to conform to the SNMPv2 framework, then each occurrence of the TRAP-TYPE macro must be changed to a corresponding invocation of the NOTIFICATION-TYPE macro.

(1) The IMPORTS statement must not reference RFC-1215.

(2) The ENTERPRISES clause must be removed.

(3) The VARIABLES clause must be renamed to the OBJECTS clause.

(4) The STATUS clause must be added.

Additional Constraints: STATUS is always current.

(5) The value of an invocation of the NOTIFICATION-TYPE macro is an OBJECT IDENTIFIER, not an INTEGER, and must be changed accordingly.

Additional Constraints: This value must be equal to the value used for the ENTERPRISE clause of the SNMPv1 mapping, succeeded by the value of the TRAP-TYPE macro of the SNMPv1 mapping, with the combination of the two enclosed in curly brackets ("{}"). For example, a TRAP-TYPE value of 1 with ENTERPRISE value onSMI2toInternetNotification in the SNMPv1 framework will map to NOTIFICATION-TYPE value of {onSMI2toInternetNotification 1 }.

5. Constraints on SNMPv2 Protocol Usage

5.1 Approach

As with SNMPv1, SNMPv2 set operations on RowStatus and associated objects are constrained to leave the object in a state consistent with OSI usage. Among other things, that means that multiple columns having consistency dependencies must jointly be updated in a single operation when the class InstancePointer has status of active or outOfService.

The following table reflects error-status usage for SNMPv2.

noError	No constraints, except that certain conditions are interpreted as errors that would not otherwise be.
tooBig	No constraints.
noSuchName	Does not arise.
badValue	Does not arise.
readOnly	Does not arise.
genErr	No constraints.
noAccess	No constraints.
wrongType	No constraints.
wrongLength	No constraints.
wrongEncoding	No constraints.
wrongValue	No constraints.
noCreation	Will be returned if there is no NAME BINDING permitting row creation.
inconsistentValue	Consistency is enforced across a class InstancePointer, and may apply to the presence or absence of row entries, as well as their values.
resourceUnavailable	No constraints.
commitFailed	No constraints. Note that the class InstancePointer is still left in consistent state.
undoFailed	No constraints. Note that the class InstancePointer is still left in consistent state.
authorizationError	No constraints.
notWritable	No constraints.
inconsistentName	No constraints.

Note that attributes and objects for which read access is denied due to access control remain invisible, returning noSuchObject in the VarBind.

5.2 Discussion

The added capabilities of the SNMPv2 protocol allow most constraints required for translation to SNMPv1 to be relaxed. Existence of the authorizationError value avoids the contrived use of readOnly to signal access control violation. The codes for authorizationError and inconsistentValue provide straightforward solutions to the major problems encountered with SNMPv1 usage.

6. Summary

Given certain assumptions about the use of SNMP (sections 3 and 5), this procedure allows mechanized translation of most functionality found in ISO/CCITT MIBs to the world of SNMP. These procedures preserve the 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 may be lost during translation include the following.

- Notifications: A minimalistic set of capabilities are provided for basic notifications, but more study is needed.

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

- Scoping and Filtering: Rudimentary tools are provided for navigating translated MIBs using SNMP.

7. Acknowledgments

The following individuals have contributed to this effort.

Bob Aronoff	- NIST
Jon Biggar	- NetLabs
Mary Brady	- NIST
April Chang	- NetLabs
Ken Chapman	- Stratus Computer Inc.
Alice Chen	- Boeing
Christopher Crowell	- Cabletron Systems
Jock Embry	- Opening Technologies
Ian Emsley	- Bull S.A
Paul Golick	- IBM
Ulrich Gremmelmaier	- University of Stuttgart
Pramod Kalyanasundaram	- University of Delaware
Ken Hunter	- Hewlett-Packard
Lee LaBarre	- The MITRE Corporation
David Liu	- Northern Telecom
Jim MacLeod	- U S West
Reece Markowsky	- OSIWare
Subrata Mazumdar	- IBM
Keith McCloghrie	- Hughes LAN Systems
Owen Newnan	- U S West
Steve Ng	- MPR Teltech
Yasuhiro Ohara	- NTT
Jong-Tae Park	- KyungPook National University
George Pavlou	- University College of London
Lisa Phifer	- Bellcore
Jim Reilly	- Technical Rsch Ctr of Finland
Tom Rutt	- AT&T

Adarsh Sethi	- University of Delaware
Raj Sirsikar	- University of Delaware
Baltej Singh	- OSIWare
Mark Smith	- Hewlett-Packard
Einar Stefferud	- Network Management Associates
Mark Sylor	- Digital
Hector Trevino	- Bellcore
Huy Truong	- Tandem
Al Vincent	- U S West
Dean Voiss	- NetLabs
David Waitzman	- BBN
Graham Wisdom	- Timeplex
Yoshi Yamashita	- NKK Corporation

References

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

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

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

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

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

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

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

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

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

[RFC1157] RFC1157, 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] RFC1213, K. McCloghrie and M. Rose - Editors, Management Information Base for Network Management of TCP/IP-basedinternets: MIB-II, March 1991.

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

[RFC1441] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Introduction to version 2 of the Internet-standard Network Management Framework, April 1993.

[RFC1442] 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), April 1993.

[RFC1443] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2), April 1993.

[RFC1452] J.D. Case, K. McCloghrie, M.T. Rose, S.L.Waldbusser, Coexistence between version 1 and version 2 of the Internet Network Management Framework, April 1993.

[IIMCIMIBTRANS] ISO/CCITT and Internet Management Coexistence (IIMC): Translation of Internet MIBs to ISO/CCITT GDMO MIBs, Draft 2, May 1993.

[IIMCMIB-II] ISO/CCITT and Internet Management Coexistence (IIMC): Translation of Internet MIB-II (RFC1213) to ISO/CCITT GDMO MIB, Draft 2, May 1993.

[IIMCPROXY] ISO/CCITT and Internet Management Coexistence (IIMC): ISO/CCITT to Internet Management Proxy, Draft 2, May 1993.

[IIMCSEC] ISO/CCITT and Internet Management Coexistence (IIMC): ISO/CCITT to Internet Management Security, Draft 2, May 1993.

[NMFTR107] NM Forum and X/Open, ISO/CCITT and Internet Management: Coexistence and Interworking Strategy, Issue 1.0, 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 (Informative): 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: [Appendix B will be expanded in the next draft of this document, and will then serve as a more complete example. This Appendix may then be removed.]

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 SNMPv1 Management Information Base

```
MIBt1taSNMPv1 DEFINITIONS ::= BEGIN
     IMPORTS
          OBJECT-TYPE, internet
     FROM RFC1155-SMI;

-- IMPORTS may of course include a minimalistic subset as an
-- optimization (as illustrated above), however,
-- boilerplate for generated code may contain IMPORTS not
-- used.

-- The example which follows is similar to what might be
-- generated using a mechanized translator except that
-- it is liberally interspersed with comments.

-- Generated code must clearly indicated ASN.1 registry of
-- the module, in the DEFINITIONS production and/or in
-- comments.  Note, some SNMPv1 parsers cannot handle
-- registry of modules, so a translator should provide an
-- option whereby this information is included as comments.
-- Such information is not provided here, since this is
-- a fabricated example.  OIDs for trAttribute and
```

```
-- trMobjectClass would also be provided for a real MIB.

-- If this were a normative example, a production of the form
--   t1ta OBJECT IDENTIFIER ::= { ... } would be present.
-- The registry of this ASN.1 module would then be
--   { t1ta 0 1 }, the SNMPv1 version of
--   Trouble Administration.

t1ta OBJECT IDENTIFIER ::= { ? }

-- ******************************************************

-- This example illustrates a single translated object
-- group class table, a side table (corresponding to a
-- multi-valued attribute) and a child table.

-- This example illustrates among other things
-- translation of enumerateds, timestamps, BOOLEANs,
-- CHOICEs, ObjectInstance's and default values to SNMPv1.

-- ******************************************************

-- The general structure of this module (indentation
-- reflects the OID structure) is:

-- t1taTroubleReportTable
  -- t1taTroubleReportTableEntry
     -- t1taTroubleReportIndex
     -- t1taTroubleReportCancelRequestedByManager
     -- t1taTroubleReportManagedObjectInstance
     -- t1taTroubleReportReceivedTime
     -- t1taTroubleReportTroubleFoundNumber
     -- t1taTroubleReportTroubleFoundIdentifier
     -- t1taTroubleReportPerceivedTroubleSeverity
     -- t1taTroubleReportRowParent
     -- t1taTroubleReportRowNameBinding
     -- t1taTroubleReportRowStatus

  -- t1taTroubleReportAccessTimeLocationListTable
    -- t1taTroubleReportAccessTimeLocationListTableEntry
       -- t1taTroubleReportAccessTimeLocationListClassIndex
       -- t1taTroubleReportAccessTimeLocationListValueIndex
       -- t1taTroubleReportAccessTimeLocationListPremisesName
       -- t1taTroubleReportAccessTimeLocationList-
       --      PremisesAddress
       -- t1taTroubleReportAccessTimeLocationListRowStatus

  -- t1taTroubleReportChildTable
    -- t1taTroubleReportChildTableEntry
       -- t1taTroubleReportChildClassIndex
       -- t1taTroubleReportChildValueIndex
       -- t1taTroubleReportChild

-- ******************************************************
```

```
t1taTroubleReportTable    OBJECT-TYPE
--   class table
     SYNTAX SEQUENCE OF T1TATroubleReportTableEntry
     ACCESS  not-accessible
     STATUS  mandatory

     DESCRIPTION
     "t1taTroubleReport class table."

     REFERENCE "trMobjectClass 5"
::= { t1ta 5 }
--   5th object in TA Input and Output Subtrees

t1taTroubleReportTableEntry OBJECT-TYPE
--   class table entry
     SYNTAX   T1TATroubleReportTableEntry
     ACCESS  not-accessible
     STATUS  mandatory

     DESCRIPTION
     "t1taTroubleReportTable instance"

     INDEX  { t1taTroubleReportIndex }
::= { t1taTroubleReportTable 1 }

T1TATroubleReportTableEntry   ::= SEQUENCE {
     t1taTroubleReportIndex
          INTEGER,
     t1taTroubleReportCancelRequestedByManager
          INTEGER,
     t1taTroubleReportManagedObjectInstance
          OBJECT IDENTIFIER,
     t1taTroubleReportReceivedTime
          DisplayString, -- DateAndTime
     t1taTroubleReportTroubleFoundNumber
          INTEGER,
     t1taTroubleReportTroubleFoundIdentifier
          OBJECT IDENTIFIER,
     t1taTroubleReportPerceivedTroubleSeverity
          INTEGER,
     t1taTroubleReportParent
          OBJECT IDENTIFIER,
     t1taTroubleReportNameBinding
          OBJECT IDENTIFIER,
     t1taTroubleReportRowStatus
          INTEGER
          }

t1taTroubleReportIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS  not-accessible
     STATUS  mandatory
```

```
     DESCRIPTION
     "Provides the index by which instances of
     t1taTroubleReport table entries are distinguished."

::= { t1taTroubleReportTableEntry 1 }

t1taTroubleReportCancelRequestedByManager OBJECT-TYPE
     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.

     This object complies with the TruthValue textual
     convention of the SNMPv2 framework per RFC 1443.  It is
     functionally equivalent to a BOOLEAN type but takes
     different values."

--   Note that text citing the SNMPv2 convention is
--   concatenated to text from the input BEHAVIOUR
--   specification.

     REFERENCE "trAttribute 12"
     DEFVAL { 1 }   --   TruthValue usage for FALSE
--        troubleReportCancelRequestedByManagerDefault
--             BOOLEAN ::=  FALSE
::= { t1taTroubleReportTableEntry 2 }

t1taTroubleReportManagedObjectInstance OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS  read-write
     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.

     This object complies with the RFC 1443 InstancePointer
     textual convention of the SNMPv2 framework, which
     defines an OBJECT IDENTIFIER pointer to a conceptual
     row."


     REFERENCE "trAttribute 29"
::= { t1taTroubleReportTableEntry 3 }

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.

     This object complies with the DateAndTime textual
     convention of the SNMPv2 framework per RFC 1443.
     DateAndTime reports local time in a standard format."

     REFERENCE "trAttribute 33"
::= { t1taTroubleReportTableEntry 4 }

-- Note that t1taTroubleReportAccessTimeLocationList is not
-- assigned arc 5 because it is implemented as a side table
-- due to its set valued syntax; see below.


-- The following illustrates mapping of a CHOICE.
-- Exactly one of the following two choices will be present
-- for a given class instance.
-- 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 {
--   Note that INTEGERs of enumerated value can be mapped
--   as comments or formal syntax.
               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 5}

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 6}

t1taTroubleReportPerceivedTroubleSeverity OBJECT-TYPE
     SYNTAX INTEGER {
          outOfService(32767),  -- mapped from zero
          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 7}

t1taTroubleReportParent OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS  read-write
     STATUS  mandatory

     DESCRIPTION
     "This object is used to specify the superior class
     entry instance when creating a subordinate class entry
     instance. Likewise, it can be queried to determine the
     superior class instance (if any) for an
     existing class instance.
```

```
     This object complies with the RFC 1443 InstancePointer
     textual convention of the SNMPv2 framework, which
     defines an OBJECT IDENTIFIER pointer to a conceptual
     row.

     Behavior of this object furthermore conforms to the
     specifications of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= { t1taTroubleReportTableEntry 8 }

t1taTroubleReportNameBinding OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS  read-write
     STATUS  mandatory

     DESCRIPTION
     "This object allows specification of a
     specific NAME BINDING for creation of a class entry
     instance.  It can also be examined to
     determine the NAME BINDING in effect for an existing
     class table instance.

     Use of this object must conform to the specifications
     of IIMCOMIBTRANS ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 which
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= { t1taTroubleReportTableEntry 9 }

t1taTroubleReportRowStatus OBJECT-TYPE
     SYNTAX  INTEGER -- conforms to SNMPv2 RowStatus usage
     ACCESS  read-write
     STATUS  mandatory

     DESCRIPTION
     "This object complies with the RowStatus textual
     convention of the SNMPv2 framework per RFC 1443.
     The RowStatus convention specifies elements of procedure
     for creation, deletion and state management of
     conceptual rows.

     Behavior of this object furthermore conforms to
     subsections 2.13 and 2.14 of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 }), that
     enforce consistency with OSI management
     containment relationships, abstract services and
```

```
     permissible object states.

     These subsections specify constraints (refinements) to
     RowStatus usage beyond what is specified by RFC 1443."

::= { t1taTroubleReportTableEntry 10 }

-- ******************************************************

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

t1taTroubleReportAccessTimeLocationListTable OBJECT-TYPE
     SYNTAX  SEQUENCE OF
         T1TATroubleReportAccessTimeLocationListTableEntry
     ACCESS  not-accessible
     STATUS  mandatory

--   This attribute is in a conditional package, so it
--   may be absent from a given conceptual row.

     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"
::= { t1taTroubleReportTable 2 }

t1taTroubleReportAccessTimeLocationListTableEntry OBJECT-TYPE
     SYNTAX T1TATroubleReportAccessTimeLocationListTableEntry
     ACCESS  not-accessible
     STATUS  mandatory
     DESCRIPTION

     "t1taTroubleReportAccessTimeLocationListTable entry."
     INDEX  {
          t1taTroubleReportAccessTimeLocationListClassIndex,
          t1taTroubleReportAccessTimeLocationListValueIndex
          }
     ::= { t1taTroubleReportAccessTimeLocationListTable 1 }

T1TATroubleReportAccessTimeLocationListTableEntry
     ::=  SEQUENCE {
     t1taTroubleReportAccessTimeLocationListClassIndex
               INTEGER,
     t1taTroubleReportAccessTimeLocationListValueIndex
               INTEGER,
     t1taTroubleReportAccessTimeLocationListPremisesName
               DisplayString,
     t1taTroubleReportAccessTimeLocationListPremisesAddress
               DisplayString,
     t1taTroubleReportAccessTimeLocationListRowStatus
               INTEGER
```

```
     }

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 1 }

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 2 }

t1taTroubleReportAccessTimeLocationListPremisesName
     OBJECT-TYPE
     SYNTAX  DisplayString
     ACCESS  read-write
     STATUS  mandatory
     DESCRIPTION
     "The access time for a service location list premises
     name."
::= { t1taTroubleReportAccessTimeLocationListTableEntry 30 }

t1taTroubleReportAccessTimeLocationListPremisesAddress
     OBJECT-TYPE
     SYNTAX  DisplayString
     ACCESS  read-write
     STATUS  mandatory

     DESCRIPTION
     "The access time for a service location list premises
     address."

::= { t1taTroubleReportAccessTimeLocationListTableEntry 4 }

t1taTroubleReportAccessTimeLocationListRowStatus OBJECT-TYPE
     SYNTAX  INTEGER
     ACCESS  read-write
     STATUS  mandatory

     DESCRIPTION
     "This object complies with the RowStatus textual
     convention of the SNMPv2 framework per RFC 1443.
```

```
     The RowStatus convention specifies elements of procedure
     for creation, deletion and state management of
     conceptual rows.

     Behavior of this object furthermore conforms to
     subsections 2.13 and 2.14 of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 }), that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states.

     These subsections specify constraints (refinements) to
     RowStatus usage beyond what is specified by RFC 1443."

::= { t1taTroubleReportAccessTimeLocationListTableEntry 5 }


--  ***************************************************

t1taTroubleReportChildTable OBJECT-TYPE
     SYNTAX SEQUENCE OF T1TATroubleReportChildTableEntry
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Allows children in the ISO/CCITT Management naming
     hierarchy to be returned."

::= { t1taTroubleReportTable 3 }

t1taTroubleReportChildTableEntry OBJECT-TYPE
     SYNTAX T1TATroubleReportChildTableEntry
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
          "Entry of Child table."

     INDEX  {
          t1taTroubleReportChildClassIndex,
          t1taTroubleReportChildValueIndex
     }
::= { t1taTroubleReportChildTable 1 }

T1TATroubleReportChildTableEntry ::= SEQUENCE {
     t1taTroubleReportChildClassIndex INTEGER,
     t1taTroubleReportChildValueIndex INTEGER,
     t1taTroubleReportChild OBJECT IDENTIFIER
     -- RowStatus is not applicable
     }

t1taTroubleReportChildClassIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS not-accessible
```

```
     STATUS mandatory

     DESCRIPTION
     "Implicit index value of class table."

::= {t1taTroubleReportChildTableEntry 1}

t1taTroubleReportChildValueIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Implicit index value of child table."

::= {t1taTroubleReportChildTableEntry 2}


t1taTroubleReportChild OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-only
     STATUS mandatory

     DESCRIPTION
     "This parameter is typically used in conjunction with a
     get next operation to determine class instances
     for successive child (contained) objects within the
     parent (superior object).

     This object complies with the RFC 1443 InstancePointer
     textual convention of the SNMPv2 framework, which
     defines an OBJECT IDENTIFIER pointer to a conceptual
     row.

     Behavior of this object furthermore conforms to the
     specifications of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= {t1taTroubleReportChildTableEntry 3 }

END
```

A.3 Output SNMPv2 Management Information Base

```
MIBt1taSNMPv2 DEFINITIONS ::= BEGIN

IMPORTS
     MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
     Integer32, internet, snmpModules
```

```
          FROM SNMPv2-SMI
     DateAndTime, DisplayString, InstancePointer, RowStatus,
     TruthValue
          FROM SNMPv2-TC
     OBJECT-GROUP FROM SNMPv2-CONF;

t1taSNMPv2 MODULE-IDENTITY
--   ID of MODULE-IDENTITY is same as for containing module
--   except stripped of the prefix "MIB".
     LAST-UPDATED "9305250000Z"
     ORGANIZATION "Network Management Forum"
     CONTACT-INFO
     "Owen Newnan * U S WEST Advanced Technologies
     4001 Discovery Avenue Suite 190
     Boulder, CO 80303
     303 541-6253 fax x6250 * onewnan@advtech.uswest.com"

     DESCRIPTION
     "Provides sample SNMPv2 output for the (informative)
     example of IIMCOMIBTRANS."

::= { t1ta 0 2 }    -- SNMPv2 registry of this translation.


-- The example which follows is similar to what might be
-- generated using a mechanized translator except that
-- it is liberally interspersed with comments.

-- If this were a normative example, a production of the form
--   t1ta OBJECT IDENTIFIER ::= { ... } would be present.

t1ta OBJECT IDENTIFIER ::= { TBD }


-- IMPORTS may of course include a minimalistic subset as an
-- optimization (as illustrated above), however,
-- boilerplate for generated code may contain IMPORTS not
-- used.

-- Generated code must clearly indicated ASN.1 registry of
-- the module, in the DEFINITIONS production and/or in
-- comments.  OIDs for trAttribute and
-- trMobjectClass would also be provided for a real MIB.


-- ****************************************************

-- This example illustrates a single translated object
-- group class table, a side table (corresponding to a
-- multi-valued attribute) and a child table.

-- This example illustrates among other things
-- translation of enumerateds, timestamps, BOOLEANs,
-- CHOICEs, ObjectInstance's and default values to SNMPv2.
```

```
-- It also illustrates similarities and differences between
-- translated SNMPv1 and SNMPv2 outputs.

-- ******************************************************

-- The general structure of this module (indentation
-- reflects the OID structure) is:

-- t1taTroubleReportTable
  -- t1taTroubleReportTableEntry
     -- t1taTroubleReportIndex
     -- t1taTroubleReportCancelRequestedByManager
     -- t1taTroubleReportManagedObjectInstance
     -- t1taTroubleReportReceivedTime
     -- t1taTroubleReportTroubleFoundNumber
     -- t1taTroubleReportTroubleFoundIdentifier
     -- t1taTroubleReportPerceivedTroubleSeverity
     -- t1taTroubleReportRowParent
     -- t1taTroubleReportRowNameBinding
     -- t1taTroubleReportRowStatus

  -- t1taTroubleReportAccessTimeLocationListTable
    -- t1taTroubleReportAccessTimeLocationListTableEntry
       -- t1taTroubleReportAccessTimeLocationListClassIndex
       -- t1taTroubleReportAccessTimeLocationListValueIndex
       -- t1taTroubleReportAccessTimeLocationListPremisesName
       -- t1taTroubleReportAccessTimeLocationList-
       --      PremisesAddress
       -- t1taTroubleReportAccessTimeLocationListRowStatus

  -- t1taTroubleReportChildTable
    -- t1taTroubleReportChildTableEntry
       -- t1taTroubleReportChildClassIndex
       -- t1taTroubleReportChildValueIndex
       -- t1taTroubleReportChild

-- ******************************************************

t1taTroubleReportTable OBJECT-TYPE
--   class table
     SYNTAX SEQUENCE OF T1TATroubleReportTableEntry
     MAX-ACCESS  not-accessible
     STATUS  current

     DESCRIPTION
     "t1taTroubleReport class table."

     REFERENCE "trMobjectClass 5"
::= { t1ta 5 }
--   5th object in TA Input and Output Subtrees

t1taTroubleReportTableEntry OBJECT-TYPE
--   class table entry
     SYNTAX   T1TATroubleReportTableEntry
```

```
     MAX-ACCESS  not-accessible
     STATUS  current

     DESCRIPTION
     "t1taTroubleReportTable instance"

     INDEX  { t1taTroubleReportIndex }
::= { t1taTroubleReportTable 1 }

T1TATroubleReportTableEntry   ::= SEQUENCE {
     t1taTroubleReportIndex
          Integer32,
     t1taTroubleReportCancelRequestedByManager
          TruthValue,
     t1taTroubleReportManagedObjectInstance
          InstancePointer,
     t1taTroubleReportReceivedTime
          DateAndTime,
     t1taTroubleReportTroubleFoundNumber
          Integer32,
     t1taTroubleReportTroubleFoundIdentifier
          OBJECT IDENTIFIER,
     t1taTroubleReportPerceivedTroubleSeverity
          Integer32,
     t1taTroubleReportParent
          InstancePointer,
     t1taTroubleReportNameBinding
          OBJECT IDENTIFIER,
     t1taTroubleReportRowStatus
          RowStatus
          }

t1taTroubleReportIndex OBJECT-TYPE
     SYNTAX Integer32
     MAX-ACCESS  not-accessible
     STATUS  current

     DESCRIPTION
     "Provides the index by which instances of
     t1taTroubleReport table entries are distinguished."

::= { t1taTroubleReportTableEntry 1 }

t1taTroubleReportCancelRequestedByManager OBJECT-TYPE
     SYNTAX  TruthValue
     MAX-ACCESS read-create
     STATUS current

     DESCRIPTION
     "The Cancel Requested By Manager attribute indicates
     whether the manager has initiated the process to cancel
     a Trouble Report."

     REFERENCE "trAttribute 12"
```

```
     DEFVAL { 1 }   --   TruthValue usage for FALSE
--        troubleReportCancelRequestedByManagerDefault
--             BOOLEAN ::=  FALSE
::= { t1taTroubleReportTableEntry 2 }

t1taTroubleReportManagedObjectInstance OBJECT-TYPE
     SYNTAX InstancePointer
     MAX-ACCESS read-create
     STATUS current

     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 3 }

t1taTroubleReportReceivedTime OBJECT-TYPE
     SYNTAX  DateAndTime
     MAX-ACCESS  read-only
     STATUS  current

     DESCRIPTION
     "The Received Time attribute indicates the date and time
     when a trouble report was entered."

     REFERENCE "trAttribute 33"
::= { t1taTroubleReportTableEntry 4 }

-- Note that t1taTroubleReportAccessTimeLocationList is not
-- assigned arc 5 because it is implemented as a side table
-- due to its set valued syntax; see below.


-- The following illustrates mapping of a CHOICE.
-- Exactly one of the following two choices will be present
-- for a given class entry.

-- These are enumerated in-line (within the
-- class table) rather than in a side table because the
-- syntax cannot be multi-valued.

t1taTroubleReportTroubleFoundNumber OBJECT-TYPE
     SYNTAX  INTEGER {
--   Note that INTEGERs of enumerated value can be mapped
--   as comments or formal syntax.
               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)
                }
     MAX-ACCESS  read-only
     STATUS  current

     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 5}

t1taTroubleReportTroubleFoundIdentifier OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS  read-only
     STATUS  current

     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 6}

t1taTroubleReportPerceivedTroubleSeverity OBJECT-TYPE
     SYNTAX INTEGER {
          outOfService(32767),  -- mapped from zero
          backInService(1),
          serviceAffectingTrouble(2),
          nonServiceAffectingTrouble(3)
     }
     MAX-ACCESS  read-create
     STATUS  current
```

```
     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 7}

t1taTroubleReportParent OBJECT-TYPE
     SYNTAX InstancePointer
     MAX-ACCESS  read-create
     STATUS  current

     DESCRIPTION
     "This object is used to specify the superior class
     entry instance when creating a subordinate class
     instance. Likewise, it can be queried to determine the
     superior class instance (if any) for an
     existing class instance.

     While this object complies with the RFC 1443
     InstancePointer textual convention,
     it furthermore conforms to the
     specifications of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= { t1taTroubleReportTableEntry 8 }

t1taTroubleReportNameBinding OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS  read-create
     STATUS  current

     DESCRIPTION
     "This object allows specification of a
     specific NAME BINDING for creation of a class
     entry instance.  It can also be examined to
     determine the NAME BINDING in effect for an existing
     class table instance.

     Use of this object must conform to the specifications
     of IIMCOMIBTRANS ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 which
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= { t1taTroubleReportTableEntry 9 }

t1taTroubleReportRowStatus OBJECT-TYPE
```

```
     SYNTAX RowStatus
     MAX-ACCESS read-create
     STATUS current

     DESCRIPTION
     "This object complies with the RowStatus textual
     convention of the SNMPv2 framework per RFC 1443.

     Behavior of this object furthermore conforms to
     subsections 2.13 and 2.14 of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 }), that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states.

     These subsections specify constraints (refinements) to
     RowStatus usage beyond what is specified by RFC 1443."

::= { t1taTroubleReportTableEntry 10 }

-- ****************************************************

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

t1taTroubleReportAccessTimeLocationListTable OBJECT-TYPE
     SYNTAX SEQUENCE OF
         T1TATroubleReportAccessTimeLocationListTableEntry
     MAX-ACCESS not-accessible
     STATUS current

--   This attribute is in a conditional package, so it
--   may be absent from a given conceptual row.

     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"
::= { t1taTroubleReportTable 2 }

t1taTroubleReportAccessTimeLocationListTableEntry OBJECT-TYPE
     SYNTAX T1TATroubleReportAccessTimeLocationListTableEntry
     MAX-ACCESS  not-accessible
     STATUS  current
     DESCRIPTION

     "t1taTroubleReportAccessTimeLocationListTable entry."
     INDEX  {
          t1taTroubleReportAccessTimeLocationListClassIndex,
          t1taTroubleReportAccessTimeLocationListValueIndex
          }
     ::= { t1taTroubleReportAccessTimeLocationListTable 1 }
```

```
T1TATroubleReportAccessTimeLocationListTableEntry
     ::=  SEQUENCE {
     t1taTroubleReportAccessTimeLocationListClassIndex
               Integer32,
     t1taTroubleReportAccessTimeLocationListValueIndex
               Integer32,
     t1taTroubleReportAccessTimeLocationListPremisesName
               DisplayString,
     t1taTroubleReportAccessTimeLocationListPremisesAddress
               DisplayString,
     t1taTroubleReportAccessTimeLocationListRowStatus
               RowStatus
     }

t1taTroubleReportAccessTimeLocationListClassIndex OBJECT-TYPE
     --   side table class index
     SYNTAX Integer32
     MAX-ACCESS  not-accessible
     STATUS  current
     DESCRIPTION
     "Has same value as class entry index of managed object."
::= { t1taTroubleReportAccessTimeLocationListTableEntry 1 }

t1taTroubleReportAccessTimeLocationListValueIndex OBJECT-TYPE
     --   side table value index
     SYNTAX Integer32
     MAX-ACCESS  not-accessible
     STATUS  current

     DESCRIPTION
     "Uniquely discriminates a value of
     t1taTroubleReportAccessTimeLocationList within a managed
     object"

::= { t1taTroubleReportAccessTimeLocationListTableEntry 2 }

t1taTroubleReportAccessTimeLocationListPremisesName
     OBJECT-TYPE
     SYNTAX  DisplayString
     MAX-ACCESS  read-create
     STATUS  current
     DESCRIPTION
     "The access time for a service location list premises
     name."
::= { t1taTroubleReportAccessTimeLocationListTableEntry 30 }

t1taTroubleReportAccessTimeLocationListPremisesAddress
     OBJECT-TYPE
     SYNTAX  DisplayString
     MAX-ACCESS  read-create
     STATUS  current

     DESCRIPTION
```

```
     "The access time for a service location list premises
     address."

::= { t1taTroubleReportAccessTimeLocationListTableEntry 4 }

t1taTroubleReportAccessTimeLocationListRowStatus OBJECT-TYPE
     SYNTAX RowStatus
     MAX-ACCESS read-create
     STATUS current

     DESCRIPTION
     "This object complies with the RowStatus textual
     convention of the SNMPv2 framework per RFC 1443.

     Behavior of this object furthermore conforms to
     subsections 2.13 and 2.14 of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 }), that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states.

     These subsections specify constraints (refinements) to
     RowStatus usage beyond what is specified by RFC 1443."

::= { t1taTroubleReportAccessTimeLocationListTableEntry 5 }


--  ************************************************

t1taTroubleReportChildTable OBJECT-TYPE
     SYNTAX SEQUENCE OF T1TATroubleReportChildTableEntry
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Allows children in the ISO/CCITT Management naming
     hierarchy to be returned."

::= { t1taTroubleReportTable 3 }

t1taTroubleReportChildTableEntry OBJECT-TYPE
     SYNTAX T1TATroubleReportChildTableEntry
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
          "Entry of Child table."

     INDEX  {
          t1taTroubleReportChildClassIndex,
          t1taTroubleReportChildValueIndex
     }
::= { t1taTroubleReportChildTable 1 }
```

```
T1TATroubleReportChildTableEntry ::= SEQUENCE {
     t1taTroubleReportChildClassIndex Integer32,
     t1taTroubleReportChildValueIndex Integer32,
     t1taTroubleReportChild InstancePointer
     -- RowStatus is not applicable
     }

t1taTroubleReportChildClassIndex OBJECT-TYPE
     SYNTAX Integer32
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Implicit index value of class table."

::= {t1taTroubleReportChildTableEntry 1}

t1taTroubleReportChildValueIndex OBJECT-TYPE
     SYNTAX Integer32
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Implicit index value of child table."

::= {t1taTroubleReportChildTableEntry 2}


t1taTroubleReportChild OBJECT-TYPE
     SYNTAX InstancePointer
     MAX-ACCESS read-only
     STATUS current

     DESCRIPTION
     "This parameter is typically used in conjunction with a
     get next operation to acquire class InstancePointers for
     successive child (contained) objects within the parent.

     This object complies with the RFC 1443 InstancePointer
     textual convention of the SNMPv2 framework, which
     defines an OBJECT IDENTIFIER pointer to a conceptual
     row.

     Behavior of this object furthermore conforms to the
     specifications of IIMCOMIBTRANS
     ({ iimcManagement 3 1 5 })
     subsections 2.13 and 2.14 that
     enforce consistency with OSI management
     containment relationships, abstract services and
     permissible object states."

::= {t1taTroubleReportChildTableEntry 3 }

END
```

Appendix B (Normative): Definition of Management Information

This appendix has two parts.

(1) Section B.1 explains how (to what extent) standard Definition of Management Information [ISO10165-2] notifications can be mapped into SNMP traps.

(2) Sections B.2 and B.3 provide translated ASN.1 definitions representing [ISO10165-2] notifications, for use with SNMPv1 and SNMPv2, respectively.

B.1. Notes on ISO/CCITT DMI Translation

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.

arc	notification	syntax	mandatory attributes
1	attributeValue-Change	AttributeValue-ChangeInfo	avci
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

arc	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 InstancePointer ) 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. SNMPv1 Translation of DMI

```
MIBosidmiSNMPv1 DEFINITIONS ::= BEGIN
IMPORTS
   iimcOMIBTRANS FROM IimcAssignedOIDs
   OBJECT-TYPE FROM RFC1155-SMI;


-- ******************************************************
-- OBJECT IDENTIFIERS
-- ******************************************************


-- Objects defined within this module are assigned beneath
-- the following arc:

osidmi OBJECT IDENTIFIER ::= { iimcOMIBTRANS 2 }

-- Separate arcs are assigned beneath this arc for
-- TRAP-TYPES and their support objects:

osidmiNotifications OBJECT IDENTIFIER ::=    { osidmi 1 }
osidmiAttributes OBJECT IDENTIFIER ::=       { osidmi 2 }

-- The following prefix is used to deal with the situation
-- where OSI syntax allows a choice of INTEGER or OBJECT
-- IDENTIFIER. The SNMP mapping is to OBJECT IDENTIFIER.
-- The integer option is also mapped to OBJECT IDENTIFIER
-- by concatentating the values of osidmiIntegerChoice
-- (defined below) and then the integer in question.  The
-- result is passed as the value of the VARIABLES clause.

osidmiIntegerChoice OBJECT IDENTIFIER ::=    { osidmi 3 }


-- The registration of this ASN.1 module is:
--   { osidmi 0 1 }

-- ******************************************************
-- This module contains:

--   * TRAP-TYPE definitions corresponding to the OSI
--     Definition of Management Information (DMI)
--     NOTIFICATIONs.

--   * Supporting OBJECT-TYPEs to be referenced in the
--     VARIABLES clauses of these traps.  Those objects
--     correspond to the mandatory ATTRIBUTES of the
--     associated NOTIFICATIONS.

-- ******************************************************

-- Following are SNMPv1 mappings of the standard DMI
```

```
-- NOTIFICATIONS.  Each of these traps and OBJECT-TYPES
-- may be implemented independent of the others.

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

osidmiAttributeValueChange TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiAttributeValueChangeDefinition
          }

     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

osidmiCommunicationsAlarm TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }

     DESCRIPTION
          "This notification type is used to report when the
          object detects a communications error."

     REFERENCE "ISO 10165-2: smi2Notification 2"
::=     2

osidmiEnvironmentalAlarm TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }

     DESCRIPTION
          "This notification type is used to report  a
          problem in the environment."
```

```
     REFERENCE "ISO 10165-2: smi2Notification 3"
::=     3

osidmiEquipmentAlarm TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }

     DESCRIPTION
          "This notification type is used to report  a
          failure in the equipment."

     REFERENCE "ISO 10165-2: smi2Notification 4"
::=     4

osidmiIntegrityViolation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }

     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

osidmiObjectCreation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES{
          osidmiManagedObjectInstance,
          osidmiAdditionalText
          }

     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

osidmiObjectDeletion TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES{
          osidmiManagedObjectInstance,
          osidmiAdditionalText
          }

     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

osidmiOperationalViolation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }

     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

osidmiPhysicalViolation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }

     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

osidmiProcessingErrorAlarm TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }

     DESCRIPTION
     "This notification type is used to report processing
     failure in a managed object."

     REFERENCE "ISO 10165-2: smi2Notification 10"
::=    10

osidmiQualityOfServiceAlarm TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }

     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

osidmiRelationshipChange TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiRelationshipChangeDefinition
          }

     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

osidmiSecurityServiceOrMechanismViolation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }

     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

osidmiStateChange TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiStateChangeDefinition
          }

     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

osidmiTimeDomainViolation TRAP-TYPE
     ENTERPRISE osidmiNotifications
     VARIABLES {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
```

```
     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.

--        The managedObjectInstance ATTRIBUTE,
--        which is 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.

osidmiAdditionalText  OBJECT-TYPE
     SYNTAX OCTET STRING -- OSI GraphicString
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "This attribute is used to specify  additional
     textual information in NOTIFICATIONs "

     REFERENCE "ISO 10165-2: smi2AttributeID 7"
::= {osidmiAttributes 7}

osidmiAttributeValueChangeDefinition 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"
::= {osidmiAttributes 10}

osidmiPerceivedSeverity  OBJECT-TYPE
     SYNTAX INTEGER
     -- indeterminate (32767), critical (1), major (2),
     -- minor (3), warning (4), cleared (5)
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Reflects severity of an alarm as perceived by that
     entity or service detecting it."
     -- no ISO/CCITT BEHAVIOR description exists

     REFERENCE "ISO 10165-2: smi2AttributeID 17"
::= {osidmiAttributes 17}

osidmiProbableCause  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Reflects the probable cause of an alarm as perceived by
     the entity or service detecting it."
     -- no ISO/CCITT BEHAVIOR description exists
     -- defined in ISO 10164-4

     REFERENCE "ISO 10165-2: smi2AttributeID 18"
::= {osidmiAttributes 18}

osidmiRelationshipChangeDefinition  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"
::= {osidmiAttributes 20}

osidmiSecurityAlarmCause  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"
::= {osidmiAttributes 21}

osidmiSecurityAlarmDetector  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"
::= {osidmiAttributes 22}

osidmiSecurityAlarmSeverity  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"
::= {osidmiAttributes 23}

osidmiServiceProvider 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"
::= {osidmiAttributes 24}

osidmiServiceUser 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"
::= {osidmiAttributes 25}

osidmiStateChangeDefinition  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"
::= {osidmiAttributes 28}

osidmiManagedObjectInstance    OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER -- class InstancePointer
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Provides the class InstancePointer (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"
::= {osidmiAttributes 61}


END
```

B.3. SNMPv2 Translation of DMI

```
MIBosidmiSNMPv2 DEFINITIONS ::= BEGIN

IMPORTS
     iimcOMIBTRANS
          FROM IimcAssignedOIDs
     MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
     Integer32, internet, snmpModules
          FROM SNMPv2-SMI
     TruthValue, DisplayString, TimeStamp
          FROM SNMPv2-TC
--   GraphicString
--        FROM IIMComibtransSptSNMPv2
     OBJECT-GROUP FROM SNMPv2-CONF;


osidmiSNMPv2 MODULE-IDENTITY
LAST-UPDATED "9305250000Z"
ORGANIZATION "Network Management Forum"
CONTACT-INFO
     "Owen Newnan * U S WEST Advanced Technologies
      4001 Discovery Avenue Suite 190
      Boulder, CO 80303
      303 541-6253 fax x6250 * onewnan@advtech.uswest.com"
DESCRIPTION
     "Facilitates mechanized translation of MIBs from ISO
      GDMO to SNMPv2 SMI format."
::= { onSMI2toInternetConvergenceMIB 1 }


-- ******************************************************
-- OBJECT IDENTIFIERS
-- ******************************************************

-- Objects defined within this module are assigned beneath
-- the following arc:

osidmi OBJECT IDENTIFIER ::= { iimcOMIBTRANS 2 }

-- Separate arcs are assigned beneath this arc for
-- TRAP-TYPES and their support objects:

osidmiNotifications OBJECT IDENTIFIER ::=    { osidmi 1 }
osidmiAttributes OBJECT IDENTIFIER ::=       { osidmi 2 }

-- The following prefix is used to deal with the situation
-- where OSI syntax allows a choice of INTEGER or OBJECT
-- IDENTIFIER. The SNMP mapping is to OBJECT IDENTIFIER.
-- The integer option is also mapped to OBJECT IDENTIFIER
```

```
-- by concatentating the values of osidmiIntegerChoice
-- (defined below) and then the integer in question.  The
-- result is passed as the value of the VARIABLES clause.

osidmiIntegerChoice OBJECT IDENTIFIER ::=    { osidmi 3 }


-- The registration of this ASN.1 module is:
--   { osidmi 0 2 }

-- ******************************************************
-- This module contains:

--   * NOTIFICATION-TYPE definitions corresponding to the
--     Definition of Management Information (DMI)
--     NOTIFICATIONs.

--   * Supporting OBJECT-TYPEs to be referenced in the
--     OBJECTS clauses of these notifications.  Those
--     objects correspond to the mandatory ATTRIBUTES of
--     the associated NOTIFICATIONS.

-- ******************************************************

-- Following are SNMPv2 mappings of the standard DMI
-- NOTIFICATIONSs.  Each of these traps and OBJECT-TYPES
-- may be implemented independent of the others.

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

osidmiAttributeValueChange NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiAttributeValueChangeDefinition
          }
     STATUS current

     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"
::=    { osidmiNotification  1 }

osidmiCommunicationsAlarm NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
```

```
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }
     STATUS current

     DESCRIPTION
          "This notification type is used to report when the
          object detects a communications error."

     REFERENCE "ISO 10165-2: smi2Notification 2"
::=     { osidmiNotification  2 }

osidmiEnvironmentalAlarm NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }
     STATUS current
     DESCRIPTION
          "This notification type is used to report  a
          problem in the environment."
     REFERENCE "ISO 10165-2: smi2Notification 3"
::=     { osidmiNotification  3 }

osidmiEquipmentAlarm NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }
     STATUS current

     DESCRIPTION
          "This notification type is used to report  a
          failure in the equipment."

     REFERENCE "ISO 10165-2: smi2Notification 4"
::=     { osidmiNotification  4 }

osidmiIntegrityViolation NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
     STATUS current
```

```
     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"
::=     { osidmiNotification  5 }

osidmiObjectCreation NOTIFICATION-TYPE
     OBJECTS{
          osidmiManagedObjectInstance,
          osidmiAdditionalText
          }
     STATUS current

     DESCRIPTION
          "This notification type is used to report the
          creation of a managed object to another
          open system."

     REFERENCE "ISO 10165-2: smi2Notification 6"
::=     { osidmiNotification  6 }

osidmiObjectDeletion NOTIFICATION-TYPE
     OBJECTS{
          osidmiManagedObjectInstance,
          osidmiAdditionalText
          }
     STATUS current

     DESCRIPTION
          "This notification type is used to report the
          deletion of a managed object to another
          open system."

     REFERENCE "ISO 10165-2: smi2Notification 7"
::=     { osidmiNotification  7 }

osidmiOperationalViolation NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
     STATUS current

     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"
::=     { osidmiNotification  8 }

osidmiPhysicalViolation NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
     STATUS current

     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"
::=     { osidmiNotification  9 }

osidmiProcessingErrorAlarm NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }
     STATUS current

     DESCRIPTION
     "This notification type is used to report processing
     failure in a managed object."

     REFERENCE "ISO 10165-2: smi2Notification 10"
::=    { osidmiNotification  10 }

osidmiQualityOfServiceAlarm NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiProbableCause,
          osidmiPerceivedSeverity
          }
     STATUS current

     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"
::=     { osidmiNotification  11 }

osidmiRelationshipChange NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiRelationshipChangeDefinition
          }
     STATUS current

     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"
::=    { osidmiNotification  12 }

osidmiSecurityServiceOrMechanismViolation NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
     STATUS current

     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"
::=    { osidmiNotification  13 }

osidmiStateChange NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiStateChangeDefinition
          }
     STATUS current

     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"
::=     { osidmiNotification  14 }

osidmiTimeDomainViolation NOTIFICATION-TYPE
     OBJECTS {
          osidmiManagedObjectInstance,
          osidmiAdditionalText,
          osidmiSecurityAlarmCause,
          osidmiSecurityAlarmDetector,
          osidmiSecurityAlarmSeverity,
          osidmiServiceProvider,
          osidmiServiceUser
          }
     STATUS current

     DESCRIPTION
     "This notification is used to report that an
     event has occurred at an unexpected or prohibited time"

     REFERENCE "ISO 10165-2: smi2Notification 15"
::=     { osidmiNotification  15 }

-- ******************************************************

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

osidmiAdditionalText  OBJECT-TYPE
     SYNTAX GraphicString
     -- [Editor's note: should this be OCTET STRING?
     --  The ISO form is GraphicString.]
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "This attribute is used to specify  additional
     textual information in NOTIFICATIONs "

     REFERENCE "ISO 10165-2: smi2AttributeID 7"
::= {osidmiAttribute 7}

osidmiAttributeValueChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     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"
::= {osidmiAttribute 10}

osidmiPerceivedSeverity  OBJECT-TYPE
     SYNTAX Integer32
     -- indeterminate (32767), critical (1), major (2),
     -- minor (3), warning (4), cleared (5)
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Reflects the severity of an alarm as perceived by the
     entity or service that detected it."
     -- no ISO/CCITT BEHAVIOR description exists

     REFERENCE "ISO 10165-2: smi2AttributeID 17"
::= {osidmiAttribute 17}

osidmiProbableCause  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Reflects the probable casue of an alarm as perceived by
```

```
     the entity or service that detected it."
     -- no ISO/CCITT BEHAVIOR description exists
     -- defined in ISO 10164-4

     REFERENCE "ISO 10165-2: smi2AttributeID 18"
::= {osidmiAttribute 18}

osidmiRelationshipChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     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"
::= {osidmiAttribute 20}

osidmiSecurityAlarmCause  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "This attribute specifies the cause of the
     security alarm"

     REFERENCE "ISO 10165-2: smi2AttributeID 21"
::= {osidmiAttribute 21}

osidmiSecurityAlarmDetector OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "This attribute identifies the entity that
     detected the security alarm"

     REFERENCE "ISO 10165-2: smi2AttributeID 22"
::= {osidmiAttribute 22}

osidmiSecurityAlarmSeverity  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "This attribute indicates the severity of the
```

```
     security alarm"

     REFERENCE "ISO 10165-2: smi2AttributeID 23"
::= {osidmiAttribute 23}

osidmiServiceProvider OBJECT-TYPE
     SYNTAX DisplayString
     MAX-ACCESS not-accessible
     STATUS current

     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"
::= {osidmiAttribute 24}

osidmiServiceUser OBJECT-TYPE
     SYNTAX DisplayString
     MAX-ACCESS not-accessible
     STATUS current

     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"
::= {osidmiAttribute 25}

osidmiStateChangeDefinition  OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     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"
::= {osidmiAttribute 28}

osidmiManagedObjectInstance    OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER -- class InstancePointer
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Provides the class InstancePointer  (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"
::= {osidmiAttribute 61}

END
```

Appendix C (Normative): Support Objects For Use With Translated MIBs

This Appendix provides supporting MIB specifications to assist in the use of translated OSI MIB constructs. These specifications are:

(1) the omibtransSptNextUniqueTable, an optional table that allows an agent to allocate indices for row creation on a per-table basis,

(2) the omibtransSptTrapSuppressionTable, which allows traps to be suppressed or enabled on a per TRAP-TYPE basis, and

(3) graphicString, an SNMPv2 textual convention for this commonly-used ASN.1 type.

Items (1) and (2) are provided for use with both SNMPv1 and SNMPv2; Item (3) is provided only for use with SNMPv2.

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

```
iimcOMIBTRANS OBJECT IDENTIFIER ::= { iimcManagementDocMan 5}
```

This document further allocates registration identifiers to the two MIBs currently specified within Appendix B and Appendix C of this document:

```
osidmi          OBJECT IDENTIFIER ::= { iimcOMIBTRANS 1 }
omibtransSpt    OBJECT IDENTIFIER ::= { iimcOMIBTRANS 2 }
```

Editor's Note: [Reader comment is requested how best to structure objects registered by the NM Forum that support or comply with the translation procedures of this document, considering that compact OBJECT IDENTIFIERs are especially important for SNMP usage.]

C.1 SNMPv1 Support Objects

```
IIMComibtransSptSNMPv1 DEFINITIONS ::= BEGIN
IMPORTS
     iimcOMIBTRANS FROM IimcAssignedOIDs
     OBJECT-TYPE, internet FROM RFC1155-SMI;


-- Objects defined within this module are assigned beneath
-- the following arc:
```

```
omibtransSpt OBJECT IDENTIFIER ::= { iimcOMIBTRANS 1 }


-- The registration of this ASN.1 module is:
--   { omibtransSpt 0 1 }

-- ******************************************************
--
--   This module contains:
--
--   * A table to allocate indexes which a management
--     station can use to create conceptual rows.
--
--   * A table to selectively suppress issuance of traps.
--
-- ******************************************************

omibtransSptNextUniqueIndexTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OmibtransSptNextUniqueIndexEntry
     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 that agent.  Such values are
     assigned arbitrarily by the agent, so a manager should
     make no assumption about the sequence or magnitude of
     values returned."

::= { omibtransSpt  1 }

omibtransSptNextUniqueIndexEntry OBJECT-TYPE
     SYNTAX OmibtransSptNextUniqueIndexEntry
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Entry of omibtransSptNextUniqueIndexTable."

     INDEX { omibtransSptNextUniqueIndexIndex }
::= { omibtransSptNextUniqueIndexTable 1 }

OmibtransSptNextUniqueIndexEntry ::=
     SEQUENCE {
     omibtransSptNextUniqueIndexIndex
          OBJECT IDENTIFIER,
     omibtransSptNextUniqueIndex
          INTEGER
```

```
     }

omibtransSptNextUniqueIndexIndex OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Provides a prefix indicating the type of conceptual row
     to be created by a management station."

::= { omibtransSptNextUniqueIndexEntry 1 }

omibtransSptNextUniqueIndex 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 object
     typically returns a different value each time it is
     read."

::= { omibtransSptNextUniqueIndexEntry 2 }

-- ****************************************************

omibtransSptTrapSuppressionTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OmibtransSptTrapSuppressionEntry
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Provides a means to selectively suppress issuance of
     traps under the SNMPv1 framework."

::= { omibtransSpt  2 }

omibtransSptTrapSuppressionEntry OBJECT-TYPE
     SYNTAX OmibtransSptTrapSuppressionEntry
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Entry of omibtransSptTrapSuppressionTable."

     INDEX {
          omibtransSptTrapSuppressionEnterprise,
          omibtransSptTrapSuppressionTrapNumber
          }
::= { omibtransSptTrapSuppressionTable 1 }
```

```
OmibtransSptTrapSuppressionEntry ::=
     SEQUENCE {
          omibtransSptTrapSuppressionEnterprise
               OBJECT IDENTIFIER,
          omibtransSptTrapSuppressionTrapNumber
               INTEGER,
          omibtransSptTrapSuppressionFlag
               INTEGER
          }

omibtransSptTrapSuppressionEnterprise OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Supplies the Enterprise OBJECT IDENTIFIER for the
     trap of interest."

::= { omibtransSptTrapSuppressionEntry 1 }

omibtransSptTrapSuppressionTrapNumber OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS not-accessible
     STATUS mandatory

     DESCRIPTION
     "Supplies the trap number (value of the TRAP-TYPE macro)
     for the trap of interest."

::= { omibtransSptTrapSuppressionEntry 2 }

omibtransSptTrapSuppressionFlag OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-write
     STATUS mandatory

     DESCRIPTION
     "This object determines whether traps of a given type
     will be issued or not.

     This object complies with the TruthValue textual
     convention of the SNMPv2 framework per RFC 1443.
     Thus it is functionally equivalent to a BOOLEAN type but
     assumes different values, i.e., false(1) or true(2).
     A value of true indicates a trap of the given type shall
     be suppressed."

::= { omibtransSptTrapSuppressionEntry 3 }


END
```

C.2 SNMPv2 Support Objects

```
IIMComibtransSptSNMPv2 DEFINITIONS ::= BEGIN
IMPORTS
     iimcOMIBTRANS
          FROM IimcAssignedOIDs
     MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
     Integer32, internet, snmpModules
          FROM SNMPv2-SMI
     TruthValue, DisplayString, TimeStamp
          FROM SNMPv2-TC
     OBJECT-GROUP FROM SNMPv2-CONF;


iimcomibtransSptSNMPv2 MODULE-IDENTITY
     LAST-UPDATED "9305250000Z"
     ORGANIZATION "Network Management Forum"
     CONTACT-INFO
     "Owen Newnan * U S WEST Advanced Technologies
     4001 Discovery Avenue Suite 190
     Boulder, CO 80303
     303 541-6253 fax x6250 * onewnan@advtech.uswest.com"

     DESCRIPTION
     "Facilitates mechanized translation of MIBs from OSI
     GDMO to the SNMPv1 and SNMPv2 frameworks."

::= { omibtransSpt 0 2 }


-- ******************************************************
-- OBJECT IDENTIFIERS
-- ******************************************************


-- Objects defined within this module are assigned beneath
-- the following arc:

omibtransSpt OBJECT IDENTIFIER ::= { iimcOMIBTRANS 1 }


-- The registration of this ASN.1 module is:
--   { omibtransSpt 0 2 }

-- ******************************************************
-- This module contains:

--   * A table to allocate indexes which a management
--     station can use to create conceptual rows.

--   * A textual convention corresponding to the OSI
```

```
--     GraphicString type.

-- ****************************************************

omibtransSptNextUniqueIndexTable OBJECT-TYPE
     SYNTAX SEQUENCE OF OmibtransSptNextUniqueIndexEntry
     MAX-ACCESS not-accessible
     STATUS current

     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 that agent.  Such values are
     assigned arbitrarily by the agent, so a manager should
     make no assumption about the sequence or magnitude of
     values returned."

::= { omibtransSpt  1 }

omibtransSptNextUniqueIndexEntry OBJECT-TYPE
     SYNTAX OmibtransSptNextUniqueIndexEntry
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Entry of omibtransSptNextUniqueIndexTable."

     INDEX { omibtransSptNextUniqueIndexIndex }
::= { omibtransSptNextUniqueIndexTable 1 }

OmibtransSptNextUniqueIndexEntry ::=
     SEQUENCE {
     omibtransSptNextUniqueIndexIndex
          OBJECT IDENTIFIER,
     omibtransSptNextUniqueIndex
          Integer32
     }

omibtransSptNextUniqueIndexIndex OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     MAX-ACCESS not-accessible
     STATUS current

     DESCRIPTION
     "Provides a prefix indicating the type of conceptual row
     to be created by a management station."

::= { omibtransSptNextUniqueIndexEntry 1 }

omibtransSptNextUniqueIndex OBJECT-TYPE
```

```
     SYNTAX Integer32
     MAX-ACCESS read-only
     STATUS current

     DESCRIPTION
     "Returns the next index to be used for creation of a
     conceptual row in the indicated table.
     This object typically returns a different value each
     time it is read."

::= { omibtransSptNextUniqueIndexEntry 2 }

-- ****************************************************

GraphicString ::= TEXTUAL-CONVENTION
     STATUS    current

     DESCRIPTION
     "The coding and usage of this convention is the same as
     for the ISO/CCITT GraphicString type except that the
     string is not tagged."

     REFERENCE "{ [To be provided.] }"
     SYNTAX    OCTET STRING

END
```