Network Working Group
Request for Comments: 3159
Category: Standards Track

K. McCloghrie
M. Fine
Cisco Systems
J. Seligson
K. Chan
Nortel Networks
S. Hahn
R. Sahita
Intel
A. Smith
Allegro Networks
F. Reichmeyer
PFN

August 2001

Structure of Policy Provisioning Information (SPPI)

Status of this Memo

Copyright Notice

Abstract

This document, the Structure of Policy Provisioning Information (SPPI), defines the adapted subset of SNMP's Structure of Management Information (SMI) used to write Policy Information Base (PIB) modules.

RFC 2748 defines the COPS protocol, and RFC 2749 describes how the COPS protocol is used to provide for the outsourcing of policy decisions for RSVP. Another usage of the COPS protocol, for the provisioning of policy, is introduced in RFC 3084. In this provisioning model, the policy information is viewed as a collection of Provisioning Classes (PRCs) and Provisioning Instances (PRIs) residing in a virtual information store, termed the Policy Information Base (PIB). Collections of related Provisioning Classes are defined in a PIB module.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Table of Contents

1. Use of the SMI

The SPPI and PIB modules are based on SNMP's SMI and MIB modules, which use an adapted subset of the ASN.1 data definition language [ASN1]. The decision to base the definition of PIB modules on this format allows for the leveraging of the community's knowledge, experience and tools of the SMI and MIB modules.

1.1. Terminology Translation

The SMI uses the term "managed objects" to refer to object types, both tabular types with descriptors such as xxxTable and xxxEntry, as well as scalar and columnar object types. The SPPI does not use the term "object" so as to avoid confusion with COPS protocol objects. Instead, the SPPI uses the term Provisioning Class (PRC) for the table and row definitions (the xxxTable and xxxEntry objects, respectively), and Provisioning Instance (PRI) for an instantiation of a row definition. For a columnar object of a table definition, the SPPI uses the term "attribute" of a Provisioning Class. (The SPPI does not support the equivalent of the SMI's scalar objects.)

1.2. Overview

SNMP's SMI is divided into five parts: module definitions, object definitions, notification definitions [SMI], textual convention definitions [TC] and conformance definitions [CONF].

- The SMI's MODULE-IDENTITY macro is used to convey the semantics of a MIB module. The SPPI uses this macro to convey the semantics of a PIB module.

- The SMI's OBJECT-TYPE macro is used to convey the syntax and semantics of managed objects. The SPPI uses this macro to convey the syntax and semantics of PRCs and their attributes.

- The SMI's notification definitions are not used (at this time) by the SPPI. (Note that the use of the keyword 'notify' in the SPPI is not related to the SMI's notifications).

- The SMI's TEXTUAL CONVENTION macro allows new data types to be defined. The SPPI uses this macro to define new data types having particular syntax and semantics which is common to several attributes of one of more PRCs.

- The SMI's conformance definitions define several macros: the OBJECT-GROUP macro, the NOTIFICATION-GROUP macro, the MODULE-COMPLIANCE macro and the AGENT-CAPABILITIES macro. The SPPI uses the OBJECT-GROUP and MODULE-COMPLIANCE macros to specify acceptable lower-bounds of implementation of the attributes of PRCs, and thereby indirectly, acceptable lower-bounds of implementation of the PRCs themselves. The NOTIFICATION-GROUP macro is not used (at this time) by the SPPI. Potential usage by the SPPI of the AGENT- CAPABILITIES macro is for further study.

2. Structure of this Specification

The SMI is specified in terms of an ASN.1 definition together with descriptive text for each element introduced in that ASN.1 definition. This document specifies the SPPI also via a ASN.1 definition, which is a modified version of the SMI's definition, together with descriptive text for only those elements in the SPPI's ASN.1 definition which have differences from the SMI's. For elements in the ASN.1 definition which have no descriptive text in this specification, the reader is referred to the SMI's descriptive text for that element.

3. Definitions

```
COPS-PR-SPPI DEFINITIONS ::= BEGIN

IMPORTS    ObjectName, SimpleSyntax, ExtUTCTime, mgmt
                                              FROM SNMPv2-SMI;

-- the root for PIB definitions

      pib           OBJECT IDENTIFIER ::= { mgmt 2 }

-- definitions for PIB modules

MODULE-IDENTITY MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  SubjectPart                          -- new
                  "LAST-UPDATED" value(Update ExtUTCTime)
                  "ORGANIZATION" Text
                  "CONTACT-INFO" Text
                  "DESCRIPTION" Text
                  RevisionPart

    VALUE NOTATION ::=
                  value(VALUE OBJECT IDENTIFIER)

    SubjectPart ::=                                    -- new
                  "SUBJECT-CATEGORIES" "{" Categories "}"
                   -- see IANA Considerations section
    Categories ::=                                     -- new
                  CategoryIDs
                | "all"
    CategoryIDs ::=                                    -- new
                  CategoryID
                | CategoryIDs "," CategoryID
    CategoryID ::=                                     -- new
                  identifier "(" number ")"  -- number is positive

    RevisionPart ::=
                  Revisions
                | empty
    Revisions ::=
                  Revision
                | Revisions Revision
    Revision ::=
                  "REVISION" value(Update ExtUTCTime)
                  "DESCRIPTION" Text
```

```
    -- a character string as defined in [SMI]
    Text ::= value(IA5String)
END

--

OBJECT-IDENTITY MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

    VALUE NOTATION ::=
                  value(VALUE OBJECT IDENTIFIER)

    Status ::=
                  "current"
                | "deprecated"
                | "obsolete"

    ReferPart ::=
                  "REFERENCE" Text
                | empty

    -- a character string as defined in [SMI]
    Text ::= value(IA5String)
END


-- syntax of attributes

-- the "base types" defined here are:
--   3 built-in ASN.1 types: INTEGER, OCTET STRING, OBJECT IDENTIFIER
--   7 application-defined types: Integer32, IpAddress, Unsigned32,
--         TimeTicks, Opaque, Integer64 and Unsigned64

ObjectSyntax ::=
    CHOICE {
        simple
            SimpleSyntax,

          -- note that SEQUENCEs for table and row definitions
          -- are not mentioned here...

        application-wide
            ApplicationSyntax
    }
```

```
-- application-wide types

ApplicationSyntax ::=
    CHOICE {
        ipAddress-value
            IpAddress,

        timeticks-value
            TimeTicks,

        arbitrary-value
            Opaque,

        unsigned-integer-value
            Unsigned32,

        large-integer-value                         -- new
            Integer64,

        large-unsigned-integer-value                -- new
            Unsigned64
    }


-- the following 5 types are copied from the SMI

-- indistinguishable from INTEGER, but never needs more than
-- 32-bits for a two's complement representation
Integer32 ::=
        INTEGER (-2147483648..2147483647)

-- (this is a tagged type for historical reasons)
IpAddress ::=
    [APPLICATION 0]
        IMPLICIT OCTET STRING (SIZE (4))
-- ******* THIS TYPE DEFINITION IS DEPRECATED *******
-- The IpAddress type represents a 32-bit internet
-- IPv4 address.  It is represented as an OctetString
-- of length 4, in network byte-order.
-- Note that the IpAddress type is present for
-- historical reasons. IPv4 and IPv6 addresses should
-- be represented using the INET-ADDRESS-MIB
-- defined in [INETADDR].

-- an unsigned 32-bit quantity
Unsigned32 ::=
    [APPLICATION 2]
        IMPLICIT INTEGER (0..4294967295)
```

```
-- hundredths of seconds since an epoch
TimeTicks ::=
    [APPLICATION 3]
        IMPLICIT INTEGER (0..4294967295)

--for backward compatibility only
Opaque ::=
    [APPLICATION 4]
        IMPLICIT OCTET STRING


-- the following 2 types are not present in the SMI

Integer64 ::=
    [APPLICATION 10]
        IMPLICIT INTEGER (-9223372036854775808..9223372036854775807)

Unsigned64 ::=
    [APPLICATION 11]
        IMPLICIT INTEGER (0..18446744073709551615)

-- definition for Provisioning Classes and their attributes
-- (differences from the SMI are noted in the ASN.1 comments)

OBJECT-TYPE MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  "SYNTAX" Syntax
                  UnitsPart
                  "PIB-ACCESS" Access       -- modified
                  PibReferencesPart         -- new
                  PibTagPart                -- new
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ErrorsPart                -- new
                  ReferPart
                  IndexPart                 -- modified
                  MibIndexPart              -- modified
                  UniquePart                -- new
                  DefValPart

    VALUE NOTATION ::=
                  value(VALUE ObjectName)

    Syntax ::=   -- Must be one of the following:
                       -- a base type (or its refinement),
                       -- a textual convention (or its refinement), or
                       -- a BITS pseudo-type
```

```
                 type
              | "BITS" "{" NamedBits "}"

    NamedBits ::= NamedBit
              | NamedBits "," NamedBit

    NamedBit ::=  identifier "(" number ")" -- number is nonnegative

    UnitsPart ::=
                "UNITS" Text
              | empty

    Access ::=                                    -- modified
                "install"
              | "notify"
              | "install-notify"
              | "report-only"

    Status ::=
                "current"
              | "deprecated"
              | "obsolete"

    ErrorsPart ::=                                -- new
                "INSTALL-ERRORS" "{" Errors "}"
              | empty

    Errors ::=                                    -- new
                Error
              | Errors "," Error
    Error ::=                                     -- new
                identifier "(" number ")"   -- number is positive

    ReferPart ::=
                "REFERENCE" Text
              | empty

    IndexPart ::=
                "PIB-INDEX" "{" Index "}"       -- new
              | "AUGMENTS"  "{" Entry "}"
              | "EXTENDS"   "{" Entry "}"       -- new
              | empty
    Index ::=
                  -- the correspondent OBJECT-TYPE invocation
                value(ObjectName)
    Entry ::=
                  -- use the INDEX value of the
                  -- correspondent OBJECT-TYPE invocation
```

```
                value(ObjectName)
    MibIndexPart ::=
                "INDEX"   "{" IndexTypePart "}"
              | empty
    IndexTypePart ::=
                IndexTypes
              | IndexTypes "," ImpliedIndex
              | ImpliedIndex
    IndexTypes ::=
                Index
              | IndexTypes "," Index
    ImpliedIndex ::=
                "IMPLIED" Index


    PibReferencesPart ::=
                   -- for use with ReferenceId TC
                "PIB-REFERENCES" "{" Entry "}"
              |  empty

    PibTagPart ::=
                  -- for use with TagReferenceId TC
                 "PIB-TAG" "{" Attr "}"
              |  empty

    Attr ::=       -- specifies an attribute
                   value(ObjectName)

    UniquePart ::=                                -- new
                "UNIQUENESS"    "{" UniqueTypes "}"
              | "UNIQUENESS"    "{" "}"
              |  empty
    UniqueTypes ::=
                UniqueType
              | UniqueTypes "," UniqueType
    UniqueType ::=
                  -- the correspondent OBJECT-TYPE invocation
                value(ObjectName)

    DefValPart ::= "DEFVAL" "{" Defvalue "}"
              | empty

    Defvalue ::=  -- must be valid for the type specified in
                -- SYNTAX clause of same OBJECT-TYPE macro
                value(ObjectSyntax)
              | "{" BitsValue "}"

    BitsValue ::= BitNames
```

```
                | empty

    BitNames ::=  BitName
                | BitNames "," BitName

    BitName ::= identifier

    -- a character string as defined in [SMI]
    Text ::= value(IA5String)
END


-- definitions for conformance groups

OBJECT-GROUP MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  ObjectsPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

    VALUE NOTATION ::=
                  value(VALUE OBJECT IDENTIFIER)

    ObjectsPart ::=
                  "OBJECTS" "{" Objects "}"
    Objects ::=
                  Object
                | Objects "," Object
    Object ::=
                  value(ObjectName)

    Status ::=
                  "current"
                | "deprecated"
                | "obsolete"

    ReferPart ::=
                  "REFERENCE" Text
                | empty

    -- a character string as defined in [SMI]
    Text ::= value(IA5String)
END


-- definitions for compliance statements
```

```
MODULE-COMPLIANCE MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  ModulePart

    VALUE NOTATION ::=
                  value(VALUE OBJECT IDENTIFIER)

    Status ::=
                  "current"
                | "deprecated"
                | "obsolete"

    ReferPart ::=
                  "REFERENCE" Text
                | empty

    ModulePart ::=
                  Modules
    Modules ::=
                  Module
                | Modules Module
    Module ::=
                  -- name of module --
                  "MODULE" ModuleName
                  MandatoryPart
                  CompliancePart

    ModuleName ::=
                  -- identifier must start with uppercase letter
                  identifier ModuleIdentifier
                  -- must not be empty unless contained
                  -- in MIB Module
                | empty
    ModuleIdentifier ::=
                  value(OBJECT IDENTIFIER)
                | empty

    MandatoryPart ::=
                  "MANDATORY-GROUPS" "{" Groups "}"
                | empty

    Groups ::=
                  Group
                | Groups "," Group
```

```
    Group ::=
                  value(OBJECT IDENTIFIER)

    CompliancePart ::=
                  Compliances
                | empty

    Compliances ::=
                  Compliance
                | Compliances Compliance
    Compliance ::=
                  ComplianceGroup
                | Object

    ComplianceGroup ::=
                  "GROUP" value(OBJECT IDENTIFIER)
                  "DESCRIPTION" Text

    Object ::=
                  "OBJECT" value(ObjectName)
                  InstallSyntaxPart                    -- modified
                  AccessPart
                  "DESCRIPTION" Text

    -- must be a refinement for object's SYNTAX clause
    InstallSyntaxPart ::= "SYNTAX" Syntax
                | empty

    Syntax ::=    -- Must be one of the following:
                       -- a base type (or its refinement),
                       -- a textual convention (or its refinement), or
                       -- a BITS pseudo-type
                  type
                | "BITS" "{" NamedBits "}"

    NamedBits ::= NamedBit
                | NamedBits "," NamedBit

    NamedBit ::= identifier "(" number ")" -- number is nonnegative

    AccessPart ::=
                  "PIB-MIN-ACCESS" Access            -- modified
                | empty
    Access ::=                                       -- modified
                  "not-accessible"
                | "install"
                | "notify"
                | "install-notify"
```

```
                | "report-only"

    -- a character string as defined in [SMI]
    Text ::= value(IA5String)
END

-- definition of textual conventions

TEXTUAL-CONVENTION MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  DisplayPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  "SYNTAX" Syntax

    VALUE NOTATION ::=
                   value(VALUE Syntax)      -- adapted ASN.1

    DisplayPart ::=
                  "DISPLAY-HINT" Text
                | empty

    Status ::=
                  "current"
                | "deprecated"
                | "obsolete"

    ReferPart ::=
                  "REFERENCE" Text
                | empty

    -- a character string as defined in [SMI]
    Text ::= value(IA5String)

    Syntax ::=   -- Must be one of the following:
                       -- a base type (or its refinement), or
                       -- a BITS pseudo-type
                  type
                | "BITS" "{" NamedBits "}"

    NamedBits ::= NamedBit
                | NamedBits "," NamedBit

    NamedBit ::=  identifier "(" number ")" -- number is nonnegative

END
```

```
END

COPS-PR-SPPI-TC   PIB-DEFINITIONS ::= BEGIN

IMPORTS    Unsigned32, MODULE-IDENTITY, TEXTUAL-CONVENTION, pib
                                                FROM COPS-PR-SPPI;

copsPrSppiTc    MODULE-IDENTITY
    SUBJECT-CATEGORIES   { all }
    LAST-UPDATED "200108160000Z"
    ORGANIZATION "IETF RAP WG"
    CONTACT-INFO "Keith McCloghrie
                  Cisco Systems, Inc.
                  170 West Tasman Drive,
                  San Jose, CA 95134-1706 USA
                  Phone: +1 408 526 5260
                  Email: kzm@cisco.com

                  Ravi Sahita
                  Intel
                  2111 NE 25th Avenue
                  Hillsboro, OR 97124 USA
                  Phone: +1 503 712 1554
                  Email: ravi.sahita@intel.com "
    DESCRIPTION
            "The PIB module containing a set of Textual Conventions
            which have general applicability to all PIB modules."
    REVISION     "200108160000Z"
    DESCRIPTION
         "Initial version, published in RFC 3159."
    ::= { pib 1 }

InstanceId ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The textual convention for use by an attribute which is used
        as the instance-identifying index of a PRC, i.e., an attribute
        named in a PIB-INDEX clause.  The value of an attribute with
        this syntax is always greater than zero.  PRIs of the same PRC
        need not have contiguous values for their instance-identifying
        attribute."
    SYNTAX       Unsigned32 (1..4294967295)

ReferenceId ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "A textual convention for use by an attribute which is used as
        a pointer in order to reference an instance of a particular
```

```
        PRC.  An attribute with this syntax must not be used in a
        PIB-INDEX clause , and its description must specify the
        particular PRC to which the referenced PRI will belong.
        For an attribute of this type, the referenced PRI must exist.
        Furthermore, it is an error to try to delete a PRI that is
        referenced by another instance without first deleting/modifying
        the referencing instance.  The definition of an attribute with
        this syntax can permit the attribute to have a value of zero to
        indicate that it is not currently pointing to a PRI."
    SYNTAX       Unsigned32

Prid ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
         "Represents a pointer to a PRI, i.e,. to an instance of a
         PRC.  The value is the OID name of the PRC's row definition,
         appended with one sub-identifier containing the value of the
         InstanceId value for the referenced instance.  The definition
         of an attribute with this syntax can permit the attribute to
         have a value of 0.0 to indicate that it is not currently
         pointing to a PRI."
    SYNTAX       OBJECT IDENTIFIER

TagId ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
         "Represents a tag value, such that all instances of a
         particular PRC having the same tag value form a tag list.
         A tag list is identified by the tag value shared by all
         instances in that tag list."
    SYNTAX       Unsigned32 (1..4294967295)

TagReferenceId ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
         "Represents a reference to a tag list of instances of a
         particular PRC.  The particular PRC must have an attribute
         with the syntax of TagId.  The tag list consists of
         all instances which have the same value of the TagId
         attribute.  Reference to the tag list is via the attribute
         with the syntax of TagReferenceId containing the tag
         value which identifies the tag list.
         The definition of an attribute with this syntax can permit
         the attribute to have a value of 0 to indicate that it is
         not currently referencing a tag list."
    SYNTAX       Unsigned32
END
```

4. PIB Modules

The names of all standard PIB modules must be unique (but different versions of the same module should have the same name). Developers of enterprise PIB modules are encouraged to choose names for their modules that will have a low probability of colliding with standard or other enterprise modules.

The first line of a PIB module is:

```
PIB-MODULE-NAME   PIB-DEFINITIONS ::= BEGIN
```

where PIB-MODULE-NAME is the module name.

Like the SMI, additional ASN.1 macros must not be defined in PIB modules.

4.1. Importing Definitions

Like the SMI, a PIB module which needs to reference an external definition, must use the IMPORTS statement to identify both the descriptor and the module in which the descriptor is defined, where a module is identified by its ASN.1 module name.

In particular, a PIB module imports each of the base data types that it uses from COPS-PR-SPPI (defined in this document), and may import as required from other PIB modules. A PIB module may import, from the SMI, (subtree) OIDs for the purpose of defining new OIDs. A PIB module may also import, from MIB modules, OID assignments as well as textual convention definitions providing that their underlying syntax is supported by the SPPI. However, the following must not be included in an IMPORTS statement:

- named types defined by ASN.1 itself, specifically: INTEGER, OCTET STRING, OBJECT IDENTIFIER, SEQUENCE, SEQUENCE OF type,

- the BITS construct.

For each ASN.1 macro that a PIB uses, it must import that macro's definition from the COPS-PR-SPPI.

4.2. Reserved Keywords

In addition to the reserved keywords listed in the SMI, the following must not be used as descriptors or module names:

EXTENDS INSTALL-ERRORS Integer64 PIB-MIN-ACCESS PIB-ACCESS PIB-INDEX PIB-REFERENCES PIB-TAG SUBJECT-CATEGORIES UNIQUENESS Unsigned64

5. Naming Hierarchy

The SPPI uses the same OBJECT IDENTIFIER naming hierarchy as the SMI. That is, OIDs are typically assigned to PIB modules from the subtree administered by the Internet Assigned Numbers Authority (IANA). However, like the SMI, the SPPI does not prohibit the definition of PRCs in other portions of the OID tree.

6. Mapping of the MODULE-IDENTITY macro

6.1. Mapping of the SUBJECT-CATEGORIES clause

The SUBJECT-CATEGORIES clause, which must be present, identifies one or more categories of provisioning data for which this PIB module defines provisioning information. For use with the COPS-PR protocol, the individual subject categories are mapped to COPS Client Types [COPS-PR]. IANA Considerations for SPPI SUBJECT-CATEGORIES follow the same requirements as specified in [COPS] IANA Considerations for COPS Client Types. The subject categories are identified either:

- via the keyword "all", indicating the PIB module defines provisioning information relevant for all subject categories (and thus, all COPS Client Types), or

- a list of named-number enumerations, where each number which must be greater than zero, identifies a subject category, and is mapped to the Client Type which is identified by that same number in the COPS protocol. The namespace for these named numbers is global and therefore the labels should be assigned consistently across PIB modules. At present time, no more than one named-number enumeration should be specified.

Note that the list of categories specified in a PIB module's SUBJECT-CATEGORIES clause is not exclusive. That is, some other specification might (e.g., at a future date) specify additional COPS Client Types to which the module is relevant.

When a PIB module applies to multiple subject categories, that PIB module exists in multiple virtual information stores, one for each Client-Type. A PIB module with SUBJECT-CATEGORIES "all" uses the named- number specified in the SUBJECT-CATEGORIES of the PIB it is associated with, as the COPS Client-Type when it is sent over COPS.

7. Mapping of the OBJECT-TYPE macro

The SPPI requires that all attribute definitions be contained within a PRC, i.e., within a table definition.

7.1. Mapping of the SYNTAX clause

The SYNTAX clause, which must be present within the definition of an attribute, defines the abstract data structure of that attribute. The data structure must be one of the following: a base type, the BITS construct, or a textual convention.

The SYNTAX clause must also be present for the table and row definitions of a PRC, and in this case must be a SEQUENCE OF or SEQUENCE (see section 8.1.7 below).

The base types are an extended subset of the SMI's base types:

- built-in ASN.1 types: INTEGER, OCTET STRING, OBJECT IDENTIFIER,

- application-defined types: Integer32, Unsigned32, TimeTicks, Integer64 and Unsigned64.

A textual convention is a newly-defined type defined as a sub-type of a base type [TC]. The value of an attribute whose syntax is defined using a textual convention is encoded "on-the-wire" according to the textual convention's underlying base type.

Note that the set of base types has been chosen so as to provide sufficient variety of on-the-wire encodings for attribute values; base types should contain a minimum of semantics. Semantics should, to the extent possible, be incorporated into a data type through the use of a textual convention.

The differences from the SMI in the semantics of ObjectSyntax are now described.

7.1.1. Counter32

The Counter32 type is not supported by the SPPI.

7.1.2. Gauge32

The Gauge32 type is not supported by the SPPI.

7.1.3. Opaque

The Opaque type is provided solely for backward-compatibility, and shall not be used for newly-defined object types. The Opaque type supports the capability to pass arbitrary ASN.1 syntax. A value is encoded using the ASN.1 Basic Encoding Rules [ASN1] into a string of octets. This, in turn, is encoded as an OCTET STRING, in effect "double-wrapping" the original ASN.1 value. Note that a conforming implementation need only be able to accept and recognize opaquely-encoded data. It need not be able to unwrap the data and then interpret its contents. A requirement on "standard" PIB modules is that no object may have a SYNTAX clause value of Opaque.

7.1.4. IpAddress

The IpAddress type is provided solely for backward-compatibility, and shall not be used for newly-defined object types. Instead, It is recommended to use the InetAddressType/InetAddress pair TCs as defined in RFC2851 [INETADDR].

7.1.5. Counter64

The Counter64 type is not supported by the SPPI.

7.1.6. Integer64

The Integer64 type represents integer-valued information between -2^63 and 2^63-1 inclusive (-9223372036854775808 to 9223372036854775807 decimal). While Integer64 may be sub-typed to be more constrained, if the constraint results in all possible values being contained in the range (-2147483648..2147483647), then the Integer32 type must be used instead of Integer64.

7.1.7. Unsigned64

The Unsigned64 type represents integer-valued information between 0 and 2^64-1 inclusive (0 to 18446744073709551615 decimal). While Unsigned64 may be sub-typed to be more constrained, if the constraint results in all possible values being contained in the range (0..4294967295), then the Unsigned32 type must be used instead of Unsigned64.

7.1.8. Provisioning Classes

The operations (on PIBs) supported by the SPPI apply exclusively to PRCs. Each PRC is modelled as a tabular structure, i.e., a table. Each instance of a particular PRC has the same set of attributes. The set of attributes which belong to every instance of a particular PRC is modelled as a row in the table. Note that a PRC must have no more than 127 attributes. The usage of subids (for PRC attributes) beyond 127 (that is 128 and above) is reserved for Mapping PIBs to MIBs (see Appendix A). PRCs that require more than 127 attributes must use the AUGMENTS clause to augment the PRC containing the initial 127 attributes to add additional attributes. Definition of Provisioning Classes is formalized by using the OBJECT-TYPE macro to define both:

- the PRC as a whole, called the table definition, and

- the characteristics of every instance of a particular PRC, called the row definition.

In the table definition, the SYNTAX clause has the form:

```
SEQUENCE OF <EntryType>
```

where <EntryType> refers to the SEQUENCE type of its attribute definitions. In the row definition, the SYNTAX clause has the form:

```
<EntryType>
```

where <EntryType> is a SEQUENCE type defined as follows:

```
<EntryType> ::= SEQUENCE { <type1>, ... , <typeN> }
```

where there is one <type> for each attribute, and each <type> is of the form:

```
<descriptor> <syntax>
```

where <descriptor> is the descriptor naming an attribute, and <syntax> has the value of that attribute's SYNTAX clause, except that both sub- typing information and the named values for enumerated integers or the named bits for the BITS construct, are omitted from <syntax>.

7.2. Mapping of the MAX-ACCESS clause

The MAX-ACCESS clause is not supported by the SPPI.

7.3. Mapping of the PIB-ACCESS clause

The PIB-ACCESS clause must be present for a PRC's table definition, and must not be present for any other OBJECT-TYPE definition. The PIB-ACCESS clause defines what kind of access is appropriate for the PRC.

- the value "install" is used to indicate a PRC which a PDP can install in the PEP as provisioning information.

- the value "notify" is used to indicate a PRC for which the PEP must notify the PDP of all its instances and attribute values of that PRC.

- the value "install-notify" is used to indicate the uncommon type of PRC which has both characteristics: "install" and "notify".

- the value "report-only" is used to indicate a PRC which has neither the "install" characteristic nor the "notify" characteristic. However, instances of such a PRC may be included in synchronous/asynchronous reports generated by the PEP. (Note: PRCs having the "install" and/or "notify" characteristics may also be included in reports generated by the PEP.)

7.4. Mapping of the INSTALL-ERRORS clause

The INSTALL-ERRORS clause, which may optionally be present for a PRC's table definition, and must be absent otherwise, lists one or more potential reasons for rejecting an install or a removal of an instance of the PRC. Each reason consists of a named-number enumeration, where the number represents a PRC-specific error-code to be used in a COPS protocol message, as the Error Sub-code, with the Error-Code set to priSpecificError (see [COPS-PR]). The semantics of each named-number enumeration should be described in the PRC's DESCRIPTION clause.

The numbers listed in an INSTALL-ERRORS must be greater than zero and less than 65536. If this clause is not present, an install/remove can still fail, but no PRC-specific error is available to be reported.

7.5. Mapping of the PIB-INDEX clause

The PIB-INDEX clause, which must be present for a row definition (unless an AUGMENTS or an EXTENDS clause is present instead), and must be absent otherwise, defines identification information for instances of the PRC.

The PIB-INDEX clause includes exactly one descriptor. This descriptor specifies an attribute (typically, but not necessarily of the same PRC) which is used to identify an instance of that PRC. The syntax of this attribute is REQUIRED to be InstanceId (a textual convention with an underlying syntax of Unsigned32), and it has no semantics other than its use in identifying the PRC instance. The OBJECT IDENTIFIER which identifies an instance of a PRC is formed by appending one sub- identifier to the OID which identifies that PRC's row definition. The value of the additional sub-identifier is that instance's value of the attribute specified in the INDEX clause.

Note that SPPI does not permit use of the IMPLIED keyword in a PIB-INDEX clause.

7.6. Mapping of the INDEX clause

The INDEX clause is optionally present if a PIB-INDEX clause is present, and must be absent otherwise. If present, the INDEX clause can contain any number of attributes, and is used only by the algorithmic conversion of a PIB to a MIB (see Appendix A).

An IMPLIED keyword can be present in an INDEX clause if so desired.

7.7. Mapping of the AUGMENTS clause

The AUGMENTS clause, which must not be present except in row definitions, is an alternative to the PIB-INDEX clause and the EXTENDS clause. Every row definition has exactly one of: a PIB-INDEX clause, an AUGMENTS clause, or an EXTENDS clause.

A row definition which has a PIB-INDEX clause is called a base row definition. A row definition which has an AUGMENTS clause is called a row augmentation, where the AUGMENTS clause names the base row definition which is augmented by this row augmentation. (Thus, a row augmentation cannot itself be augmented.)

A PRC whose row definition is a row augmentation is called an augmenting PRC. Instances of an augmenting PRC are identified according to the PIB-INDEX clause of the base row definition named in the AUGMENTS clause. Further, instances of an augmenting PRC exist according to the same semantics as instances of the PRC which it augments. As such, when an instance of a PRC is installed or removed, an instance of every PRC which augments it is also installed or removed. (for more details, see [COPS-PR]).

7.8. Mapping of the EXTENDS clause

The EXTENDS clause, which must not be present except in row definitions, is an alternative to the PIB-INDEX clause and the AUGMENTS clause. Every row definition has exactly one of: a PIB-INDEX clause, an AUGMENTS clause, or an EXTENDS clause.

A row definition which has an EXTENDS clause is called a sparse row augmentation, where the EXTENDS clause names the row definition which is sparsely-augmented by this sparse row augmentation. The sparsely-augmented row can be a base row definition, or another sparse row augmentation.

A PRC whose row definition is a sparse row augmentation is called a sparsely augmenting PRC. Instances of a sparsely augmenting PRC are identified according to the PIB-INDEX clause of the row definition named in the sparsely augmenting PRC's EXTENDS clause.

An instance of a sparsely augmenting PRC can not exist unless a corresponding instance of the PRC which it sparsely augments exists. As such, when an instance of a PRC is removed, an instance of any PRC which sparsely augments it is also removed. However, an instance of a sparsely augmenting PRC need not exist when the corresponding instance of the PRC that it sparsely augments exists. Thus, an instance of a sparsely augmenting PRC can be installed at the same time as or subsequent to the installation of, and can be removed prior to the removal of, the corresponding instance of the PRC that it sparsely augments. So, instances of a sparsely augmenting PRC must be installed explicitly, but are removed either implicitly (via removal of the augmented PRI) or explicitly. When a sparsely augmented PRC is installed, both instances, the instance of the sparsely augmented PRC and the instance of the sparsely augmenting PRC must be sent in one COPS message.

7.8.1. Relation between PIB-INDEX, AUGMENTS and EXTENDS clauses

When defining instance identification information for a PRC:

- If there is a one-to-one correspondence between instances of this PRC and instances of an existing PRC, then the AUGMENTS clause should be used.

- Otherwise, if there is a sparse relationship between instances of this PRC and instances of an existing PRC (that is, there is a one to zero or one correspondence between instances of a sparsely augmented PRC and the instances of the PRC that sparsely augments it.), then an EXTENDS clause should be used.

- Otherwise, a PIB-INDEX clause should be used which names its own InstanceId attribute.

7.9. Mapping of the UNIQUENESS clause

The UNIQUENESS clause, which is optionally present for any row definition, lists a set of zero or more of the PRC's attributes, for which no two instances of the PRC can have the same set of values. The specified set of attributes provide a necessary and sufficient set of values by which to identify an instance of this PRC. The attribute contained in the PIB-INDEX clause may not be present in the UNIQUENESS clause. By definition, an attribute may not appear more than once in a UNIQUENESS clause. A UNIQUENESS clause containing zero attributes indicates that it's possible for two instances of the PRC to have identical values for all attributes except, of course, for the one named in the PIB-INDEX clause.

If a PRC and its sparsely augmenting PRC both have UNIQUENESS clauses, then the UNIQUENESS constraint for instances of each PRC MUST be applied according to the UNIQUENESS clause in the corresponding PRC definition. Note that a sparsely augmenting PRC thus can override the UNIQUENESS clause of the PRC it sparsely augments.

Even though the UNIQUENESS clause is optional, its inclusion is recommended wherever it provides useful information.

7.10. Mapping of the PIB-REFERENCES clause

The PIB-REFERENCES clause, which must be present for any attribute which has the SYNTAX of ReferenceId, and must be absent otherwise, names the PRC, an instance of which is referenced by the ReferenceId attribute. For example usages of the PIB-REFERENCES clause, see Appendix B.

7.11. Mapping of the PIB-TAG clause

The PIB-TAG clause, which must be present for an attribute which has the SYNTAX TagReferenceId, and must be absent otherwise, is used to indicate that this attribute references a "tag list" of instances of another PRC. Such a tag list (similar in concept to the usage of the same term in [APPL]) is formed by all instances of the other PRC which have the same (tag) value of a particular attribute of that other PRC. The particular attribute of the other PRC, which must have the SYNTAX TagId, is named in the PIB-TAG clause. For an example usage of the PIB-TAG clause, see Appendix B.

8. Mapping of the OBJECT-IDENTITY macro

The OBJECT-IDENTITY macro is used in PIB modules to define information about an OBJECT IDENTIFIER assignment.

9. Mapping of the OBJECT-GROUP macro

For conformance purposes, it is useful to define a conformance group as a collection of related PRCs and their attributes. The OBJECT-GROUP macro (directly) defines the collection of attributes which belong to a conformance group. Since each attribute included in the collection belongs to a PRC, the collection of related PRCs which belong to a conformance group is also specified (indirectly) as the set of PRCs to which the included attributes belong.

9.1. Mapping of the OBJECTS clause

The OBJECTS clause, which must be present, is used to specify each attribute contained in the conformance group. Each of the specified attributes must be defined in the same PIB module as the OBJECT-GROUP macro appears.

It is required that every attribute defined in a PIB module be contained in at least one conformance group. This avoids the common error of adding a new attribute to a module and forgetting to add the new attribute to a group.

10. Mapping of the MODULE-COMPLIANCE macro

The MODULE-COMPLIANCE macro is used to convey a minimum set of requirements with respect to implementation of one or more PIB modules.

A requirement on all "standard" PIB modules is that a corresponding MODULE-COMPLIANCE specification is also defined, either in the same module or in a companion module.

10.1. Mapping of the MODULE clause

The MODULE clause, which must be present, is repeatedly used to name each PIB module for which compliance requirements are being specified. Each PIB module is named by its module name, and optionally, by its associated OBJECT IDENTIFIER as well. The module name can be omitted when the MODULE-COMPLIANCE invocation occurs inside a PIB module, to refer to the encompassing PIB module.

10.1.1. Mapping of the MANDATORY-GROUPS clause

The MANDATORY-GROUPS clause, which need not be present, names the one or more conformance groups within the correspondent PIB module which are unconditionally mandatory for implementation. If an agent claims compliance to the PIB module, then it must implement each and every attribute (and therefore the PRCs to which they belong) within each conformance group listed.

10.1.2. Mapping of the GROUP clause

The GROUP clause, which need not be present, is repeatedly used to name each conformance group which is conditionally mandatory for compliance to the PIB module. The GROUP clause can also be used to name unconditionally optional groups. A group named in a GROUP clause must be absent from the correspondent MANDATORY-GROUPS clause.

Conditionally mandatory groups include those which are mandatory only if a particular protocol is implemented, or only if another group is implemented. A GROUP clause's DESCRIPTION specifies the conditions under which the group is conditionally mandatory.

A group which is named in neither a MANDATORY-GROUPS clause nor a GROUP clause, is unconditionally optional for compliance to the PIB module.

10.1.3. Mapping of the OBJECT clause

The OBJECT clause, which need not be present, is repeatedly used to specify each attribute for which compliance has a refined requirement with respect to the PIB module definition. The attribute must be present in one of the conformance groups named in the correspondent MANDATORY-GROUPS clause or GROUP clauses.

By definition, each attribute specified in an OBJECT clause follows a MODULE clause which names the PIB module in which that attribute is defined. Therefore, the use of an IMPORTS statement, to specify from where such attributes are imported, is redundant and is not required in a PIB module.

10.1.3.1. Mapping of the SYNTAX clause

The SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the attribute named in the correspondent OBJECT clause. The refined syntax is the minimum level of support needed for this attribute in order to be compliant.

10.1.3.2. Mapping of the WRITE-SYNTAX clause

The WRITE-SYNTAX clause is not supported by the SPPI.

10.1.3.3. Mapping of the PIB-MIN-ACCESS clause

The PIB-MIN-ACCESS clause, which need not be present, is used to define the minimal level of access for the attribute named in the correspondent OBJECT clause. If this clause is absent, the minimal level of access is the same as the maximal level specified in the PIB-ACCESS clause of the correspondent invocation of the OBJECT-TYPE macro. If present, this clause must specify a subset of the access specified in the correspondent PIB-ACCESS clause, where: "install" is a subset of "install-notify", "notify" is a subset of "install-notify", and "not- accessible" is a subset of all other values.

An implementation is compliant if the level of access it provides is the same or a superset of the minimal level in the MODULE-COMPLIANCE macro and the same or a subset of the maximal level in the PIB-ACCESS clause.

11. Textual Conventions

When designing a PIB module, it is often useful to define new data types similar to those defined in the SPPI. In comparison to a type defined in the SPPI, each of these new types has a different name, a similar syntax, and specific semantics. These newly defined types are termed textual conventions, and are used for the convenience of humans reading the PIB module.

Attributes defined using a textual convention are always encoded by means of the rules that define their underlying type.

11.1. Mapping of the TEXTUAL-CONVENTION macro

The TEXTUAL-CONVENTION macro is used to convey the syntax and semantics associated with a textual convention. It should be noted that the expansion of the TEXTUAL-CONVENTION macro is something which conceptually happens during implementation and not during run-time.

The name of a textual convention must consist of one or more letters or digits, with the initial character being an upper case letter. The name must not conflict with any of the reserved words listed in section 5.2, should not consist of all upper case letters, and shall not exceed 64 characters in length. (However, names longer than 32 characters are not recommended.) The hyphen is not allowed in the name of a textual convention (except for use in information modules

converted from SMIv1 which allowed hyphens in ASN.1 type assignments). Further, all names used for the textual conventions defined in all "standard" PIB modules shall be unique.

11.1.1. Mapping of the DISPLAY-HINT clause

The DISPLAY-HINT clause, which need not be present, gives a hint as to how the value of an instance of an object with the syntax defined using this textual convention might be displayed. The DISPLAY-HINT clause must not be present if the Textual Convention is defined with a syntax of: OBJECT IDENTIFIER, or any enumerated syntax (BITS or INTEGER). The determination of whether it makes sense for other syntax types is dependent on the specific definition of the Textual Convention.

The rules for the format specification of the hint are the same as specified in Section 3.1 of [TC].

11.1.2. Mapping of the SYNTAX clause

The SYNTAX clause, which must be present, defines abstract data structure corresponding to the textual convention. The data structure must be one of the following: a base type (see the SYNTAX clause of an OBJECT-TYPE macro), or the BITS construct. Note that this means that the SYNTAX clause of a Textual Convention can not refer to a previously defined Textual Convention.

11.1.2.1. Sub-typing of Textual Conventions

The SYNTAX clause of a TEXTUAL CONVENTION macro may be sub-typed in the same way as the SYNTAX clause of an OBJECT-TYPE macro.

12. Extending a PIB Module

PIBs may be revised as implementation experience is gained. However, changes with potential to cause disruption to interoperability between the previous PIB and the revised PIB are not allowed.

12.1. PIB Modules

For any change, the invocation of the MODULE-IDENTITY macro must be updated to include information about the revision: specifically, updating the LAST-UPDATED clause, adding a pair of REVISION and DESCRIPTION clauses, and making any necessary changes to existing clauses, including the ORGANIZATION and CONTACT-INFO clauses.

Note that any definition contained in an existing PIB is available to be IMPORT-ed by any other PIB, and is referenced in an IMPORTS clause via the PIB module name. Thus, a PIB module name should not be changed. Definitions should not be moved from one PIB to another.

Also note that obsolete definitions must not be removed from PIB modules since their descriptors may still be referenced by other PIB modules, and the OBJECT IDENTIFIERs used to name them must never be re-assigned. The EXTENDS/AUGMENTS clause should be used to extend previous definitions depending on the information to be represented.

Changes to an existing PIB can be made in several ways:

- Additional PRCs can be added to a PIB or an existing one deprecated.

- Attributes can be added to, or deprecated from, an existing PRC. Note that an ASN.1 value of the correct type or an ASN.1 NULL value must be sent even for deprecated attributes to maintain interoperability. New attributes must be added in sequence after the existing ones.

- An existing PRC can be extended or augmented with a new PRC defined in another (perhaps enterprise specific) PIB.

Additional named-number enumerations may be added to a SUBJECT-CATEGORIES clause.

12.2. Object Assignments

If any non-editorial change is made to any clause of a object assignment, then the OBJECT IDENTIFIER value associated with that object assignment must also be changed, along with its associated descriptor. Note that the max subid for PRC attributes is 127 (See Section 7.1.8)

12.3. Object Definitions

An object definition may be revised in any of the following ways:

- A SYNTAX clause containing an enumerated INTEGER may have new enumerations added or existing labels changed. Similarly, named bits may be added or existing labels changed for the BITS construct.

- The value of a SYNTAX clause may be replaced by a textual convention, providing the textual convention is defined to use the same primitive ASN.1 type, has the same set of values, and has identical semantics.

- A UNITS clause may be added.

- A STATUS clause value of "current" may be revised as "deprecated" or "obsolete". Similarly, a STATUS clause value of "deprecated" may be revised as "obsolete". When making such a change, the DESCRIPTION clause should be updated to explain the rationale.

- Clarifications and additional information may be included in the DESCRIPTION clause.

- An INSTALL-ERRORS clause may be added or an existing INSTALL-ERRORS clause have additional errors defined.

- A REFERENCE clause may be added or updated.

- A DEFVAL clause may be added or updated.

- A PRC may be augmented by adding new objects at the end of the row, and making the corresponding update to the SEQUENCE definition.

- Entirely new objects may be defined, named with previously unassigned OBJECT IDENTIFIER values.

Otherwise, if the semantics of any previously defined object are changed (i.e., if a non-editorial change is made to any clause other than those specifically allowed above), then the OBJECT IDENTIFIER value associated with that object must also be changed. Note that changing the descriptor associated with an existing object is considered a semantic change, as these strings may be used in an IMPORTS statement.

Appendix A: Mapping a PIB to a MIB

Since the SPPI is modelled on the SMI, a PIB can be potentially algorithmically mapped into a MIB. This mapping is achieved by means of the following rules:

- Modify the module's module name by appending "-MIB" to the name.
- Change the OID assigned to the MODULE-IDENTITY to be different value.
- Replace the keyword PIB-DEFINITIONS with the keyword DEFINITIONS.
- Modify the module names of all external references to PIB modules by appending "-MIB" to each such module name.
- For each PRC definition, if an INDEX clause is absent, change the "PIB-INDEX" keyword to "INDEX"; otherwise, delete the PIB-INDEX clause.
- Delete all of the following clauses: PIB-ACCESS, PIB-REFERENCES, PIB-TAG, UNIQUENESS, INSTALL-ERRORS, and SUBJECT-CATEGORIES.
- Change all PIB-MIN-ACCESS clauses to MIN-ACCESS clauses, modifying "install" and "install-notify" to "read-create", and "notify" to "read-only".
- Add a MAX-ACCESS clause for each OBJECT-TYPE. For each table definition and row definition, the MAX-ACCESS is "not-accessible". For each attribute that is in the INDEX clause, the MAX-ACCESS is "not-accessible". For the remaining attributes, the MAX-ACCESS is "read-create".
- Add a columnar attribute of type RowStatus with a descriptor and appropriate DESCRIPTION. The descriptor can be formed by appending the nine characters "RowStatus" to the end of the PRC's descriptor (truncated if necessary to avoid the resulting descriptor being too long). A Subid beyond 127 (i.e., 128 and above) can be used as the OID for this columnar attribute.
- Modify any SYNTAX clause which has a base data type which is not allowed in the SMI, either to be a valid SMI data type or to omit the OBJECT-TYPE or TEXTUAL-CONVENTION definition and all references to it. Since it is not clear (at this time) which is the best SMI data type to use, the conversion SHOULD provide a configurable option allowing a choice from at least the following:

- convert to an OCTET STRING of the relevant size. Specifically, this option would map both Integer64 and Unsigned64 to OCTET STRING (SIZE(8)), or

- omit them from the conversion, or

- map Integer64 and Unsigned64 to Counter64 (even though this has problems representing negative numbers, and unwanted counter semantics.)

Appendix B: Example usage of PIB-REFERENCES and PIB-TAG clauses

The following example demonstrates the use of the PIB-REFERENCES and PIB-TAG clauses.

In this example, the PIB-REFERENCES clause is used by the qosIfDscpMapQueue attribute to indicate the PRC of which it references an instance, and similarly, by the qosIfDscpMapThresh attribute.

The qosIfDscpMapTable PRC has an instance for each DSCP of a particular "map", but there is no PRC defined for a map itself; rather, a map consists of all instances of qosIfDscpMapTable which have the same value of qosIfDscpMapMapId. That is, a tag list is formed by all instances of qosIfDscpMapTable which have the same value of qosIfDscpMapMapId. This tag list is referenced by the attribute qosIfDscpAssignDscpMap, and its use of the PIB-TAG clause indicates this.

```
qosIfDscpAssignTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF QosIfDscpAssignEntry
    PIB-ACCESS      install
    STATUS          current
    DESCRIPTION " "
    ::= { qosIfParameters 9 }

qosIfDscpAssignEntry OBJECT-TYPE
    SYNTAX          QosIfDscpAssignEntry
    STATUS          current
    DESCRIPTION
        "An instance of the qosIfDscpAssign class."
    PIB-INDEX       { qosIfDscpAssignPrid }
    UNIQUENESS      { qosIfDscpAssignName, qosIfDscpAssignRoles }
    ::= { qosIfDscpAssignTable 1 }

QosIfDscpAssignEntry ::= SEQUENCE {
        qosIfDscpAssignPrid       InstanceId,
        qosIfDscpAssignName       SnmpAdminString,
```

```
        qosIfDscpAssignRoles      RoleCombination,
        qosIfDscpAssignDscpMap    TagReferenceId
}

qosIfDscpAssignDscpMap OBJECT-TYPE
    SYNTAX         TagReferenceId
    PIB-TAG        { qosIfDscpMapMapId }  -- attribute defined below
    STATUS         current
    DESCRIPTION
        "The DSCP map which is applied to interfaces of type
        qosIfDscpAssignName which have a role combination of
        qosIfDscpAssignRoles."
    ::= { qosIfDscpAssignEntry 3 }

--
-- DSCP to Queue and Threshold Mapping Table
--

qosIfDscpMapTable OBJECT-TYPE
    SYNTAX         SEQUENCE OF QosIfDscpMapEntry
    PIB-ACCESS     install
    STATUS         current
    DESCRIPTION
        "Assigns DSCP values to queues and thresholds for an arbitrary
        DSCP map.  This map can then be assigned to various interface
        and role combination pairs."
    ::= { qosIfParameters 10 }

qosIfDscpMapEntry OBJECT-TYPE
    SYNTAX         QosIfDscpMapEntry
    STATUS         current
    DESCRIPTION
        "An instance of the qosIfDscpMap class."
    PIB-INDEX      { qosIfDscpMapPrid }
    UNIQUENESS     { qosIfDscpMapMapId, qosIfDscpMapDscp }
    ::= { qosIfDscpMapTable 1 }

QosIfDscpMapEntry ::= SEQUENCE {
        qosIfDscpMapPrid       InstanceId,
        qosIfDscpMapMapId      TagId,
        qosIfDscpMapDscp       Dscp,
        qosIfDscpMapQueue      ReferenceId,
        qosIfDscpMapThresh     ReferenceId
}

qosIfDscpMapMapId OBJECT-TYPE
    SYNTAX         TagId
    STATUS         current
```

```
    DESCRIPTION
        "An integer that identifies the DSCP map to which this PRI
        belongs."
    ::= { qosIfDscpMapEntry 2 }

qosIfDscpMapQueue OBJECT-TYPE
    SYNTAX         ReferenceId
    PIB-REFERENCES { qosIfQueueEntry }
    STATUS         current
    DESCRIPTION
        "This attribute maps the DSCP specified by qosIfDscpMapDscp to
        the queue identified by qosIfQueuePrid in qosIfQueueTable.
        For a given DSCP map, all the queues must belong to a single
        queue set."
    ::= { qosIfDscpMapEntry 4 }

qosIfDscpMapThresh OBJECT-TYPE
    SYNTAX         ReferenceId
    PIB-REFERENCES { qosIfThresholdEntry }
    STATUS         current
    DESCRIPTION
        "This attribute maps the DSCP specified by qosIfDscpMapDscp to
        the threshold identified by qosIfThresholdId in
        qosIfThresholdTable.  The threshold set to which this
        threshold belongs must be assigned to the queue specified by
        qosIfDscpMapQueue."
    ::= { qosIfDscpMapEntry 5 }
```

Security Considerations

This document defines a language with which to define provisioning information. The language itself has no security impact on the Internet.

IANA Considerations

The root of the subtree administered by the Internet Assigned Numbers Authority (IANA) for the Internet is:

```
internet       OBJECT IDENTIFIER ::= { iso 3 6 1 }
```

That is, the Internet subtree of OBJECT IDENTIFIERs starts with the prefix:

```
1.3.6.1.
```

Several branches underneath this subtree are used for network management:

```
   mgmt           OBJECT IDENTIFIER ::= { internet 2 }
   experimental   OBJECT IDENTIFIER ::= { internet 3 }
   private        OBJECT IDENTIFIER ::= { internet 4 }
   enterprises    OBJECT IDENTIFIER ::= { private 1 }
```

The mgmt(2) subtree is used to identify "standard" objects.

This document defines

```
   pib            OBJECT IDENTIFIER ::= { mgmt 2 }
```

as the root for PIBs defined to be carried over [COPS-PR]. This Object Identifier is a high level assignment that needs to be registered with [IANA]. Root Object Identifiers for future "standards track" PIBs will also need to be registered and MUST use Object Identifiers below this oid. A standards track PIB can only be assigned an OID by IANA if the PIB is approved by the IESG as a "standards track" document. Experimental and enterprise PIBs MUST be defined under the "experimental" and "enterprises" Object Identifiers respectively.

The PIB module "copsPrSppiTc" is defined in this document as a standard module and hence, needs a subid assignment under the "pib" oid from IANA.

SPPI SUBJECT-CATEGORIES are mapped to COPS Client Types. IANA Considerations for SUBJECT-CATEGORIES follow the same requirements as specified in [COPS] IANA Considerations for COPS Client Types. Thus, a new PIB can define a new COPS Client Type in the "standards", "experimental" or "enterprise" space, and when approved that would mean that a new COPS Client Type gets assigned. IANA must update the registry for COPS Client Types (where applicable as described in [COPS] IANA Considerations) as a result.

Authors' Addresses

Keith McCloghrie
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706 USA

Phone: +1 408 526 5260
Email: kzm@cisco.com

Michael Fine
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706 USA

Phone: +1 408 527 8218
EMail: mfine@cisco.com

John Seligson
Nortel Networks, Inc.
4401 Great America Parkway
Santa Clara, CA 95054 USA

Phone: +1 408 495 2992
EMail: jseligso@nortelnetworks.com

Kwok Ho Chan
Nortel Networks, Inc.
600 Technology Park Drive
Billerica, MA 01821 USA

Phone: +1 978 288 8175
EMail: khchan@nortelnetworks.com

Scott Hahn
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124 USA

Phone: +1 503 264 8231
EMail: scott.hahn@intel.com

Ravi Sahita
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124 USA

Phone: +1 503 712 1554
EMail: ravi.sahita@intel.com

Andrew Smith
Allegro Networks
6399 San Ignacio Ave.
San Jose, CA 95119 USA

Fax: +1 415 345 1827
EMail: andrew@allegronetworks.com

Francis Reichmeyer
PFN Inc.
University Park at MIT
26 Landsdowne Street
Cambridge, MA 02139 USA

Phone: +1 617 494 9980
EMail: franr@pfn.com

References

[COPS] Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R. and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[COPS-RSVP] Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R. and A. Sastry, " COPS usage for RSVP", RFC 2749, January 2000.

[COPS-PR] Reichmeyer, F., Herzog, S., Chan, K., Durham, D., Yavatkar, R., Gai, S., McCloghrie, K. and A. Smith, "COPS Usage for Policy Provisioning", RFC 3084, March 2001.

[SMI] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[TC] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[CONF] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[APPL] Levi, D., Meyer, P. and B. Stewart, "SNMP Applications", RFC 2573, April 1999.

[ASN1] Information processing systems -- Open Systems Interconnection -- Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, December 1987.

[INETADDR] Daniele, M., Haberman, B., Routhier, S. and J. Schoenwaelder "Textual Conventions for Internet Network Addresses", RFC 2851, June 2000.

[IANA] http://www.isi.edu/in-notes/iana/assignments/smi-numbers

[IANA-CONSIDERATIONS] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.