

Network Working Group  
Internet Draft

K. McCloghrie  
M. Fine  
Cisco Systems  
J. Seligson  
K. Chan  
Nortel Networks  
S. Hahn  
R. Sahita  
Intel  
A. Smith  
No Affiliation  
F. Reichmeyer  
PFN

20 September 2000

## **Structure of Policy Provisioning Information (SPPI)**

[draft-ietf-rap-sppi-02.txt](#)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#). 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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Expires March 2001

[Page 1]

## **1. Introduction**

[RFC 2748](#) [[COPS](#)] defines the COPS protocol, and [RFC 2749](#) [[COPS-RSVP](#)] 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 [[COPS-PR](#)]. 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. PIB modules are written using an adapted subset of SNMP's Structure of Management Information (SMI) [[SMI](#), [TC](#), [CONF](#)]. It is the purpose of this document, the Structure of Policy Provisioning Information (SPPI), to define that adapted subset.

### **1.1. Change Log**

This log to be removed as and when this draft is published as an RFC.

#### **1.1.1. Changes made in version published on 13 July 2000**

- included definition of the TEXTUAL-CONVENTION macro in the SPPI's ASN.1 module so that TC's in PIBs can use data types not present in the SMI.
- renamed the CLIENT-TYPES clause to be the SUBJECT-CATEGORIES clause in order to be more generic.
- renamed the POLICY-ACCESS clause to be the PIB-ACCESS clause for consistency. Added an extra parameter on the PIB-ACCESS clause for use as the sub-identifier for a RowStatus column when converting to a MIB.
- added new clauses: EXTENDS, PIB-INDEX, PIB-REFERENCES, PIB-TAG, and PIB-MODULES.
- renamed the MIN-ACCESS clause to be the PIB-MIN-ACCESS clause.
- created a new PIB module to contain the TC's defined in the SPPI.
- defined new TC's: Prid, PolicyTagId, PolicyTagReference.
- added Appendix with example usage of PIB-REFERENCE and PIB-TAG.
- added detail on carrying an INSTALL-ERROR in COPS-PR messages.

Expires March 2001

[Page 2]

#### **1.1.2. Changes made in version published on 20 September 2000**

- copied (unmodified) the definitions of the OBJECT-IDENTITY and OBJECT-GROUP macros into this document.
- changed syntax of PolicyTagId and PolicyTagReference to Unsigned32.
- changed the PolicyXxx TC names to remove "Policy" and be more consistent, i.e., PolicyTagId, PolicyReferenceId and PolicyTagReference to TagId, ReferenceId and TagReferenceId.
- made the UNIQUENESS clause optional, but recommended wherever it provides useful information.
- changed usage of the PIB-INDEX and INDEX clauses to be more consistent: a PIB-INDEX clause is now always required, and always has the same meaning. The INDEX clause is now optional, and is only used for the algorithmic conversion to a MIB.
- changed default OID value for an added RowStatus column to be 127.
- removed the PIB-MODULES clause.
- changed the meaning of PRC to be Provisioning Class, and of PRI to be Provisioning Instance.
- required the algorithmic conversion to a MIB to have a configurable option with respect to how Integer64 and Unsigned64 are mapped to the SMI.
- specified that 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 PIB module is relevant.
- updated the Reserved Keywords.
- copied the definitions of IpAddress, Unsigned32, TimeTicks from the SMI into COPS-PR-SPPI, and clarified that PIB modules must import each base data type that it uses from COPS-PR-SPPI, and may import, from the SMI, (subtree) OIDs for the purpose of defining new OIDs.
- various typos.

Expires March 2001

[Page 3]

## **2. 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.

### **2.1. 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.)

### **2.2. 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.
- 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-

Expires March 2001

[Page 4]

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.

### **3. 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.



#### 4. Definitions

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

IMPORTS      ObjectName, SimpleSyntax, ExtUTCTime
                                FROM SNMPv2-SMI;

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

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

Expires March 2001

[Page 6]

```
-- 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
-- 6 application-defined types: Integer32, IPAddress, Unsigned32,
--    TimeTicks, Integer64 and Unsigned64

ObjectSyntax ::=
    CHOICE {
        simple
            SimpleSyntax,

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

Expires March 2001

[Page 7]

```
        application-wide
        ApplicationSyntax
    }

-- application-wide types

ApplicationSyntax ::=
    CHOICE {
        ipAddress-value
            IPAddress,

        timeticks-value
            TimeTicks,

        unsigned-integer-value
            Unsigned32,

        large-integer-value
            Integer64,
            -- new

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

-- the following 4 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))

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

-- hundredths of seconds since an epoch
TimeTicks ::=
    [APPLICATION 3]
```

Expires March 2001

[Page 8]

```
IMPLICIT INTEGER (0..4294967295)
```

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

```
Integer64 ::=
```

```
  [APPLICATION 7]
```

```
    IMPLICIT INTEGER (-9223372036854775808..9223372036854775807)
```

```
Unsigned64
```

```
  [APPLICATION 8]
```

```
    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" AccessPart    -- 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
```

Expires March 2001

[Page 9]



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

UnitsPart ::=
    "UNITS" Text
  | empty

AccessPart ::=
    Access
  | Access ", " number -- number is positive

Access ::=
    "install"
  | "notify"
  | "install-notify"

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

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

Errors ::=
    Error
  | Errors ", " Error

Error ::=
    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
```

Expires March 2001

[Page 10]

```

        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)

```

Expires March 2001

[Page 11]

```
        | "{" 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
```

Expires March 2001

[Page 12]

```
-- 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 ::=
```

Expires March 2001

[Page 13]



```
        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 "}"
```

Expires March 2001

[Page 14]

```
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"

-- 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
```

Expires March 2001

[Page 15]

```
-- 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
```



Expires March 2001

[Page 17]

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 an 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."

SYNTAX            Unsigned32

END



Expires March 2001

[Page 18]

## **5. 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.

### **5.1. 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.

### **5.2. Reserved Keywords**

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

Expires March 2001

[Page 19]

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

## **6. 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.

## **7. Mapping of the MODULE-IDENTITY macro**

### **7.1. 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](#)]. 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.

Expires March 2001

[Page 20]

## **8. Mapping of the OBJECT-TYPE macro**

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

### **8.1. 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, IPAddress, 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. Thus, the IPAddress and TimeTicks data types should really be defined as textual conventions because they contain semantics. However, they are defined here as base types so as to avoid confusion with the SMI which defines them as base types.

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

#### **8.1.1. Counter32**

The Counter32 type is not supported by the SPPI.

Expires March 2001

[Page 21]

#### **8.1.2. Gauge32**

The Gauge32 type is not supported by the SPPI.

#### **8.1.3. Opaque**

The Opaque type is not supported by the SPPI.

#### **8.1.4. Counter64**

The Counter64 type is not supported by the SPPI.

#### **8.1.5. 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.

#### **8.1.6. 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.

#### **8.1.7. 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. This model 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:



Expires March 2001

[Page 22]

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>.

## **8.2. Mapping of the MAX-ACCESS clause**

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

## **8.3. 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 PIB-ACCESS clause also optionally (default value is 127) provides a number which is used in the algorithmic conversion of a PIB to a MIB (see [Appendix A](#)).

- 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".

Expires March 2001

[Page 23]

#### **8.4. 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 Sub-Error 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.

#### **8.5. 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.

#### **8.6. 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.

Expires March 2001

[Page 24]

### **8.7. 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 [\[COPS-PR\]](#)).

### **8.8. 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

Expires March 2001

[Page 25]

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.

#### **8.8.1. 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, then an EXTENDS clause should be used.
- Otherwise, a PIB-INDEX clause should be used which names its own InstanceId attribute.

#### **8.9. Mapping of the UNIQUENESS clause**

The UNIQUENESS clause, which is optionally present for any row definition which has a PIB-INDEX clause, and must be absent otherwise, 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.

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



Expires March 2001

[Page 26]

#### **8.10. 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-REFERENCE clause, see [Appendix B](#).

#### **8.11. 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.

### **9. Mapping of the OBJECT-IDENTITY macro**

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

### **10. 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.

#### **10.1. 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

Expires March 2001

[Page 27]

adding a new attribute to a module and forgetting to add the new attribute to a group.

## **11. 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.

### **11.1. 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.

#### **11.1.1. 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.

#### **11.1.2. 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.

#### **11.1.3. 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.

##### **11.1.3.1. 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.

##### **11.1.3.2. Mapping of the WRITE-SYNTAX clause**

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

##### **11.1.3.3. 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.

Expires March 2001

[Page 29]

## **12. 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.

### **12.1. 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.

#### **12.1.1. 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.

##### **12.1.1.1. 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.



Expires March 2001

[Page 30]

### **13. Extending a PIB Module**

The SMI's rules for extending an information module are augmented with the following rules:

#### **13.1. OBJECT-TYPE Definitions**

An invocation of the OBJECT-TYPE macro may also be revised in any of the following ways:

- An INSTALL-ERRORS clause may be added or an existing INSTALL-ERRORS clause have additional errors defined.
- Additional named-number enumerations may be added to a SUBJECT-CATEGORIES clause.

#### **14. Appendix A: Mapping a PIB to a MIB**

Since the SPPI is modelled on the SMI, a PIB can be easily and 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). The optional number provided by the PIB-ACCESS clause is used as the OID for this columnar attribute. If no number is provided by the PIB-ACCESS clause, then the default number 127 is used.
- 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

Expires March 2001

[Page 32]

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.)

## 15. [Appendix B](#): Example usage of PIB-REFERENCE and PIB-TAG clauses

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

In this example, the PIB-REFERENCE 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
```

Expires March 2001

[Page 34]

"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-REFERENCE    qosIfQueueTable  
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-REFERENCE    qosIfThresholdTable  
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 }

Expires March 2001

[Page 36]

## **16. Security Considerations**

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

## **17. 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  
Fax: +1 415 345 1827  
Email: ah\_smith@pacbell.net

Francis Reichmeyer  
PFN Inc.  
University Park at MIT  
26 Landsdowne Street  
Cambridge, MA 02139  
Phone: +1 617 494 9980  
Email: franr@pfn.com

## **18. 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" Internet Draft, [draft-ietf-rap-cops-pr-04.txt](#), August 2000.

### **[SMI]**

McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser. "Structure of Management Information Version 2 (SMIV2)", [RFC 2578](#), April 1999.

[TC] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser. "Textual Conventions for SMIV2", [RFC 2579](#), April 1999.



## [CONF]

McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., and S. Waldbusser. "Conformance Statements for SMIV2", [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.



**19. Full Copyright Statement**

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."



## Table of Contents

<a href="#">1</a>	<a href="#">Introduction</a>	<a href="#">2</a>
<a href="#">1.1</a>	<a href="#">Change Log</a>	<a href="#">2</a>
<a href="#">1.1.1</a>	<a href="#">Changes made in version published on 13 July 2000</a>	<a href="#">2</a>
<a href="#">1.1.2</a>	<a href="#">Changes made in version published on 20 September 2000</a>	<a href="#">3</a>
<a href="#">2</a>	<a href="#">Use of the SMI</a>	<a href="#">4</a>
<a href="#">2.1</a>	<a href="#">Terminology Translation</a>	<a href="#">4</a>
<a href="#">2.2</a>	<a href="#">Overview</a>	<a href="#">4</a>
<a href="#">3</a>	<a href="#">Structure of this Specification</a>	<a href="#">5</a>
<a href="#">4</a>	<a href="#">Definitions</a>	<a href="#">6</a>
<a href="#">5</a>	<a href="#">PIB Modules</a>	<a href="#">19</a>
<a href="#">5.1</a>	<a href="#">Importing Definitions</a>	<a href="#">19</a>
<a href="#">5.2</a>	<a href="#">Reserved Keywords</a>	<a href="#">19</a>
<a href="#">6</a>	<a href="#">Naming Hierarchy</a>	<a href="#">20</a>
<a href="#">7</a>	<a href="#">Mapping of the MODULE-IDENTITY macro</a>	<a href="#">20</a>
<a href="#">7.1</a>	<a href="#">Mapping of the SUBJECT-CATEGORIES clause</a>	<a href="#">20</a>
<a href="#">8</a>	<a href="#">Mapping of the OBJECT-TYPE macro</a>	<a href="#">21</a>
<a href="#">8.1</a>	<a href="#">Mapping of the SYNTAX clause</a>	<a href="#">21</a>
<a href="#">8.1.1</a>	<a href="#">Counter32</a>	<a href="#">21</a>
<a href="#">8.1.2</a>	<a href="#">Gauge32</a>	<a href="#">22</a>
<a href="#">8.1.3</a>	<a href="#">Opaque</a>	<a href="#">22</a>
<a href="#">8.1.4</a>	<a href="#">Counter64</a>	<a href="#">22</a>
<a href="#">8.1.5</a>	<a href="#">Integer64</a>	<a href="#">22</a>
<a href="#">8.1.6</a>	<a href="#">Unsigned64</a>	<a href="#">22</a>
<a href="#">8.1.7</a>	<a href="#">Provisioning Classes</a>	<a href="#">22</a>
<a href="#">8.2</a>	<a href="#">Mapping of the MAX-ACCESS clause</a>	<a href="#">23</a>
<a href="#">8.3</a>	<a href="#">Mapping of the PIB-ACCESS clause</a>	<a href="#">23</a>
<a href="#">8.4</a>	<a href="#">Mapping of the INSTALL-ERRORS clause</a>	<a href="#">24</a>
<a href="#">8.5</a>	<a href="#">Mapping of the PIB-INDEX clause</a>	<a href="#">24</a>
<a href="#">8.6</a>	<a href="#">Mapping of the INDEX clause</a>	<a href="#">24</a>
<a href="#">8.7</a>	<a href="#">Mapping of the AUGMENTS clause</a>	<a href="#">25</a>
<a href="#">8.8</a>	<a href="#">Mapping of the EXTENDS clause</a>	<a href="#">25</a>
<a href="#">8.8.1</a>	<a href="#">Relation between PIB-INDEX, AUGMENTS and EXTENDS clauses</a>	<a href="#">26</a>
<a href="#">8.9</a>	<a href="#">Mapping of the UNIQUENESS clause</a>	<a href="#">26</a>
<a href="#">8.10</a>	<a href="#">Mapping of the PIB-REFERENCES clause</a>	<a href="#">27</a>
<a href="#">8.11</a>	<a href="#">Mapping of the PIB-TAG clause</a>	<a href="#">27</a>
<a href="#">9</a>	<a href="#">Mapping of the OBJECT-IDENTITY macro</a>	<a href="#">27</a>
<a href="#">10</a>	<a href="#">Mapping of the OBJECT-GROUP macro</a>	<a href="#">27</a>
<a href="#">10.1</a>	<a href="#">Mapping of the OBJECTS clause</a>	<a href="#">27</a>
<a href="#">11</a>	<a href="#">Mapping of the MODULE-COMPLIANCE macro</a>	<a href="#">28</a>
<a href="#">11.1</a>	<a href="#">Mapping of the MODULE clause</a>	<a href="#">28</a>
<a href="#">11.1.1</a>	<a href="#">Mapping of the MANDATORY-GROUPS clause</a>	<a href="#">28</a>

Expires March 2001

[Page 41]

<a href="#">11.1.1.2</a>	Mapping of the GROUP clause .....	<a href="#">28</a>
<a href="#">11.1.1.3</a>	Mapping of the OBJECT clause .....	<a href="#">29</a>
<a href="#">11.1.1.3.1</a>	Mapping of the SYNTAX clause .....	<a href="#">29</a>
<a href="#">11.1.1.3.2</a>	Mapping of the WRITE-SYNTAX clause .....	<a href="#">29</a>
<a href="#">11.1.1.3.3</a>	Mapping of the PIB-MIN-ACCESS clause .....	<a href="#">29</a>
<a href="#">12</a>	Textual Conventions .....	<a href="#">30</a>
<a href="#">12.1</a>	Mapping of the TEXTUAL-CONVENTION macro .....	<a href="#">30</a>
<a href="#">12.1.1</a>	Mapping of the SYNTAX clause .....	<a href="#">30</a>
<a href="#">12.1.1.1</a>	Sub-typing of Textual Conventions .....	<a href="#">30</a>
<a href="#">13</a>	Extending a PIB Module .....	<a href="#">31</a>
<a href="#">13.1</a>	OBJECT-TYPE Definitions .....	<a href="#">31</a>
<a href="#">14</a>	<a href="#">Appendix A</a> : Mapping a PIB to a MIB .....	<a href="#">32</a>
<a href="#">15</a>	<b><a href="#">Appendix B</a>: Example usage of PIB-REFERENCE and PIB-TAG clauses .....</b>	<b><a href="#">34</a></b>
<a href="#">16</a>	Security Considerations .....	<a href="#">37</a>
<a href="#">17</a>	Authors' Addresses .....	<a href="#">37</a>
<a href="#">18</a>	References .....	<a href="#">38</a>
<a href="#">19</a>	Full Copyright Statement .....	<a href="#">40</a>