Network Working Group
Request for Comments: 1904
Obsoletes: 1444
Category: Standards Track

SNMPv2 Working Group
J. Case
SNMP Research, Inc.
K. McCloghrie
Cisco Systems, Inc.
M. Rose
Dover Beach Consulting, Inc.
S. Waldbusser
International Network Services

January 1996

Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)

Status of this Memo

Table of Contents

1. Introduction 2
1.1 A Note on Terminology 3
2. Definitions 3
2.1 The OBJECT-GROUP macro 3
2.2 The NOTIFICATION-GROUP macro 4
2.3 The MODULE-COMPLIANCE macro 5
2.4 The AGENT-CAPABILITIES macro 7
3. Mapping of the OBJECT-GROUP macro 9
3.1 Mapping of the OBJECTS clause 10
3.2 Mapping of the STATUS clause 10
3.3 Mapping of the DESCRIPTION clause 10
3.4 Mapping of the REFERENCE clause 10
3.5 Mapping of the OBJECT-GROUP value 10
3.6 Usage Example 11
4. Mapping of the NOTIFICATION-GROUP macro 11
4.1 Mapping of the NOTIFICATIONS clause 11
4.2 Mapping of the STATUS clause 11
4.3 Mapping of the DESCRIPTION clause 12
4.4 Mapping of the REFERENCE clause 12
4.5 Mapping of the NOTIFICATION-GROUP value 12
4.6 Usage Example 12
5. Mapping of the MODULE-COMPLIANCE macro 12
5.1 Mapping of the STATUS clause 13

5.2 Mapping of the DESCRIPTION clause 13
5.3 Mapping of the REFERENCE clause 13
5.4 Mapping of the MODULE clause 13
5.4.1 Mapping of the MANDATORY-GROUPS clause 13
5.4.2 Mapping of the GROUP clause 14
5.4.3 Mapping of the OBJECT clause 14
5.4.3.1 Mapping of the SYNTAX clause 14
5.4.3.2 Mapping of the WRITE-SYNTAX clause 15
5.4.3.3 Mapping of the MIN-ACCESS clause 15
5.4.4 Mapping of the DESCRIPTION clause 15
5.5 Mapping of the MODULE-COMPLIANCE value 15
5.6 Usage Example 16
6. Mapping of the AGENT-CAPABILITIES macro 16
6.1 Mapping of the PRODUCT-RELEASE clause 17
6.2 Mapping of the STATUS clause 17
6.3 Mapping of the DESCRIPTION clause 17
6.4 Mapping of the REFERENCE clause 17
6.5 Mapping of the SUPPORTS clause 18
6.5.1 Mapping of the INCLUDES clause 18
6.5.2 Mapping of the VARIATION clause 18
6.5.2.1 Mapping of the SYNTAX clause 18
6.5.2.2 Mapping of the WRITE-SYNTAX clause 18
6.5.2.3 Mapping of the ACCESS clause 19
6.5.2.4 Mapping of the CREATION-REQUIRES clause 19
6.5.2.5 Mapping of the DEFVAL clause 20
6.5.2.6 Mapping of the DESCRIPTION clause 20
6.6 Mapping of the AGENT-CAPABILITIES value 20
6.7 Usage Example 20
7. Extending an Information Module 22
7.1 Conformance Groups 22
7.2 Compliance Definitions 22
7.3 Capabilities Definitions 22
8. Security Considerations 23
9. Editor’s Address 23
10. Acknowledgements 23
11. References 24

1. Introduction

A management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines authentication, authorization, access control, and privacy policies.

Management stations execute management applications which monitor and control managed elements. Managed elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled via access to their management information.

Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written using a subset of OSI's Abstract Syntax Notation One (ASN.1) [1], termed the Structure of Management Information (SMI) [2].

It may be useful to define the acceptable lower-bounds of implementation, along with the actual level of implementation achieved. It is the purpose of this document to define the notation used for these purposes.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155 (STD 16), 1157 (STD 15), and 1212 (STD 16), is termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).

2. Definitions

```
SNMPv2-CONF DEFINITIONS ::= BEGIN

-- 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(Name ObjectName)

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

    ReferPart ::=
                "REFERENCE" Text
              | empty

    -- uses the NVT ASCII character set
    Text ::= """ string """
END


-- more definitions for conformance groups

NOTIFICATION-GROUP MACRO ::=
BEGIN
    TYPE NOTATION ::=
                NotificationsPart
                "STATUS" Status
                "DESCRIPTION" Text
                ReferPart

    VALUE NOTATION ::=
                value(VALUE OBJECT IDENTIFIER)

    NotificationsPart ::=
                "NOTIFICATIONS" "{" Notifications "}"
    Notifications ::=
                Notification
              | Notifications "," Notification
    Notification ::=
                value(Name NotificationName)

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

    ReferPart ::=
                "REFERENCE" Text
              | empty

    -- uses the NVT ASCII character set
    Text ::= """ string """
```

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

    ModuleName ::=
                  modulereference ModuleIdentifier
                -- must not be empty unless contained
                -- in MIB Module
                | empty
    ModuleIdentifier ::=
                  value(ModuleID OBJECT IDENTIFIER)
                | empty

    MandatoryPart ::=
                  "MANDATORY-GROUPS" "{" Groups "}"
```

```
            | empty

    Groups ::=
                Group
            | Groups "," Group
    Group ::=
                value(Group OBJECT IDENTIFIER)

    CompliancePart ::=
                Compliances
            | empty

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

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

    Object ::=
                "OBJECT" value(Name ObjectName)
                SyntaxPart
                WriteSyntaxPart
                AccessPart
                "DESCRIPTION" Text

    -- must be a refinement for object's SYNTAX clause
    SyntaxPart ::=
                "SYNTAX" type(SYNTAX)
            | empty

    -- must be a refinement for object's SYNTAX clause
    WriteSyntaxPart ::=
                "WRITE-SYNTAX" type(WriteSYNTAX)
            | empty

    AccessPart ::=
                "MIN-ACCESS" Access
            | empty
    Access ::=
                "not-accessible"
            | "accessible-for-notify"
            | "read-only"
            | "read-write"
```

```
                | "read-create"

    -- uses the NVT ASCII character set
    Text ::= """" string """"
END


-- definitions for capabilities statements

AGENT-CAPABILITIES MACRO ::=
BEGIN
    TYPE NOTATION ::=
                  "PRODUCT-RELEASE" Text
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  ModulePart

    VALUE NOTATION ::=
                  value(VALUE OBJECT IDENTIFIER)

    Status ::=
                  "current"
                | "obsolete"

    ReferPart ::=
                "REFERENCE" Text
              | empty

    ModulePart ::=
                  Modules
                | empty
    Modules ::=
                  Module
                | Modules Module
    Module ::=
                  -- name of module --
                  "SUPPORTS" ModuleName
                  "INCLUDES" "{" Groups "}"
                  VariationPart

    ModuleName ::=
                  identifier ModuleIdentifier
    ModuleIdentifier ::=
                  value(ModuleID OBJECT IDENTIFIER)
                | empty

    Groups ::=
```

```
                Group
              | Groups "," Group
    Group ::=
                value(Name OBJECT IDENTIFIER)

    VariationPart ::=
                Variations
              | empty
    Variations ::=
                Variation
              | Variations Variation

    Variation ::=
                ObjectVariation
              | NotificationVariation

    NotificationVariation ::=
                "VARIATION" value(Name NotificationName)
                AccessPart
                "DESCRIPTION" Text

    ObjectVariation ::=
                "VARIATION" value(Name ObjectName)
                SyntaxPart
                WriteSyntaxPart
                AccessPart
                CreationPart
                DefValPart
                "DESCRIPTION" Text

    -- must be a refinement for object's SYNTAX clause
    SyntaxPart ::=
                "SYNTAX" type(SYNTAX)
              | empty

    -- must be a refinement for object's SYNTAX clause
    WriteSyntaxPart ::=
                "WRITE-SYNTAX" type(WriteSYNTAX)
              | empty

    AccessPart ::=
                "ACCESS" Access
              | empty

    Access ::=
                "not-implemented"
              -- only "not-implemented" for notifications
              | "accessible-for-notify"
```

```
                | "read-only"
                | "read-write"
                | "read-create"
                -- following is for backward-compatibility only
                | "write-only"

    CreationPart ::=
                  "CREATION-REQUIRES" "{" Cells "}"
                | empty

    Cells ::=
                  Cell
                | Cells "," Cell

    Cell ::=
                  value(Cell ObjectName)

    DefValPart ::=
                  "DEFVAL" "{" value(Defval ObjectSyntax) "}"
                | empty

    -- uses the NVT ASCII character set
    Text ::= """" string """"
END


END
```

3. Mapping of the OBJECT-GROUP macro

For conformance purposes, it is useful to define a collection of related managed objects. The OBJECT-GROUP macro is used to define each such collection of related objects. It should be noted that the expansion of the OBJECT-GROUP macro is something which conceptually happens during implementation and not during run-time.

To "implement" an object, a SNMPv2 entity acting in an agent role must return a reasonably accurate value for management protocol retrieval operations; similarly, if the object is writable, then in response to a management protocol set operation, a SNMPv2 entity must accordingly be able to reasonably influence the underlying managed entity. If a SNMPv2 entity acting in an agent role can not implement an object, the management protocol provides for the SNMPv2 entity to return an exception or error, e.g, noSuchObject [4]. Under no circumstances shall a SNMPv2 entity return a value for objects which it does not implement -- it must always return the appropriate exception or error, as described in the protocol specification [4].

3.1. Mapping of the OBJECTS clause

The OBJECTS clause, which must be present, is used to name each object contained in the conformance group. Each of the named objects must be defined in the same information module as the OBJECT-GROUP macro appears, and must have a MAX-ACCESS clause value of "accessible-for-notify", "read-only", "read-write", or "read-create".

It is required that every object defined in an information module with a MAX-ACCESS clause other than "not-accessible" be contained in at least one object group. This avoids the common error of adding a new object to an information module and forgetting to add the new object to a group.

3.2. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that the definition is obsolete, but that an implementor may wish to support the group to foster interoperability with older implementations.

3.3. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual definition of that group, along with a description of any relations to other groups. Note that generic compliance requirements should not be stated in this clause. However, implementation relationships between this group and other groups may be defined in this clause.

3.4. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a group defined in some other information module. This is useful when de-osifying a MIB module produced by some other organization.

3.5. Mapping of the OBJECT-GROUP value

The value of an invocation of the OBJECT-GROUP macro is the name of the group, which is an OBJECT IDENTIFIER, an administratively assigned name.

3.6. Usage Example

The SNMP Group [3] is described:

```
snmpGroup OBJECT-GROUP
    OBJECTS { snmpInPkts,
              snmpInBadVersions,
              snmpInASNParseErrs,
              snmpBadOperations,
              snmpSilentDrops,
              snmpProxyDrops,
              snmpEnableAuthenTraps }
    STATUS  current
    DESCRIPTION
            "A collection of objects providing basic instrumentation and
            control of an SNMPv2 entity."
    ::= { snmpMIBGroups 8 }
```

According to this invocation, the conformance group named

```
 { snmpMIBGroups 8 }
```

contains 7 objects.

4. Mapping of the NOTIFICATION-GROUP macro

For conformance purposes, it is useful to define a collection of notifications. The NOTIFICATION-GROUP macro serves this purpose. It should be noted that the expansion of the NOTIFICATION-GROUP macro is something which conceptually happens during implementation and not during run-time.

4.1. Mapping of the NOTIFICATIONS clause

The NOTIFICATIONS clause, which must be present, is used to name each notification contained in the conformance group. Each of the named notifications must be defined in the same information module as the NOTIFICATION-GROUP macro appears.

4.2. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that the definition is obsolete, but that an implementor may wish to support the group to foster

interoperability with older implementations.

4.3. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual definition of the group, along with a description of any relations to other groups. Note that generic compliance requirements should not be stated in this clause. However, implementation relationships between this group and other groups may be defined in this clause.

4.4. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a group defined in some other information module. This is useful when de-osifying a MIB module produced by some other organization.

4.5. Mapping of the NOTIFICATION-GROUP value

The value of an invocation of the NOTIFICATION-GROUP macro is the name of the group, which is an OBJECT IDENTIFIER, an administratively assigned name.

4.6. Usage Example

The SNMP Basic Notifications Group [3] is described:

```
snmpBasicNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { coldStart, authenticationFailure }
    STATUS        current
    DESCRIPTION
            "The two notifications which an SNMPv2 entity is required to
            implement."
    ::= { snmpMIBGroups 7 }
```

According to this invocation, the conformance group named

```
     { snmpMIBGroups 1 }
```

contains 2 notifications.

5. 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 MIB modules. It should be noted that the expansion of the MODULE-COMPLIANCE macro is something which conceptually happens during implementation and not during run-time.

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

5.1. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that the specification is obsolete, but that an implementor may wish to support that object to foster interoperability with older implementations.

5.2. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual definition of this compliance statement and should embody any information which would otherwise be communicated in any ASN.1 commentary annotations associated with the statement.

5.3. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a compliance statement defined in some other information module.

5.4. Mapping of the MODULE clause

The MODULE clause, which must be present, is repeatedly used to name each MIB module for which compliance requirements are being specified. Each MIB 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 MIB module, to refer to the encompassing MIB module.

5.4.1. Mapping of the MANDATORY-GROUPS clause

The MANDATORY-GROUPS clause, which need not be present, names the one or more object or notification groups within the correspondent MIB module which are unconditionally mandatory for implementation. If a SNMPv2 entity acting in an agent role claims compliance to the MIB module, then it must implement each and every object and notification within each conformance group listed. That is, if a SNMPv2 entity returns a noSuchObject exception in response to a management protocol get operation [4] for any object within any mandatory conformance group for every MIB view, or if the SNMPv2 entity cannot generate each notification listed in any conformance group under the

appropriate circumstances, then that SNMPv2 entity is not a conformant implementation of the MIB module.

5.4.2. Mapping of the GROUP clause

The GROUP clause, which need not be present, is repeatedly used to name each object and notification group which is conditionally mandatory or unconditionally optional for compliance to the MIB module. 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 MIB module.

5.4.3. Mapping of the OBJECT clause

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

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

5.4.3.1. Mapping of the SYNTAX clause

The SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent OBJECT clause. Note that if this clause and a WRITE-SYNTAX clause are both present, then this clause only applies when instances of the object named in the correspondent OBJECT clause are read.

Consult Section 9 of [2] for more information on refined syntax.

5.4.3.2. Mapping of the WRITE-SYNTAX clause

The WRITE-SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent OBJECT clause when instances of that object are written.

Consult Section 9 of [2] for more information on refined syntax.

5.4.3.3. Mapping of the MIN-ACCESS clause

The MIN-ACCESS clause, which need not be present, is used to define the minimal level of access for the object 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 correspondent invocation of the OBJECT-TYPE macro. If present, this clause must not specify a greater level of access than is specified in the correspondent invocation of the OBJECT-TYPE macro.

The level of access for certain types of objects is fixed according to their syntax definition. These types include: conceptual tables and rows, auxiliary objects, and objects with the syntax of Counter32, Counter64 (and possibly, certain types of textual conventions). A MIN-ACCESS clause should not be present for such objects.

An implementation is compliant if the level of access it provides is greater or equal to the minimal level in the MODULE-COMPLIANCE macro and less or equal to the maximal level in the OBJECT-TYPE macro.

5.4.4. Mapping of the DESCRIPTION clause

The DESCRIPTION clause must be present for each use of the GROUP or OBJECT clause. For an OBJECT clause, it contains a textual description of the refined compliance requirement. For a GROUP clause, it contains a textual description of the conditions under which the group is conditionally mandatory or unconditionally optional.

5.5. Mapping of the MODULE-COMPLIANCE value

The value of an invocation of the MODULE-COMPLIANCE macro is an OBJECT IDENTIFIER. As such, this value may be authoritatively used when referring to the compliance statement embodied by that invocation of the macro.

5.6. Usage Example

The compliance statement contained in the (hypothetical) XYZv2-MIB might be:

```
xyzMIBCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for XYZv2 entities which implement
            the XYZv2 MIB."
    MODULE  -- compliance to the containing MIB module
        MANDATORY-GROUPS { xyzSystemGroup,
                           xyzStatsGroup, xyzTrapGroup,
                           xyzSetGroup,
                           xyzBasicNotificationsGroup }

        GROUP   xyzV1Group
        DESCRIPTION
            "The xyzV1 group is mandatory only for those
             XYZv2 entities which also implement XYZv1."
::= { xyzMIBCompliances 1 }
```

According to this invocation, to claim alignment with the compliance statement named

```
{ xyzMIBCompliances 1 }
```

a system must implement the XYZv2-MIB's xyzSystemGroup, xyzStatsGroup, xyzTrapGroup, and xyzSetGroup object conformance groups, as well as the xyzBasicNotificationsGroup notifications group. Furthermore, if the XYZv2 entity also implements XYZv1, then it must also support the XYZv1Group group, if compliance is to be claimed.

6. Mapping of the AGENT-CAPABILITIES macro

The AGENT-CAPABILITIES macro is used to convey a set of capabilities present in a SNMPv2 entity acting in an agent role. It should be noted that the expansion of the AGENT-CAPABILITIES macro is something which conceptually happens during implementation and not during run-time.

When a MIB module is written, it is divided into units of conformance termed groups. If a SNMPv2 entity acting in an agent role claims to implement a group, then it must implement each and every object within that group. Of course, for whatever reason, a SNMPv2 entity might implement only a subset of the groups within a MIB module. In addition, the definition of some MIB objects leave some aspects of

the definition to the discretion of an implementor.

Practical experience has demonstrated a need for concisely describing the capabilities of an agent with respect to one or more MIB modules. The AGENT-CAPABILITIES macro allows an agent implementor to describe the precise level of support which an agent claims in regards to a MIB group, and to bind that description to the value of an instance of sysORID [3]. In particular, some objects may have restricted or augmented syntax or access-levels.

If the AGENT-CAPABILITIES invocation is given to a management-station implementor, then that implementor can build management applications which optimize themselves when communicating with a particular agent. For example, the management-station can maintain a database of these invocations. When a management-station interacts with an agent, it retrieves from the agent the values of all instances of sysORID [3]. Based on this, it consults the database to locate each entry matching one of the retrieved values of sysORID. Using the located entries, the management application can now optimize its behavior accordingly.

Note that the AGENT-CAPABILITIES macro specifies refinements or variations with respect to OBJECT-TYPE and NOTIFICATION-TYPE macros in MIB modules, NOT with respect to MODULE-COMPLIANCE macros in compliance statements.

6.1. Mapping of the PRODUCT-RELEASE clause

The PRODUCT-RELEASE clause, which must be present, contains a textual description of the product release which includes this set of capabilities.

6.2. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current ("current") or historic ("obsolete").

6.3. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual description of this set of capabilities.

6.4. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a capability statement defined in some other information module.

6.5. Mapping of the SUPPORTS clause

The SUPPORTS clause, which need not be present, is repeatedly used to name each MIB module for which the agent claims a complete or partial implementation. Each MIB module is named by its module name, and optionally, by its associated OBJECT IDENTIFIER as well.

6.5.1. Mapping of the INCLUDES clause

The INCLUDES clause, which must be present for each use of the SUPPORTS clause, is used to name each MIB group associated with the SUPPORTS clause, which the agent claims to implement.

6.5.2. Mapping of the VARIATION clause

The VARIATION clause, which need not be present, is repeatedly used to name each object or notification which the agent implements in some variant or refined fashion with respect to the correspondent invocation of the OBJECT-TYPE or NOTIFICATION-TYPE macro.

Note that the variation concept is meant for generic implementation restrictions, e.g., if the variation for an object depends on the values of other objects, then this should be noted in the appropriate DESCRIPTION clause.

By definition, each object specified in a VARIATION clause follows a SUPPORTS clause which names the information module in which that object is defined. Therefore, the use of an IMPORTS statement, to specify from where such objects are imported, is redundant and is not required in an information module.

6.5.2.1. Mapping of the SYNTAX clause

The SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent VARIATION clause. Note that if this clause and a WRITE-SYNTAX clause are both present, then this clause only applies when instances of the object named in the correspondent VARIATION clause are read.

Consult Section 9 of [2] for more information on refined syntax.

6.5.2.2. Mapping of the WRITE-SYNTAX clause

The WRITE-SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent VARIATION clause when instances of that object are written.

Consult Section 9 of [2] for more information on refined syntax.

6.5.2.3. Mapping of the ACCESS clause

The ACCESS clause, which need not be present, is used to indicate the agent provides less than the maximal level of access to the object or notification named in the correspondent VARIATION clause.

The only value applicable to notifications is "not-implemented".

The value "not-implemented" indicates the agent does not implement the object or notification, and in the ordering of possible values is equivalent to "not-accessible".

The value "write-only" is provided solely for backward compatibility, and shall not be used for newly-defined object types. In the ordering of possible values, "write-only" is less than "not-accessible".

6.5.2.4. Mapping of the CREATION-REQUIRES clause

The CREATION-REQUIRES clause, which need not be present, is used to name the columnar objects of a conceptual row to which values must be explicitly assigned, by a management protocol set operation, before the agent will allow the instance of the status column of that row to be set to ‘active’. (Consult the definition of RowStatus [5].)

If the conceptual row does not have a status column (i.e., the objects corresponding to the conceptual table were defined using the mechanisms in [6,7]), then the CREATION-REQUIRES clause, which need not be present, is used to name the columnar objects of a conceptual row to which values must be explicitly assigned, by a management protocol set operation, before the agent will create new instances of objects in that row.

This clause must not present unless the object named in the correspondent VARIATION clause is a conceptual row, i.e., has a syntax which resolves to a SEQUENCE containing columnar objects. The objects named in the value of this clause usually will refer to columnar objects in that row. However, objects unrelated to the conceptual row may also be specified.

All objects which are named in the CREATION-REQUIRES clause for a conceptual row, and which are columnar objects of that row, must have an access level of "read-create".

6.5.2.5. Mapping of the DEFVAL clause

The DEFVAL clause, which need not be present, is used to provide a refined DEFVAL value for the object named in the correspondent VARIATION clause. The semantics of this value are identical to those of the OBJECT-TYPE macro's DEFVAL clause.

6.5.2.6. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present for each use of the VARIATION clause, contains a textual description of the variant or refined implementation of the object or notification.

6.6. Mapping of the AGENT-CAPABILITIES value

The value of an invocation of the AGENT-CAPABILITIES macro is an OBJECT IDENTIFIER, which names the value of sysORID [3] for which this capabilities statement is valid.

6.7. Usage Example

Consider how a capabilities statement for an agent might be described:

```
exampleAgent AGENT-CAPABILITIES
    PRODUCT-RELEASE      "ACME Agent release 1.1 for 4BSD"
    STATUS               current
    DESCRIPTION          "ACME agent for 4BSD"

    SUPPORTS             SNMPv2-MIB
        INCLUDES         { systemGroup, snmpGroup, snmpSetGroup,
                           snmpBasicNotificationsGroup }

        VARIATION        coldStart
            DESCRIPTION  "A coldStart trap is generated on all
                         reboots."

    SUPPORTS             IF-MIB
        INCLUDES         { ifGeneralGroup, ifPacketGroup }

        VARIATION        ifAdminStatus
            SYNTAX       INTEGER { up(1), down(2) }
            DESCRIPTION  "Unable to set test mode on 4BSD"

        VARIATION        ifOperStatus
            SYNTAX       INTEGER { up(1), down(2) }
            DESCRIPTION  "Information limited on 4BSD"
```

```
    SUPPORTS             IP-MIB
        INCLUDES         { ipGroup, icmpGroup }

        VARIATION        ipDefaultTTL
            SYNTAX       INTEGER (255..255)
            DESCRIPTION  "Hard-wired on 4BSD"

        VARIATION        ipInAddrErrors
            ACCESS       not-implemented
            DESCRIPTION  "Information not available on 4BSD"

        VARIATION        ipNetToMediaEntry
            CREATION-REQUIRES { ipNetToMediaPhysAddress }
            DESCRIPTION  "Address mappings on 4BSD require
                          both protocol and media addresses"

    SUPPORTS             TCP-MIB
        INCLUDES         { tcpGroup }
        VARIATION        tcpConnState
            ACCESS       read-only
            DESCRIPTION  "Unable to set this on 4BSD"

    SUPPORTS             UDP-MIB
        INCLUDES         { udpGroup }

    SUPPORTS             EVAL-MIB
        INCLUDES         { functionsGroup, expressionsGroup }
        VARIATION        exprEntry
            CREATION-REQUIRES { evalString }
            DESCRIPTION "Conceptual row creation supported"

    ::= { acmeAgents 1 }
```

According to this invocation, an agent with a sysORID value of

```
  { acmeAgents 1 }
```

supports six MIB modules.

From SNMPv2-MIB, five conformance groups are supported.

From IF-MIB, the ifGeneralGroup and ifPacketGroup groups are supported. However, the objects ifAdminStatus and ifOperStatus have a restricted syntax.

From IP-MIB, all objects in the ipGroup and icmpGroup are supported except ipInAddrErrors, while ipDefaultTTL has a restricted range, and when creating a new instance in the ipNetToMediaTable, the set-request must create an instance of atPhysAddress.

From TCP-MIB, the tcpGroup is supported except that tcpConnState is available only for reading.

From UDP-MIB, the udpGroup is fully supported.

From the EVAL-MIB, all the objects contained in the functionsGroup and expressionsGroup conformance groups are supported, without variation. In addition, creation of new instances in the expr table is supported.

7. Extending an Information Module

As experience is gained with a published information module, it may be desirable to revise that information module.

Section 10 of [2] defines the rules for extending an information module. The remainder of this section defines how conformance groups, compliance statements, and capabilities statements may be extended.

7.1. Conformance Groups

If any non-editorial change is made to any clause of an object group then the OBJECT IDENTIFIER value associated with that object group must also be changed, along with its associated descriptor.

7.2. Compliance Definitions

If any non-editorial change is made to any clause of a compliance definition, then the OBJECT IDENTIFIER value associated with that compliance definition must also be changed, along with its associated descriptor.

7.3. Capabilities Definitions

If any non-editorial change is made to any clause of a capabilities definition, then the OBJECT IDENTIFIER value associated with that capabilities definition must also be changed, along with its associated descriptor.

8. Security Considerations

Security issues are not discussed in this memo.

9. Editor's Address

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

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

10. Acknowledgements

This document is the result of significant work by the four major contributors:

Jeffrey D. Case (SNMP Research, case@snmp.com)
Keith McCloghrie (Cisco Systems, kzm@cisco.com)
Marshall T. Rose (Dover Beach Consulting, mrose@dbc.mtview.ca.us)
Steven Waldbusser (International Network Services, stevew@uni.ins.com)

In addition, the contributions of the SNMPv2 Working Group are acknowledged. In particular, a special thanks is extended for the contributions of:

Alexander I. Alten (Novell)
Dave Arneson (Cabletron)
Uri Blumenthal (IBM)
Doug Book (Chipcom)
Kim Curran (Bell-Northern Research)
Jim Galvin (Trusted Information Systems)
Maria Greene (Ascom Timeplex)
Iain Hanson (Digital)
Dave Harrington (Cabletron)
Nguyen Hien (IBM)
Jeff Johnson (Cisco Systems)
Michael Kornegay (Object Quest)
Deirdre Kostick (AT&T Bell Labs)
David Levi (SNMP Research)
Daniel Mahoney (Cabletron)
Bob Natale (ACE*COMM)
Brian O'Keefe (Hewlett Packard)
Andrew Pearson (SNMP Research)
Dave Perkins (Peer Networks)

Randy Presuhn (Peer Networks)
Aleksey Romanov (Quality Quorum)
Shawn Routhier (Epilogue)
Jon Saperia (BGS Systems)
Bob Stewart (Cisco Systems, bstewart@cisco.com), chair
Kaj Tesink (Bellcore)
Glenn Waters (Bell-Northern Research)
Bert Wijnen (IBM)

11. References

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

[2] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[3] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907, January 1996.

[4] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[6] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", STD 16, RFC 1155, May 1990.

[7] Rose, M., and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.