Network Working Group   K. McCloghrie
Request For Comments: 1303   Hughes LAN Systems
                                                                 M. Rose
                                                  Dover Beach Consulting
                                                           February 1992

            A Convention for Describing SNMP-based Agents

Status of This Memo

Abstract

   This memo suggests a straight-forward approach towards describing
   SNMP-based agents.

Table of Contents

1. The Network Management Framework

   The Internet-standard Network Management Framework consists of
   three components. They are:

   RFC 1155 [1] which defines the SMI, the mechanisms used for
   describing and naming objects for the purpose of management.
   RFC 1212 [2] defines a more concise description mechanism,
   which is wholly consistent with the SMI.

   RFC 1213 [3] which defines MIB-II, the core set of managed
   objects for the Internet suite of protocols.

   RFC 1157 [4] which defines the SNMP, the protocol used for
   network access to managed objects.

   The Framework permits new objects to be defined for the
   purpose of experimentation and evaluation.

2. Objects

   Managed objects are accessed via a virtual information store,
   termed the Management Information Base or MIB. Within a given
   MIB module, objects are defined using RFC 1212's OBJECT-TYPE
   macro. At a minimum, each object has a name, a syntax, an
   access-level, and an implementation-status.

   The name is an object identifier, an administratively assigned
   name, which specifies an object type. The object type
   together with an object instance serves to uniquely identify a
   specific instantiation of the object. For human convenience,
   we often use a textual string, termed the OBJECT DESCRIPTOR,
   to also refer to the object type.

   The syntax of an object type defines the abstract data
   structure corresponding to that object type. The ASN.1[5]
   language is used for this purpose. However, RFC 1155
   purposely restricts the ASN.1 constructs which may be used.
   These restrictions are explicitly made for simplicity.

   The access-level of an object type defines whether it makes
   "protocol sense" to read and/or write the value of an instance
   of the object type. (This access-level is independent of any
   administrative authorization policy.)

   The implementation-status of an object type indicates whether
   the object is mandatory, optional, obsolete, or deprecated.

3. Describing Agents

   When a MIB module is written, it is divided into units of
   conformance termed groups. If an agent claims conformance to
   a group, then it must implement each and every object within
   that group. Of course, for whatever reason, an agent may
   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 regards to the
   MIB groups that it implements. This memo defines a new macro,
   MODULE-CONFORMANCE, which 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
   sysObjectID associated with the agent. In particular, some
   objects may have restricted or augmented syntax or access-
   levels.

   If the MODULE-CONFORMANCE 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 the agent's sysObjectID. Based on this,
   it consults the database. If an entry is found, then the
   management application can optimize its behavior accordingly.

   This binding to sysObjectId may not always suffice to define
   all MIB objects to which an agent can provide access. In
   particular, this situation occurs where the agent dynamically
   learns of the objects it supports, for example, an agent
   supporting SMUX peers via the SMUX protocol [6]. In these
   situations, additional MIB objects beyond sysObjectID must be
   used to name other invocations of the MODULE-CONFORMANCE macro
   to augment the description of MIB support provided by the
   agent. For example, if an agent's sysObjectID indicates that
   it supports the SMUX MIB, then each instance of smuxPidentity
   will indicate another MODULE-CONFORMANCE invocation which is
   dynamically being supported by the agent.

3.1. Definitions

```
   RFC-1303 DEFINITIONS ::= BEGIN

       IMPORTS
           ObjectName, ObjectSyntax
               FROM RFC1155-SMI
           DisplayString
               FROM RFC1213-MIB;

       MODULE-CONFORMANCE MACRO ::=
       BEGIN
           TYPE NOTATION ::=
                           "LAST-UPDATED"
                               value(update      UTCTime)
                           "PRODUCT-RELEASE"
                               value(release     DisplayString)
                           "DESCRIPTION"
                               value(description DisplayString)
                           ModulePart

           VALUE NOTATION ::=       -- agent's sysObjectID --
                           value(VALUE ObjectName)

           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(group OBJECT IDENTIFIER)

           VariationPart ::= Variations
                           | empty
```

```
          Variations ::=     Variation
                           | Variations Variation

          Variation ::=      "VARIATION" value(object ObjectName)
                             Syntax WriteSyntax Access
                             Creation DefaultValue
                             "DESCRIPTION"
                                 value(limitext DisplayString)

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

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

          Access ::=         "ACCESS" AccessValue
                           | empty

          AccessValue ::=   "read-only"
                          | "read-write"
                          | "write-only"
                          | "not-accessible"

          Creation ::=       "CREATION-REQUIRES" "{" Cells "}"

          Cells ::=          Cell
                           | Cells "," Cell

          Cell ::=           value(cell ObjectName)

          DefaultValue ::=   "DEFVAL"
                             "{" value (defval ObjectSyntax) "}"
                           | empty

       END

    END
```

3.2. Mapping of the MODULE-CONFORMANCE macro

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

3.2.1. Mapping of the LAST-UPDATED clause

   The LAST-UPDATED clause, which must be present, contains the date and
   time that this definition was last edited.

3.2.2. 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 agent.

3.2.3. Mapping of the DESCRIPTION clause

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

3.2.4. 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. (Note that
   only a few MIB modules have had OBJECT IDENTIFIERs assigned to them.)

3.2.4.1. 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
   SUPPORT clause, which the agent claims to implement.

3.2.4.2. Mapping of the VARIATION clause

   The VARIATION clause, which need not be present, is repeatedly used
   to name each MIB object which the agent implements in some variant or
   refined fashion.

3.2.4.2.1. 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 3.3 for more information on refined syntax.

3.2.4.2.2. 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 3.3 for more information on refined syntax.

3.2.4.2.3. Mapping of the ACCESS clause

   The ACCESS clause, which need not be present, is used to provide a
   refined ACCESS for the object named in the correspondent VARIATION
   clause.

3.2.4.2.4. 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 SNMP Set operation, before the agent will
   create new instances of objects in that row. This clause must not be
   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.

   The absence of this clause for a particular conceptual row indicates
   that the agent does not support the creation, via SNMP operations, of
   new object instances in that row.

3.2.4.2.5. 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 [2].

3.2.4.2.6. 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.

3.3. Refined Syntax

   The SYNTAX and WRITE-SYNTAX clauses allow an object's Syntax to be
   refined. However, not all refinements of syntax are appropriate. In
   particular,

    (1) the object's primitive or application type (as defined in
         the SMI [1]) must not be changed;

    (2) an object defined with an SMI type of OBJECT IDENTIFIER,
         IpAddress, Counter, or TimeTicks cannot be refined; and,

    (3) an object defined to have a specific set of values (e.g.,
         an INTEGER with named values) cannot have additional
         values defined for it.

3.4. Usage Example

    Consider how one might document the 4BSD/ISODE "Secure" SNMP
    agent:

    FourBSD-ISODE   DEFINITIONS ::= BEGIN

    IMPORTS
        MODULE-CONFORMANCE
            FROM RFC-1303
        -- everything --
            FROM RFCxxxx-MIB
        -- everything --
            FROM RFC1213-MIB
        -- everything --
            FROM UNIX-MIB
        -- everything --
            FROM EVAL-MIB;

    fourBSD-isode-support MODULE-CONFORMANCE
        LAST-UPDATED   "9201252354Z"
        PRODUCT-RELEASE   "ISODE 7.0 + 'Secure' SNMP
                             upgrade for SunOS 4.1"
        DESCRIPTION   "4BSD/ISODE 'Secure' SNMP"

        SUPPORTS   RFCxxxx-MIB -- SNMP Party MIB --
            INCLUDES   { partyPublic, partyPrivate }

        SUPPORTS   RFC1213-MIB
            INCLUDES   { system, interfaces, at, ip, icmp,
                              tcp, udp, snmp }

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

            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"

    VARIATION         ipDefaultTTL
        SYNTAX        INTEGER { maxttl(255) }
        DESCRIPTION "Hard-wired on 4BSD"

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

    VARIATION         ipInDiscards
        ACCESS        not-accessible
        DESCRIPTION "Information not available on 4BSD"

    VARIATION         ipRouteEntry
        CREATION-REQUIRES { ipRouteNextHop }
        DESCRIPTION "Routes on 4BSD require both
                     destination and next-hop"

    VARIATION         ipRouteType
        SYNTAX        INTEGER { direct(3), indirect(4) }
        WRITE-SYNTAX INTEGER { invalid(2), direct(3),
                            indirect(4) }
        DESCRIPTION "Information limited on 4BSD"

    VARIATION         ipRouteProto
        SYNTAX        INTEGER { other(1), icmp (4) }
        DESCRIPTION "Information limited on 4BSD"

    VARIATION         ipRouteAge
        ACCESS        not-accessible
        DESCRIPTION "Information not available on 4BSD"

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

    VARIATION         ipNetToMediaType
        SYNTAX        INTEGER { dynamic(3), static(4) }
        WRITE-SYNTAX INTEGER { invalid(2), dynamic(3),
                            static(4) }
        DESCRIPTION "Information limited on 4BSD"
```

```
            VARIATION        tcpConnState
                ACCESS       read-only
                DESCRIPTION "Unable to set this on 4BSD"

            VARIATION        tcpInErrs
                ACCESS       not-accessible
                DESCRIPTION "Information not available on 4BSD"

            VARIATION        tcpOutRsts
                ACCESS       not-accessible
                DESCRIPTION "Information not available on 4BSD"

        SUPPORTS             UNIX-MIB
            INCLUDES         { agents, mbuf, netstat }

        SUPPORTS             EVAL-MIB
            INCLUDES         { functions, expressions }

        ::= { fourBSD-isode 6 6 }

    END
```

According to this invocation, an agent with a sysObjectID of

```
    { fourBSD-isode 6 6 }
```

supports four MIB modules.

From the SNMP Party MIB, all the objects contained in the partyPublic and partyPrivate groups are supported, without variation.

From MIB-II, all groups except the egp group are supported. However, the objects

```
    ipInAddrErrors
    ipInDiscards
    ipRouteAge
    tcpInErrs
    tcpOutRsts
```

are not available, whilst the objects

```
    ifAdminStatus
    ifOperStatus
    ipDefaultTTL
    ipRouteType
    ipRouteProto
    ipNetToMediaType
```

have a restricted syntax, and the object

    tcpConnState

is available only for reading. Note that in the case of the objects

    ipRouteType
    ipNetToMediaType

the set of values which may be read is different than the set of
values which may be written. Finally, when creating a new row in the
atTable, the set-request must create an instance of atPhysAddress.
Similarly, when creating a new row in the ipRouteTable, the set-
request must create an instance of ipRouteNextHop. Similarly, when
creating a new row in the ipNetToMediaTable, the set-request must
create an instance of ipNetToMediaPhysAddress.

From the UNIX-MIB, all the objects contained in the agents, mbuf, and
netstat groups are supported, without variation.

From the EVAL-MIB, all the objects contained in the functions and
expressions groups are supported, without variation.

4. Acknowledgements

The authors gratefully acknowledge the comments of Mark van der Pol
of Hughes LAN Systems, and David T. Perkins of SynOptics
Communications.

5. References

[1] Rose M., and K. McCloghrie, "Structure and Identification of
     Management Information for TCP/IP-based internets", RFC 1155,
     Performance Systems International, Hughes LAN Systems, May 1990.

[2] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions",
     RFC 1212, Performance Systems International, Hughes LAN Systems,
     March 1991.

[3] McCloghrie K., and M. Rose, Editors, "Management Information
     Base forNetwork Management of TCP/IP-based internets", RFC 1213,
     Performance Systems International, March 1991.

[4] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple
     Network Management Protocol (SNMP), RFC 1157, SNMP Research,
     Performance Systems International, Performance Systems
     International, MIT Laboratory for Computer Science, May 1990.

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

   [6] Rose, M., "SNMP MUX Protocol and MIB", RFC 1227, Performance
        Systems International, May 1991.

6. Security Considerations

   Security issues are not discussed in this memo.

7. Authors' Addresses

   Keith McCloghrie
   Hughes LAN Systems
   1225 Charleston Road
   Mountain View, CA 94043

   Phone: (415) 966-7934
   EMail: kzm@hls.com

   Marshall T. Rose
   Dover Beach Consulting, Inc.
   420 Whisman Court
   Mountain View, CA 94043-2112

   Phone: (415) 968-1052
   EMail: mrose@dbc.mtview.ca.us