# Definitions of Managed Objects for Instance Reservation

September 16, 1996

<draft-ietf-resmib-instresmib-00.txt>

David Battle
SNMP Research, Inc.
battle@snmp.com

Ulrich Haebel
Siemens Nixdorf Informationssysteme AG
Ulrich.Haebel@mch.sni.de

## Status of this Memo

SNMP Research hereby covenants that it will not assert any claims in this draft, any continuations, divisions, or continuations-in-part of the foregoing against any party that makes, uses, sells, imports, or offers for sale, an implementation of an IETF standards-track MIB module that includes SNMP Research's

contributed instance reservation technology, or any derivative of that contribution, provided that the MIB module is employed to implement the Internet-standard network management framework. Such grant of rights is limited in that it does not include a right to incorporate such SNMP Research instance reservation technology into proprietary MIB modules.

This grant of rights will permanently terminate with respect to any party (and to any subsidiary of a party) that asserts a patent it owns or controls, either directly or indirectly, against SNMP Research or any of its subsidiaries for implementation of, or operation of any system utilizing, instance reservation technology.  The termination of rights will occur as of the date the patent is asserted against SNMP Research.

## 1.  Abstract

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes a basic set of managed objects for instance reservation in other MIB tables.

This memo does not specify a standard for the Internet community.

## 2.  The SNMPv2 Network Management Framework

The SNMPv2 Network Management Framework consists of four major components.  They are:

- STD 17, RFC 1213 [1] defines MIB-II, the core set of managed objects for the Internet suite of protocols.

- RFC 1902 Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)

- RFC 1903 Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)

- RFC 1904 Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)

- RFC 1905 Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)

- o RFC 1906 Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)

- o RFC 1907 Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)

- o RFC 1908 Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework

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

### 2.1. Object Definitions

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) defined in the SMI[2]. In particular, each object type is named by an OBJECT IDENTIFIER, an administratively assigned name. 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 refer to the object type.

## 3. Overview

Traditionally SNMP agents have been "monolithic" in nature, meaning that a single linked module was used to represent the entire MIB for a particular entity.  In recent years several "extensible agent" technologies have come to exist which split the MIB among several distinct linked modules, typically called subagents.  The existance of these subagent modules has created the need for a mechanism by when different subagents may coordinate instances in mib tables in such a way that each subagent can perform its function without collisions occuring when more than one subagent tries to present the same instance of a particualar managed object.

This draft is concerned primarily with providing a simple mechanism by which subagents can "reserve" instances in certain tables for their own use and find out about reservations made by other subagents.  This approach was taken to insure ease and speed of implementation, while allowing room for future growth.

## 4. The Structure of the MIB

The instance reservation mib consists of three tables (instanceTable, allocInstTable, and maxInstTable). The tables are used together to allow subagents to reserve particular instances (rows) in other SNMP tables. All of the objects in the mib are read only or not accessible. Subagents reserve instances by asking questions of the form "which instance should I use" or "may I reserve this instance" (that is, via GET requests) rather than making statements such as "I want to reserve this instance" (using SET requests). These get requests do the actual work of making the reservation. Making reservations using this method has the following advantages:

- Reservations may be done in a single exchange, since the return value from the get may be used to return reservation information. Since sets return no new information other than success or failure, a set and a subsequent get would be required to reserve an instance, then return instance information.

- Problems with locking and simultaneous access associated with sets in multithreaded agents are avoided.

The table instanceTable reflects the state of the instance reservation table. It is used to request reservation of a specific row of a given table or get information about what rows are reserved. instanceTable is indexed by two Object Identifiers, the OID of the table in which an instance is to be reserved, and the OID which is formed by concatenating the indices for the row being reserved. instanceUsed and instanceNotUsed are the only two accessible objects. A getexact or getnext on the instanceUsed object return an indication of whether a particular instance is in use or has been used in the past (instances which have never been used do not show up in the table at all). A getexact on an instanceUsed object which is not currently in use will reserve that instance, and return either a onceUsed or neverUsed indication. Future queries about that instance will return the "reserved" state. Thus, if a subagent knows which instance it wants, it does a getexact on the instanceTable supplying the OIDs of the table and the desired instance. If the return value indicates that the instance is not in use, the subagent may immediately start using it. A getnext will query whether an instance is in use without reserving that instance. The instanceNotUsed object (indexed by the same OIDs as instanceUsed)

is used to free a row reservation using a getexact operation in the same fashion (as instanceUsed).  Getnext operations on instanceNotUsed return the same information as getnext operations on instanceUsed.

The second table (allocInstTable) is used when a subagent does not know which instance in a given table it wants to allocate. allocInstTable is indexed by algorithm, OID of the table from which an instance is to be allocated, and instance type to allocate.  A getexact operation on allocInstInstance causes an instance allocation and reservation to occur  The algorithm specified, either firstNeverUsed or firstNotCurrentlyUsed, specifies to the entity implementing the mib which type of instance to reserve.  The instance type is formed by building an oid with the first field specifying the number of indices, and each subsequent field spcifying the enumerated type of the indices for the row to be allocated.  Getnext operations on this table will always show it empty.

The third table (maxInstTable) is used by subagents which want to use instanceTable (the first table in this mib) to directly request an instance, but which first need to inquire about the largest instance already in use.  This table is indexed by algorithm and the OID for the table for which an instance is requested.  Algorithm is specified as largestEverUsed or largestCurrentlyUsed. Getexact or getnext operations on maxInstInstance return the lexicographically largest instance currently (or ever) used in the indicated table.  Unlike the previous two tables, getexact and getnext operations have no side effects.

## 5.  Definitions

```
INSTANCEREP-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    ObjectName, Integer32, Counter32
        FROM SNMPv2-SMI;


instanceRep MODULE-IDENTITY
```

```
    LAST-UPDATED "9605071655Z"
    ORGANIZATION "SNMP Research, Inc."
    CONTACT-INFO "David Battle
          Postal: SNMP Research, Inc.
                  3001 Kimberlin Heights Road
                  Knoxville, TN  37920
          Tel:    +1 423 573 1434
          E-Mail: battle@snmp.com"
    DESCRIPTION  "This module describes a mib for reserving instances in
                  various mib tables for use by extensible agent systems."
        ::= { enterprises 99 12 17 }



-- The instance Group

instance OBJECT IDENTIFIER ::= { instanceRep 1 }

-- used as a prefix for OIDs describing a particular instance
instanceInstanceBase OBJECT IDENTIFIER ::= { instanceRep 2 }

-- used as a prefix for OIDs describing instance types
instanceTypeBase OBJECT IDENTIFIER ::= { instanceRep 3 }

instanceTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF InstanceEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "A table of reserved instances containing presently and
                 historically reserved instances.  Different contexts may be
                 used by different subagents in allocating instances so that,
                 for example, only a subagent which allocates an instance may
                 release it."
::= { instance 1 }

instanceEntry OBJECT-TYPE
    SYNTAX     InstanceEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "An entry describing a particular reserved (or previously
                 reserved) instance containing objects which describe the
                 instance's current state."
    INDEX { instanceObject, IMPLIED instanceInstance }
::= { instanceTable 1 }
```

```
  InstanceEntry ::= SEQUENCE {
      instanceObject       OBJECT IDENTIFIER,
      instanceInstance     OBJECT IDENTIFIER,
      instanceUsed         INTEGER,
      instanceNotUsed      INTEGER
  }

  instanceObject OBJECT-TYPE
      SYNTAX     OBJECT IDENTIFIER
      MAX-ACCESS not-accessible
      STATUS     current
      DESCRIPTION "The oid of the table to be associated with the allocated
                   instance.  This should be the oid of the SEQUENCE, not
                   the oid of any particular object."
  ::= { instanceEntry 1 }

  instanceInstance OBJECT-TYPE
      SYNTAX     OBJECT IDENTIFIER
      MAX-ACCESS not-accessible
      STATUS     current
      DESCRIPTION "The instance to be allocated or released encoded as a oid.
                   The instance is prefixed with the instanceInstanceBase oid
                   to ensure that it is a legal oid."
  ::= { instanceEntry 2 }

  instanceUsed OBJECT-TYPE
      SYNTAX     INTEGER
                    {
                        neverUsed(1),
                        onceUsed(2),
                        reserved(3)
                    }
      MAX-ACCESS read-only
      STATUS     current
      DESCRIPTION "The state of the allocated instance as indicated by the
                   state table below.  A get-exact operation on this
                   object causes the indicated state transitions.  The state
                   returned is indicated as a separate column in the state
                   table."
  ::= { instanceEntry 3 }

  --
  -- Operation:        Get-Exact on instanceUsed    Get-Exact on
instanceNotUsed
  --
  -- Current State  State Returned  State Entered  State Returned  State
Entered
```

```
  -- neverUsed      neverUsed       reserved       NULL            neverUsed
  -- onceUsed       onceUsed        reserved       onceUsed        onceUsed
  -- reserved       reserved        reserved       onceUsed        onceUsed
  --
  -- The idea is that subagents will use a get-exact on a particular instance
of
  -- instanceUsed to allocate a row and a get-exact on a particular instance of
  -- instanceNotUsed in order to free the row.  The State Returned column
  -- indicates what should be returned for get-exact.  For get-next the
  -- current state should be returned and should not be changed.  The subagent
  -- can tell whether the instance we succesfully allocated by looking at
  -- the state returned from the get-exact; neverUsed or onceUsed means
success.
  -- reserved means failure (ie it was already reserved).
  --
  -- NULL is used above to indicate that in this situation the agent should
  -- behave as though the instance does not exist.
  --
  -- Note also that the instance reservation mib may be implemented in such
  -- a way that only get-exacts in a certain "context" will change the
  -- state variables instanceUsed and instanceNotUsed.  instanceUsed and
  -- instanceNotUsed are shadow objects which always match each other in
  -- their internal value.
  --


  instanceNotUsed OBJECT-TYPE
      SYNTAX     INTEGER
                    {
                        neverUsed(1),
                        onceUsed(2),
                        reserved(3)
                    }
      MAX-ACCESS read-only
      STATUS     current
      DESCRIPTION "The state of the allocated instance as indicated by the
                   state table above.  A get-exact operation on this
                   object causes the indicated state transitions.  The state
                   returned is indicated as a separate column in the state
                   table."
  ::= { instanceEntry 4 }

  allocInstTable OBJECT-TYPE
      SYNTAX     SEQUENCE OF AllocInstEntry
      MAX-ACCESS not-accessible
      STATUS     current
      DESCRIPTION "An ephemeral list used for allocation of instances.  Entries
```

```
                 persist only long enough to return a value, then disappear.
                 Different contexts may be used by different subagents in
                 allocating instances so that, for example, only a subagent
                 which allocates an instance may release it."
::= { instance 2 }

allocInstEntry OBJECT-TYPE
    SYNTAX     AllocInstEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "An entry used for allocating instances containing objects
                 need for the description of the variety of instance to be
                 allocated."
    INDEX { allocInstAlgorithm, allocInstObject, IMPLIED allocInstType }
::= { allocInstTable 1 }

AllocInstEntry ::= SEQUENCE {
    allocInstAlgorithm   INTEGER,
    allocInstObject      OBJECT IDENTIFIER,
    allocInstType        OBJECT IDENTIFIER,
    allocInstInstance    OBJECT IDENTIFIER
}

allocInstAlgorithm OBJECT-TYPE
    SYNTAX     INTEGER {
                  firstNeverUsed(1),
                  firstNotCurrentlyUsed(2)
                }
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "The algorithm to be used in allocating the instance."
::= { allocInstEntry 1 }

allocInstObject OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "The oid of the table to be associated with the allocated
                 instance.  This should be the oid of the SEQUENCE, not
                 the oid of any particular object."
::= { allocInstEntry 2 }

allocInstType OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS not-accessible
```

```
      STATUS     current
      DESCRIPTION "The type of instance to be allocated.  The initial portion
                   of this oid consists of instanceTypeBase.  The subsequent
                   sub-identifiers have the following meaning:
                   Sub-Id Value          Meaning
                     1      n    total number of indices in the instance
                     2     type  the type of the first index as indicated below
                     x     type  the type of the (x-1)th index as indicated
below
                     n+1   type  the type of the last index as indicated below

                   Value   Type of Index
                     1       INTEGER
                     2       ..."
  ::= { allocInstEntry 3 }

  allocInstInstance OBJECT-TYPE
      SYNTAX     OBJECT IDENTIFIER
      MAX-ACCESS read-only
      STATUS     current
      DESCRIPTION "A get-exact request on this object returns an instance
                   encoded as an oid which was allocated according to the
                   algorithm specified in the allocInstAlgorithm index.  The
                   allocated instance will be of the type indicated by the
                   allocInstType index and will be associated with the table
                   indicated by allocInstObject.  When an instance is allocated
                   using this table it causes a cooresponding entry to appear
                   in the instanceTable."
  ::= { allocInstEntry 4 }


  maxInstTable OBJECT-TYPE
      SYNTAX     SEQUENCE OF AllocInstEntry
      MAX-ACCESS not-accessible
      STATUS     current
      DESCRIPTION "A table of the maximum instances currently reserved."
  ::= { instance 3 }

  maxInstEntry OBJECT-TYPE
      SYNTAX     MaxInstEntry
      MAX-ACCESS not-accessible
      STATUS     current
      DESCRIPTION "An entry describing the maximum instance for a particular
                   table."
      INDEX { maxInstAlgorithm, maxInstObject }
  ::= { maxInstTable 1 }
```

```
MaxInstEntry ::= SEQUENCE {
    maxInstAlgorithm   INTEGER,
    maxInstObject      OBJECT IDENTIFIER,
    maxInstInstance    OBJECT IDENTIFIER
}

maxInstAlgorithm OBJECT-TYPE
    SYNTAX     INTEGER {
                  largestEverUsed(1),
                  largestCurrentlyUsed(2)
                }
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "The algorithm to be used in computing the maximum instance."
::= { maxInstEntry 1 }

maxInstObject OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION "The oid of the table for which the maximum instance should
                 be computed.  This should be the oid of the SEQUENCE, not
                 the oid of any particular object."
::= { maxInstEntry 2 }

maxInstInstance OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION "The largest instance reserved in the indicated table
                 computed according to the indicated algorithm.  Items
                 in the onceUsed state are not considered for the
                 largestCurrentlyUsed algorithm, while they are considered
                 for the largestEverUsed algorithm.  The value of this object
                 will be identical to the value of of the appropriate
                 instance of the instanceInstance object."

::= { maxInstEntry 4 }

END
```

## 6. References

[1] McCloghrie, K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

[2] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Structure of Management Information for Version 2 of the Simple Network Mana" gement Protocol (SNMPv2)" , RFC 1902, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mello n University, January 1996.

## 7. Security Considerations

Security issues are not discussed in this memo.

## 8. Authors' Addresses

David Battle
SNMP Research, Inc.
3001 Kimberlin Heights Rd.
Knoxville, TN  37920-9716
US

Phone: +1 423 573 1434
Email: battle@snmp.com

Ulrich Haebel
Siemens Nixdorf Informationssysteme AG
Otto-Hahn-Ring 6
81739 Muenchen
Germany

Phone: +49 89 636 42141
Email: Ulrich.Haebel@mch.sni.de

Table of Contents

9