Network Working Group
Request for Comments: 3231
Obsoletes: 2591
Category: Standards Track

D. Levi
Nortel Networks
J. Schoenwaelder
TU Braunschweig

January 2002

Definitions of Managed Objects for Scheduling Management Operations

Status of this Memo

Copyright Notice

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes a set of managed objects that are used to schedule management operations periodically or at specified dates and times.

This document obsoletes RFC 2591.

Table of Contents

1 Introduction 2
2 The SNMP Management Framework 2
3 Overview 3
3.1 Periodic Schedules 4
3.2 Calendar Schedules 4
3.3 One-shot Schedules 5
3.4 Time Transitions 5
3.5 Actions 5
4 Definitions 6
5 Usage Examples 20
5.1 Starting a script to ping devices every 20 minutes 20
5.2 Starting a script at the next Friday the 13th 21
5.3 Turning an interface off during weekends 22

6 Security Considerations 23
7 Intellectual Property 25
8 Changes from RFC 2591 25
9 Acknowledgments 26
10 References 26
11 Editors' Addresses 28
12 Full Copyright Statement 29

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes a set of managed objects that are used to schedule management operations periodically or at specified dates and times.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. The SNMP Management Framework

The SNMP Management Framework presently consists of five major components:

- An overall architecture, described in RFC 2571 [RFC2571].

- Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and described in STD 16, RFC 1155 [RFC1155], STD 16, RFC 1212 [RFC1212] and RFC 1215 [RFC1215]. The second version, called SMIv2, is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

- Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [RFC1157]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [RFC1901] and RFC 1906 [RFC1906]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [RFC1906], RFC 2572 [RFC2572] and RFC 2574 [RFC2574].

- o Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [RFC1157]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [RFC1905].

- o A set of fundamental applications described in RFC 2573 [RFC2573] and the view-based access control mechanism described in RFC 2575 [RFC2575].

A more detailed introduction to the current SNMP Management Framework can be found in RFC 2570 [RFC2570].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIv2. A MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

3. Overview

The MIB defined in this memo provides the scheduling of actions periodically or at specified dates and times. The actions can be used to realize on-duty / off-duty schedules or to trigger management functions in a distributed management application.

Schedules can be enabled or disabled by modifying a control object. This allows for pre-configured schedules which are activated or deactivated by some other management functions.

The term ‘scheduler’ is used throughout this memo to refer to the entity which implements the scheduling MIB and which invokes the actions at the specified points in time.

3.1. Periodic Schedules

Periodic schedules are based on fixed time periods between the initiation of scheduled actions. Periodic schedules are defined by specifying the number of seconds between two initiations. The time needed to complete the action is usually not known by the scheduler and does therefore not influence the next scheduling point.

Implementations must guarantee that action invocations will not occur before their next scheduled time. However, implementations may be forced to delay invocations in the face of local constraints (e.g., a heavy load on higher-priority tasks). An accumulation of such delays would result in a drift of the scheduling interval with respect to time, and should be avoided.

Scheduled actions collecting statistical data should retrieve time stamps from the data source and not rely on the accuracy of the periodic scheduler in order to obtain accurate statistics.

3.2. Calendar Schedules

Calendar schedules trigger scheduled actions at specified days of the week and days of the month. Calendar schedules are therefore aware of the notion of months, days, weekdays, hours and minutes.

It is possible to specify multiple values for each calendar item. This provides a mechanism for defining complex schedules. For example, a schedule could be defined which triggers an action every 15 minutes on a given weekday.

Months, days and weekdays are specified using the objects schedMonth, schedDay and schedWeekDay of type BITS. Setting multiple bits to one in these objects causes an OR operation. For example, setting the bits monday(1) and friday(5) in schedWeekDay restricts the schedule to Mondays and Fridays.

The bit fields for schedMonth, schedDay and schedWeekDay are combined using an AND operation. For example, setting the bits june(5) and july(6) in schedMonth and combining it with the bits monday(1) and friday(5) set in schedWeekDay will result in a schedule which is restricted to every Monday and Friday in the months June and July. Wildcarding of calendar items is achieved by setting all bits to one.

It is possible to define calendar schedules that will never trigger an action. For example, one can define a calendar schedule which should trigger an action on February 31st. Schedules like this will simply be ignored by the scheduler.

Finally, calendar schedules are always expressed in local time. A scalar, schedLocalTime, is provided so that a manager can retrieve the notion of local time and the offset to GMT time.

3.3. One-shot Schedules

One-shot Schedules are similar to calendar schedules. The difference between a calendar schedule and a one-shot schedule is that a one-shot schedule will automatically disable itself once an action has been invoked.

3.4. Time Transitions

A time transition occurs when the Schedule MIB's notion of time (as reported by schedLocalTime) is changed so that time continuity is lost. Time transitions may be caused by daylight savings times or administrative changes of the system's notion of time.

There are two possible situations when a time transition occurs.

First, time may be set backwards, in which case particular times will appear to occur twice. These are called 'ambiguous times'. Second, time may be set forwards, in which case particular times will not occur. These are called 'nonexistent times'.

When an action is configured in the Schedule MIB to occur at an ambiguous time, the action will be invoked at all occurrences of the ambiguous time. For example, if an action is scheduled to occur at 2:10 am, and a time transition occurs at 3:00 am which sets the clock back to 2:00 am, the action will be invoked twice.

When an action is configured in the Schedule MIB to occur at a nonexistent time, the action will not be invoked at all. For example, if an action is scheduled to occur at 2:10 am, and a time transition occurs at 2:00 am which sets the clock to 3:00 am, the action will not be invoked.

3.5. Actions

Scheduled actions are modeled by SNMP set operations on local MIB variables. Scheduled actions described in this MIB are further restricted to objects of type INTEGER. This restriction does not limit the usefulness of the MIB. Simple schedules such as on-duty / off-duty schedules for resources that have a status MIB object (e.g. ifAdminStatus) are possible.

More complex actions can be realized by triggering a management script which is responsible for performing complex state transitions. A management script can also be used to perform SNMP set operations on remote SNMP engines.

4. Definitions

```
DISMAN-SCHEDULE-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Integer32, Unsigned32, Counter32, mib-2, zeroDotZero
        FROM SNMPv2-SMI

    TEXTUAL-CONVENTION,
    DateAndTime, RowStatus, StorageType, VariablePointer
        FROM SNMPv2-TC

    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF

    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB;

schedMIB MODULE-IDENTITY
    LAST-UPDATED "200201070000Z"
    ORGANIZATION "IETF Distributed Management Working Group"
    CONTACT-INFO
        "WG EMail:  disman@dorothy.bmc.com
         Subscribe: disman-request@dorothy.bmc.com

         Chair:     Randy Presuhn
                    BMC Software, Inc.
         Postal:    Office 1-3141
                    2141 North First Street
                    San Jose,  California 95131
                    USA
         EMail:     rpresuhn@bmc.com
         Phone:     +1 408 546-1006

         Editor:    David B. Levi
                    Nortel Networks
         Postal:    4401 Great America Parkway
                    Santa Clara, CA 95052-8185
                    USA
         EMail:     dlevi@nortelnetworks.com
         Phone:     +1 865 686 0432
```

```
            Editor:    Juergen Schoenwaelder
                       TU Braunschweig
            Postal:    Bueltenweg 74/75
                       38106 Braunschweig
                       Germany
            EMail:     schoenw@ibr.cs.tu-bs.de
            Phone:     +49 531 391-3283"
       DESCRIPTION
           "This MIB module defines a MIB which provides mechanisms to
            schedule SNMP set operations periodically or at specific
            points in time."
       REVISION    "200201070000Z"
       DESCRIPTION
           "Revised version, published as RFC 3231.

            This revision introduces a new object type called
            schedTriggers.  Created new conformance and compliance
            statements that take care of the new schedTriggers object.

            Several clarifications have been added to remove ambiguities
            that were discovered and reported by implementors."
       REVISION    "199811171800Z"
       DESCRIPTION
           "Initial version, published as RFC 2591."
       ::= { mib-2 63 }

   --
   -- The various groups defined within this MIB definition:
   --

   schedObjects       OBJECT IDENTIFIER ::= { schedMIB 1 }
   schedNotifications OBJECT IDENTIFIER ::= { schedMIB 2 }
   schedConformance   OBJECT IDENTIFIER ::= { schedMIB 3 }

   --
   -- Textual Conventions:
   --

   SnmpPduErrorStatus ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
           "This TC enumerates the SNMPv1 and SNMPv2 PDU error status
            codes as defined in RFC 1157 and RFC 1905.  It also adds a
            pseudo error status code 'noResponse' which indicates a
            timeout condition."
       SYNTAX      INTEGER {
                       noResponse(-1),
                       noError(0),
```

```
                          tooBig(1),
                          noSuchName(2),
                          badValue(3),
                          readOnly(4),
                          genErr(5),
                          noAccess(6),
                          wrongType(7),
                          wrongLength(8),
                          wrongEncoding(9),
                          wrongValue(10),
                          noCreation(11),
                          inconsistentValue(12),
                          resourceUnavailable(13),
                          commitFailed(14),
                          undoFailed(15),
                          authorizationError(16),
                          notWritable(17),
                          inconsistentName(18)
                      }

   --
   -- Some scalars which provide information about the local time zone.
   --

   schedLocalTime OBJECT-TYPE
       SYNTAX      DateAndTime (SIZE (11))
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The local time used by the scheduler.  Schedules which
            refer to calendar time will use the local time indicated
            by this object.  An implementation MUST return all 11 bytes
            of the DateAndTime textual-convention so that a manager
            may retrieve the offset from GMT time."
       ::= { schedObjects 1 }

   --
   -- The schedule table which controls the scheduler.
   --

   schedTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF SchedEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table defines scheduled actions triggered by
            SNMP set operations."
       ::= { schedObjects 2 }
```

```
   schedEntry OBJECT-TYPE
       SYNTAX      SchedEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "An entry describing a particular scheduled action.

            Unless noted otherwise, writable objects of this row
            can be modified independent of the current value of
            schedRowStatus, schedAdminStatus and schedOperStatus.
            In particular, it is legal to modify schedInterval
            and the objects in the schedCalendarGroup when
            schedRowStatus is active and schedAdminStatus and
            schedOperStatus are both enabled."
       INDEX { schedOwner, schedName }
       ::= { schedTable 1 }

   SchedEntry ::= SEQUENCE {
       schedOwner          SnmpAdminString,
       schedName           SnmpAdminString,
       schedDescr          SnmpAdminString,
       schedInterval       Unsigned32,
       schedWeekDay        BITS,
       schedMonth          BITS,
       schedDay            BITS,
       schedHour           BITS,
       schedMinute         BITS,
       schedContextName    SnmpAdminString,
       schedVariable       VariablePointer,
       schedValue          Integer32,
       schedType           INTEGER,
       schedAdminStatus    INTEGER,
       schedOperStatus     INTEGER,
       schedFailures       Counter32,
       schedLastFailure    SnmpPduErrorStatus,
       schedLastFailed     DateAndTime,
       schedStorageType    StorageType,
       schedRowStatus      RowStatus,
       schedTriggers       Counter32
   }

   schedOwner OBJECT-TYPE
       SYNTAX      SnmpAdminString (SIZE(0..32))
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "The owner of this scheduling entry.  The exact semantics of
            this string are subject to the security policy defined by
```

```
            the security administrator."
        ::= { schedEntry 1 }

    schedName OBJECT-TYPE
        SYNTAX      SnmpAdminString (SIZE(1..32))
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The locally-unique, administratively assigned name for this
             scheduling entry.  This object allows a schedOwner to have
             multiple entries in the schedTable."
        ::= { schedEntry 2 }

    schedDescr OBJECT-TYPE
        SYNTAX      SnmpAdminString
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
            "The human readable description of the purpose of this
             scheduling entry."
        DEFVAL      { "" }
        ::= { schedEntry 3 }

    schedInterval OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "seconds"
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
            "The number of seconds between two action invocations of
             a periodic scheduler.  Implementations must guarantee
             that action invocations will not occur before at least
             schedInterval seconds have passed.

             The scheduler must ignore all periodic schedules that
             have a schedInterval value of 0.  A periodic schedule
             with a scheduling interval of 0 seconds will therefore
             never invoke an action.

             Implementations may be forced to delay invocations in the
             face of local constraints.  A scheduled management function
             should therefore not rely on the accuracy provided by the
             scheduler implementation.

             Note that implementations which maintain a list of pending
             activations must re-calculate them when this object is
             changed."
        DEFVAL      { 0 }
```

```
        ::= { schedEntry 4 }

   schedWeekDay OBJECT-TYPE
       SYNTAX      BITS {
                       sunday(0),
                       monday(1),
                       tuesday(2),
                       wednesday(3),
                       thursday(4),
                       friday(5),
                       saturday(6)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The set of weekdays on which the scheduled action should
            take place.  Setting multiple bits will include several
            weekdays in the set of possible weekdays for this schedule.
            Setting all bits will cause the scheduler to ignore the
            weekday.

            Note that implementations which maintain a list of pending
            activations must re-calculate them when this object is
            changed."
       DEFVAL      { {} }
       ::= { schedEntry 5 }

   schedMonth OBJECT-TYPE
       SYNTAX      BITS {
                       january(0),
                       february(1),
                       march(2),
                       april(3),
                       may(4),
                       june(5),
                       july(6),
                       august(7),
                       september(8),
                       october(9),
                       november(10),
                       december(11)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The set of months during which the scheduled action should
            take place.  Setting multiple bits will include several
            months in the set of possible months for this schedule.
```

```
            Setting all bits will cause the scheduler to ignore the
            month.

            Note that implementations which maintain a list of pending
            activations must re-calculate them when this object is
            changed."
       DEFVAL      { {} }
       ::= { schedEntry 6 }

   schedDay OBJECT-TYPE
       SYNTAX      BITS {
                       d1(0),   d2(1),   d3(2),   d4(3),   d5(4),
                       d6(5),   d7(6),   d8(7),   d9(8),   d10(9),
                       d11(10), d12(11), d13(12), d14(13), d15(14),
                       d16(15), d17(16), d18(17), d19(18), d20(19),
                       d21(20), d22(21), d23(22), d24(23), d25(24),
                       d26(25), d27(26), d28(27), d29(28), d30(29),
                       d31(30),
                       r1(31),  r2(32),  r3(33),  r4(34),  r5(35),
                       r6(36),  r7(37),  r8(38),  r9(39),  r10(40),
                       r11(41), r12(42), r13(43), r14(44), r15(45),
                       r16(46), r17(47), r18(48), r19(49), r20(50),
                       r21(51), r22(52), r23(53), r24(54), r25(55),
                       r26(56), r27(57), r28(58), r29(59), r30(60),
                       r31(61)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The set of days in a month on which a scheduled action
            should take place.  There are two sets of bits one can
            use to define the day within a month:

            Enumerations starting with the letter 'd' indicate a
            day in a month relative to the first day of a month.
            The first day of the month can therefore be specified
            by setting the bit d1(0) and d31(30) means the last
            day of a month with 31 days.

            Enumerations starting with the letter 'r' indicate a
            day in a month in reverse order, relative to the last
            day of a month.  The last day in the month can therefore
            be specified by setting the bit r1(31) and r31(61) means
            the first day of a month with 31 days.

            Setting multiple bits will include several days in the set
            of possible days for this schedule.  Setting all bits will
            cause the scheduler to ignore the day within a month.
```

```
            Setting all bits starting with the letter 'd' or the
            letter 'r' will also cause the scheduler to ignore the
            day within a month.

            Note that implementations which maintain a list of pending
            activations must re-calculate them when this object is
            changed."
       DEFVAL      { {} }
       ::= { schedEntry 7 }

   schedHour OBJECT-TYPE
       SYNTAX      BITS {
                       h0(0),   h1(1),   h2(2),   h3(3),   h4(4),
                       h5(5),   h6(6),   h7(7),   h8(8),   h9(9),
                       h10(10), h11(11), h12(12), h13(13), h14(14),
                       h15(15), h16(16), h17(17), h18(18), h19(19),
                       h20(20), h21(21), h22(22), h23(23)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The set of hours within a day during which the scheduled
            action should take place.

            Note that implementations which maintain a list of pending
            activations must re-calculate them when this object is
            changed."
       DEFVAL      { {} }
       ::= { schedEntry 8 }

   schedMinute OBJECT-TYPE
       SYNTAX      BITS {
                       m0(0),   m1(1),   m2(2),   m3(3),   m4(4),
                       m5(5),   m6(6),   m7(7),   m8(8),   m9(9),
                       m10(10), m11(11), m12(12), m13(13), m14(14),
                       m15(15), m16(16), m17(17), m18(18), m19(19),
                       m20(20), m21(21), m22(22), m23(23), m24(24),
                       m25(25), m26(26), m27(27), m28(28), m29(29),
                       m30(30), m31(31), m32(32), m33(33), m34(34),
                       m35(35), m36(36), m37(37), m38(38), m39(39),
                       m40(40), m41(41), m42(42), m43(43), m44(44),
                       m45(45), m46(46), m47(47), m48(48), m49(49),
                       m50(50), m51(51), m52(52), m53(53), m54(54),
                       m55(55), m56(56), m57(57), m58(58), m59(59)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
```

```
          "The set of minutes within an hour when the scheduled action
           should take place.

           Note that implementations which maintain a list of pending
           activations must re-calculate them when this object is
           changed."
      DEFVAL      { {} }
      ::= { schedEntry 9 }

  schedContextName OBJECT-TYPE
      SYNTAX      SnmpAdminString (SIZE(0..32))
      MAX-ACCESS  read-create
      STATUS      current
      DESCRIPTION
          "The context which contains the local MIB variable pointed
           to by schedVariable."
      DEFVAL      { "" }
      ::= { schedEntry 10 }

  schedVariable OBJECT-TYPE
      SYNTAX      VariablePointer
      MAX-ACCESS  read-create
      STATUS      current
      DESCRIPTION
          "An object identifier pointing to a local MIB variable
           which resolves to an ASN.1 primitive type of INTEGER."
      DEFVAL      { zeroDotZero }
      ::= { schedEntry 11 }

  schedValue OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-create
      STATUS      current
      DESCRIPTION
          "The value which is written to the MIB object pointed to by
           schedVariable when the scheduler invokes an action.  The
           implementation shall enforce the use of access control
           rules when performing the set operation on schedVariable.
           This is accomplished by calling the isAccessAllowed abstract
           service interface as defined in RFC 2571.

           Note that an implementation may choose to issue an SNMP Set
           message to the SNMP engine and leave the access control
           decision to the normal message processing procedure."
      DEFVAL      { 0 }
      ::= { schedEntry 12 }

  schedType OBJECT-TYPE
```

```
       SYNTAX      INTEGER {
                       periodic(1),
                       calendar(2),
                       oneshot(3)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The type of this schedule.  The value periodic(1) indicates
            that this entry specifies a periodic schedule.  A periodic
            schedule is defined by the value of schedInterval.  The
            values of schedWeekDay, schedMonth, schedDay, schedHour
            and schedMinute are ignored.

            The value calendar(2) indicates that this entry describes a
            calendar schedule.  A calendar schedule is defined by the
            values of schedWeekDay, schedMonth, schedDay, schedHour and
            schedMinute.  The value of schedInterval is ignored.  A
            calendar schedule will trigger on all local times that
            satisfy the bits set in schedWeekDay, schedMonth, schedDay,
            schedHour and schedMinute.

            The value oneshot(3) indicates that this entry describes a
            one-shot schedule.  A one-shot schedule is similar to a
            calendar schedule with the additional feature that it
            disables itself by changing in the `finished'
            schedOperStatus once the schedule triggers an action.

            Note that implementations which maintain a list of pending
            activations must re-calculate them when this object is
            changed."
       DEFVAL      { periodic }
       ::= { schedEntry 13 }

   schedAdminStatus OBJECT-TYPE
       SYNTAX      INTEGER {
                       enabled(1),
                       disabled(2)
                   }
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The desired state of the schedule."
       DEFVAL      { disabled }
       ::= { schedEntry 14 }

   schedOperStatus OBJECT-TYPE
       SYNTAX      INTEGER {
```

```
                       enabled(1),
                       disabled(2),
                       finished(3)
                   }
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The current operational state of this schedule.  The state
            enabled(1) indicates this entry is active and that the
            scheduler will invoke actions at appropriate times.  The
            disabled(2) state indicates that this entry is currently
            inactive and ignored by the scheduler.  The finished(3)
            state indicates that the schedule has ended.  Schedules
            in the finished(3) state are ignored by the scheduler.
            A one-shot schedule enters the finished(3) state when it
            deactivates itself.

            Note that the operational state must not be enabled(1)
            when the schedRowStatus is not active."
       ::= { schedEntry 15 }

   schedFailures OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This variable counts the number of failures while invoking
            the scheduled action.  This counter at most increments once
            for a triggered action."
       ::= { schedEntry 16 }

   schedLastFailure OBJECT-TYPE
       SYNTAX      SnmpPduErrorStatus
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The most recent error that occurred during the invocation of
            a scheduled action.  The value noError(0) is returned
            if no errors have occurred yet."
       DEFVAL      { noError }
       ::= { schedEntry 17 }

   schedLastFailed OBJECT-TYPE
       SYNTAX      DateAndTime
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The date and time when the most recent failure occurred.
```

```
            The value '0000000000000000'H is returned if no failure
            occurred since the last re-initialization of the scheduler."
       DEFVAL      { '0000000000000000'H }
       ::= { schedEntry 18 }

   schedStorageType OBJECT-TYPE
       SYNTAX      StorageType
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "This object defines whether this scheduled action is kept
            in volatile storage and lost upon reboot or if this row is
            backed up by non-volatile or permanent storage.

            Conceptual rows having the value `permanent' must allow
            write access to the columnar objects schedDescr,
            schedInterval, schedContextName, schedVariable, schedValue,
            and schedAdminStatus.  If an implementation supports the
            schedCalendarGroup, write access must be also allowed to
            the columnar objects schedWeekDay, schedMonth, schedDay,
            schedHour, schedMinute."
       DEFVAL      { volatile }
       ::= { schedEntry 19 }

   schedRowStatus OBJECT-TYPE
       SYNTAX      RowStatus
       MAX-ACCESS  read-create
       STATUS      current
       DESCRIPTION
           "The status of this scheduled action.  A control that allows
            entries to be added and removed from this table.

            Note that the operational state must change to enabled
            when the administrative state is enabled and the row
            status changes to active(1).

            Attempts to destroy(6) a row or to set a row
            notInService(2) while the operational state is enabled
            result in inconsistentValue errors.

            The value of this object has no effect on whether other
            objects in this conceptual row can be modified."
       ::= { schedEntry 20 }

   schedTriggers OBJECT-TYPE
       SYNTAX      Counter32
       MAX-ACCESS  read-only
       STATUS      current
```

```
        DESCRIPTION
            "This variable counts the number of attempts (either
             successful or failed) to invoke the scheduled action."
        ::= { schedEntry 21 }

    --
    -- Notifications that are emitted to indicate failures.  The
    -- definition of schedTraps makes notification registrations
    -- reversible (see STD 58, RFC 2578).
    --

    schedTraps OBJECT IDENTIFIER ::= { schedNotifications 0 }

    schedActionFailure NOTIFICATION-TYPE
        OBJECTS     { schedLastFailure, schedLastFailed }
        STATUS      current
        DESCRIPTION
            "This notification is generated whenever the invocation of a
             scheduled action fails."
        ::= { schedTraps 1 }

    -- conformance information

    schedCompliances OBJECT IDENTIFIER ::= { schedConformance 1 }
    schedGroups      OBJECT IDENTIFIER ::= { schedConformance 2 }

    -- compliance statements

    schedCompliance2 MODULE-COMPLIANCE
        STATUS      current
        DESCRIPTION
            "The compliance statement for SNMP entities which implement
             the scheduling MIB."
        MODULE      -- this module
        MANDATORY-GROUPS {
               schedGroup2, schedNotificationsGroup
        }
        GROUP  schedCalendarGroup
        DESCRIPTION
            "The schedCalendarGroup is mandatory only for those
             implementations that support calendar based schedules."
        OBJECT schedType
        DESCRIPTION
            "The values calendar(2) or oneshot(3) are not valid for
             implementations that do not implement the
             schedCalendarGroup.  Such an implementation must return
             inconsistentValue error responses for attempts to set
             schedAdminStatus to calendar(2) or oneshot(3)."
```

```
        ::= { schedCompliances 2 }

    schedGroup2 OBJECT-GROUP
        OBJECTS {
            schedDescr, schedInterval, schedContextName,
            schedVariable, schedValue, schedType,
            schedAdminStatus, schedOperStatus, schedFailures,
            schedLastFailure, schedLastFailed, schedStorageType,
            schedRowStatus, schedTriggers
        }
        STATUS      current
        DESCRIPTION
            "A collection of objects providing scheduling capabilities."
        ::= { schedGroups 4 }

    schedCalendarGroup OBJECT-GROUP
        OBJECTS {
            schedLocalTime, schedWeekDay, schedMonth,
            schedDay, schedHour, schedMinute
        }
        STATUS      current
        DESCRIPTION
            "A collection of objects providing calendar based schedules."
        ::= { schedGroups 2 }

    schedNotificationsGroup NOTIFICATION-GROUP
        NOTIFICATIONS {
            schedActionFailure
        }
        STATUS      current
        DESCRIPTION
            "The notifications emitted by the scheduler."
        ::= { schedGroups 3 }

    --
    -- Deprecated compliance and conformance group definitions
    -- from RFC 2591.
    --

    schedCompliance MODULE-COMPLIANCE
        STATUS      deprecated
        DESCRIPTION
            "The compliance statement for SNMP entities which implement
             the scheduling MIB."
        MODULE      -- this module
        MANDATORY-GROUPS {
               schedGroup, schedNotificationsGroup
        }
```

```
        GROUP  schedCalendarGroup
        DESCRIPTION
            "The schedCalendarGroup is mandatory only for those
             implementations that support calendar based schedules."
        OBJECT schedType
        DESCRIPTION
            "The values calendar(2) or oneshot(3) are not valid for
             implementations that do not implement the
             schedCalendarGroup.  Such an implementation must return
             inconsistentValue error responses for attempts to set
             schedAdminStatus to calendar(2) or oneshot(3)."
        ::= { schedCompliances 1 }

    schedGroup OBJECT-GROUP
        OBJECTS {
            schedDescr, schedInterval, schedContextName,
            schedVariable, schedValue, schedType,
            schedAdminStatus, schedOperStatus, schedFailures,
            schedLastFailure, schedLastFailed, schedStorageType,
            schedRowStatus
        }
        STATUS      deprecated
        DESCRIPTION
            "A collection of objects providing scheduling capabilities."
        ::= { schedGroups 1 }

    END
```

5. Usage Examples

This section presents some examples how the scheduling MIB can be used to schedule scripts with the Script MIB [RFC3165] or to realize on-duty/off-duty schedules by modifying status objects of other MIB modules.

5.1. Starting a script to ping devices every 20 minutes

It is assumed that the schedule entry is owned by schedOwner = "joe" and its name is schedName = "ping". The instance identifier for the scheduling entry is therefore 3.106.111.101.4.112.105.110.103.

It is further assumed that the smLaunchTable entry is owned by smLaunchOwner = "joe" and its name is smLaunchName = "ping-devs". The complete object identifier for the smLaunchStart object is therefore
smLaunchStart.3.106.111.101.9.112.105.110.103.45.100.101.118.115.
The script lives in the context identified by the string "engine1".

The configuration of the scheduler entry which launches the script every 20 minutes would look as follows:

```
schedInterval.3.106.111.101.4.112.105.110.103 = 1200

schedValue.3.106.111.101.4.112.105.110.103 = 0
schedContextName.3.106.111.101.4.112.105.110.103 = "engine1"
schedVariable.3.106.111.101.4.112.105.110.103 =
  smLaunchStart.3.106.111.101.9.112.105.110.103.45.100.101.118.115

schedType.3.106.111.101.4.112.105.110.103 = periodic(1)
schedAdminStatus.3.106.111.101.4.112.105.110.103 = enabled(1)
schedStorageType.3.106.111.101.4.112.105.110.103 = nonVolatile(3)
schedRowStatus.3.106.111.101.4.112.105.110.103 = active(1)
```

All the remaining columns in the schedTable represent status information and are not shown here.

5.2. Starting a script at the next Friday the 13th

It is assumed that the schedule entry is owned by schedOwner = "joe" and its name is schedName = "13th". The instance identifier for the scheduling entry is therefore 3.106.111.101.4.49.51.116.104.

It is further assumed that the smLaunchTable entry is owned by smLaunchOwner = "joe" and its name is smLaunchName = "ghost". The complete object identifier for the smLaunchStart object is therefore smLaunchStart.3.106.111.101.5.103.104.111.115.116. The script lives in the context identified by the string "engine1".

The configuration of the scheduler entry which launches the script on the next Friday 13th at midnight would look as follows:

```
schedWeekDay.3.106.111.101.4.49.51.116.104 = { friday }
schedMonth.3.106.111.101.4.49.51.116.104 = {
      january, february, march, april, may, june,
      july, august, september, october, november, december
}
schedDay.3.106.111.101.4.49.51.116.104 = { d13 }
schedHour.3.106.111.101.4.49.51.116.104 = { h0 }
schedMinute.3.106.111.101.4.49.51.116.104 = { m0 }

schedValue.3.106.111.101.4.49.51.116.104 = 0
schedContextName.3.106.111.101.4.49.51.116.104 = "engine1"
schedVariable.3.106.111.101.4.49.51.116.104 =
  smLaunchStart.3.106.111.101.5.103.104.111.115.116
```

```
   schedType.3.106.111.101.4.49.51.116.104 = oneshot(3)
   schedAdminStatus.3.106.111.101.4.49.51.116.104 = enabled(2)
   schedStorageType.3.106.111.101.4.49.51.116.104 = nonVolatile(3)
   schedRowStatus.3.106.111.101.4.49.51.116.104 = active(1)
```

All the remaining columns in the schedTable represent status information and are not shown here.

5.3. Turning an interface off during weekends

This example assumes that a network interface should be taken down during weekends. The interface table (ifTable) of the IF-MIB [RFC2863] is assumed to exist in the context identified by an empty string and the index of the interface is ifIndex = 6.

The scheduling entry which brings the interface down on every Friday evening at 20:30 (8:30 pm) is owned by schedOwner = "bob" and its name is schedName = "if-off". The instance identifier for the scheduling entry is therefore 3.98.111.98.6.105.102.45.111.102.102.

```
   schedWeekDay.3.98.111.98.6.105.102.45.111.102.102 = { friday }
   schedMonth.3.98.111.98.6.105.102.45.111.102.102 = {
         january, february, march, april, may, june,
         july, august, september, october, november, december
   }
   schedDay.3.98.111.98.6.105.102.45.111.102.102 = {
         d1, d2, d3, d4, d5, d6, d7, d8, d9, d10,
         d11, d12, d13, d14, d15, d16, d17, d18, d19, d20,
         d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31
   }
   schedHour.3.98.111.98.6.105.102.45.111.102.102 = { h20 }
   schedMinute.3.98.111.98.6.105.102.45.111.102.102 = { m30 }
   schedValue.3.98.111.98.6.105.102.45.111.102.102 = down(2)
   schedContextName.3.98.111.98.6.105.102.45.111.102.102 = ""
   schedVariable.3.98.111.98.6.105.102.45.111.102.102 =
     ifAdminStatus.6

   schedType.3.98.111.98.6.105.102.45.111.102.102 = calendar(2)
   schedAdminStatus.3.98.111.98.6.105.102.45.111.102.102 = enabled(1)
   schedStorageType.3.98.111.98.6.105.102.45.111.102.102 =
     nonVolatile(3)
   schedRowStatus.3.98.111.98.6.105.102.45.111.102.102 = active(1)
```

The scheduling entry which brings the interface up on every Monday morning at 5:30 is owned by schedOwner = "bob" and its name is schedName = "if-on". The instance identifier for the scheduling entry is therefore 3.98.111.98.5.105.102.45.111.110.

The entry in the schedTable which brings the interface up again on every Monday morning at 5:30 looks as follows:

```
schedWeekDay.3.98.111.98.5.105.102.45.111.110 = { monday }
schedMonth.3.98.111.98.5.105.102.45.111.110 = {
      january, february, march, april, may, june,
      july, august, september, october, november, december
}
schedDay.3.98.111.98.5.105.102.45.111.110 = {
      d1, d2, d3, d4, d5, d6, d7, d8, d9, d10,
      d11, d12, d13, d14, d15, d16, d17, d18, d19, d20,
      d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31
}
schedHour.3.98.111.98.5.105.102.45.111.110 = { h5 }
schedMinute.3.98.111.98.5.105.102.45.111.110 = { m30 }

schedValue.3.98.111.98.5.105.102.45.111.110 = up(1)
schedContextName.3.98.111.98.5.105.102.45.111.110 = ""
schedVariable.3.98.111.98.5.105.102.45.111.110 = ifAdminStatus.6

schedType.3.98.111.98.5.105.102.45.111.110 = calendar(2)
schedAdminStatus.3.98.111.98.5.105.102.45.111.110 = enabled(1)
schedStorageType.3.98.111.98.5.105.102.45.111.110 = nonVolatile(3)
schedRowStatus.3.98.111.98.5.105.102.45.111.110 = active(1)
```

A similar configuration could be used to control other schedules. For example, one could change the "if-on" and "if-off" schedules to enable and disable the periodic scheduler defined in the first example.

6. Security Considerations

There are a number of management objects defined in this MIB that have a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

SNMPv1 by itself is not a secure environment. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model RFC 2574 [RFC2574] and the View-based Access Control Model RFC 2575 [RFC2575] is recommended.

It is then a customer/user responsibility to ensure that the SNMP entity giving access to an instance of this MIB, is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

Scheduled SNMP set operations must use the security credentials that were present when the corresponding row in the scheduling entry was created. An implementation must therefore record and maintain the credentials for every scheduling entry.

An implementation must ensure that access control rules are applied when doing the set operation. This is accomplished by calling the isAccessAllowed abstract service interface defined in RFC 2571 [RFC2571]:

```
statusInformation =          -- success or errorIndication
  isAccessAllowed(
  IN   securityModel         -- Security Model in use
  IN   securityName          -- principal who wants to access
  IN   securityLevel         -- Level of Security
  IN   viewType              -- read, write, or notify view
  IN   contextName           -- context containing variableName
  IN   variableName          -- OID for the managed object
       )
```

The securityModel, securityName and securityLevel parameters are set to the values that were recorded when the scheduling entry was created. The viewType parameter must select the write view and the contextName and variableName parameters are taken from the schedContextName and schedVariableName values of the scheduling entry.

This MIB limits scheduled actions to objects in the local MIB. This avoids security problems with the delegation of access rights. However, it might be possible for a user of this MIB to own some schedules that might trigger far in the future. This can cause security risks if the security administrator did not properly update the access control lists when a user is withdrawn from an SNMP engine. Therefore, entries in the schedTable SHOULD be cleaned up whenever a user is removed from an SNMP engine.

To facilitate the provisioning of access control by a security administrator using the View-Based Access Control Model (VACM) defined in RFC 2575 [RFC2575] for tables in which multiple users may need to independently create or modify entries, the initial index is used as an "owner index". Such an initial index has a syntax of

SnmpAdminString, and can thus be trivially mapped to a securityName or groupName as defined in VACM, in accordance with a security policy.

All entries in related tables belonging to a particular user will have the same value for this initial index. For a given user's entries in a particular table, the object identifiers for the information in these entries will have the same subidentifiers (except for the "column" subidentifier) up to the end of the encoded owner index. To configure VACM to permit access to this portion of the table, one would create vacmViewTreeFamilyTable entries with the value of vacmViewTreeFamilySubtree including the owner index portion, and vacmViewTreeFamilyMask "wildcarding" the column subidentifier. More elaborate configurations are possible.

7. Intellectual Property

8. Changes from RFC 2591

The following list documents major changes from the previous version of this document, published as RFC 2591:

- Updated the SNMP Management Framework boilerplate and the references.

- Added revision clauses to the module identity macro.

- Clarified the behavior during time transitions.

- Clarified that schedInterval and schedCalendarGroup objects can be modified regardless of the current value of schedRowStatus, schedAdminStatus and schedOperStatus.

- Added some additional boilerplate text to the security considerations section.

- Clarified that implementations must re-calculate any pending action invocations when scheduling parameters are modified.

- Clarified that schedOperStatus must not be enabled while the schedRowStatus is not active.

- Clarified that schedRowStatus can not be changed as long as the schedOperStatus is enabled.

- Clarified that implementations can delegate the isAccessAllowed check by sending themself an SNMP Set message.

- Added the schedTriggers object which counts the total number of triggers.

- Added DEFVALs for schedContextName, schedVariable, and schedValue and updated the schedRowStatus description.

- Deprecated schedCompliance, schedGroup and created schedCompliance2 and schedGroup2 that take care of the new schedTriggers object.

9. Acknowledgments

This document was produced by the IETF Distributed Management (DISMAN) working group.

10. References

[RFC2571] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

[RFC1155] Rose, M. and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, May 1990.

[RFC1212] Rose, M. and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.

[RFC1215] Rose, M., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.

[RFC2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC1157] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, May 1990.

[RFC1901] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[RFC1906] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[RFC2572] Case, J., Harrington D., Presuhn R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[RFC2574] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[RFC1905] 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.

[RFC2573] Levi, D., Meyer, P. and B. Stewart, "SNMPv3 Applications", RFC 2573, April 1999.

[RFC2575] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, April 1999.

[RFC2570] Case, J., Mundy, R., Partain, D. and B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

[RFC2028] Hovey, R. and S. Bradner, "The Organizations Involved in the IETF Standards Process", BCP 11, RFC 2028, October 1996.

[RFC3165] Levi, D. and J. Schoenwaelder, "Definitions of Managed Objects for the Delegation of Management Scripts", RFC 3165, August 2001.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

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

11. Editors' Addresses

David B. Levi
Nortel Networks
4401 Great America Parkway
Santa Clara, CA 95052-8185
USA

Phone: +1 865 686 0432
EMail: dlevi@nortelnetworks.com

Juergen Schoenwaelder
TU Braunschweig
Bueltenweg 74/75
38106 Braunschweig
Germany

Phone: +49 531 391-3283
EMail: schoenw@ibr.cs.tu-bs.de

12. Full Copyright Statement

Acknowledgement

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