Society of Cable Telecommunication Engineers

ENGINEERING COMMITTEE
HFC Management Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 38-8 2009

Hybrid Fiber/Coax Outside Plant Status Monitoring
SCTE-HMS-DOWNLOAD-MIB
Management Information Base (MIB) Definition

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or nonmember of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members, whether used domestically or internationally.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the Standards. Such adopting party assumes all risks associated with adoption of these Standards or Recommended Practices, and accepts full responsibility for any damage and/or claims arising from the adoption of such Standards or Recommended Practices.

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquires into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this standard have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at http://www.scte.org.

140 Philips Road
Exton, PA 19341

Contents

1. Scope

This document contains the definitions used to maintain one or more loadable firmware images on an HMS transponder.

2. Copyright

The MIB definition found in this document may be incorporated directly in products without further permission from the copyright owner, SCTE.

3. Normative Reference

IETF RFC 1155
SCTE 37

4. Informative Reference

None

5. Terms and Definitions

This document defines the following terms:

Management Information Base (MIB) - the specification of information in a manner that allows standard access through a network management protocol.

6. Requirements

This section defines the mandatory syntax of the SCTE-HMS-DOWNLOAD-MIB. It follows the IETF Simple Network Management Protocol (SNMP) for defining the managed objects.

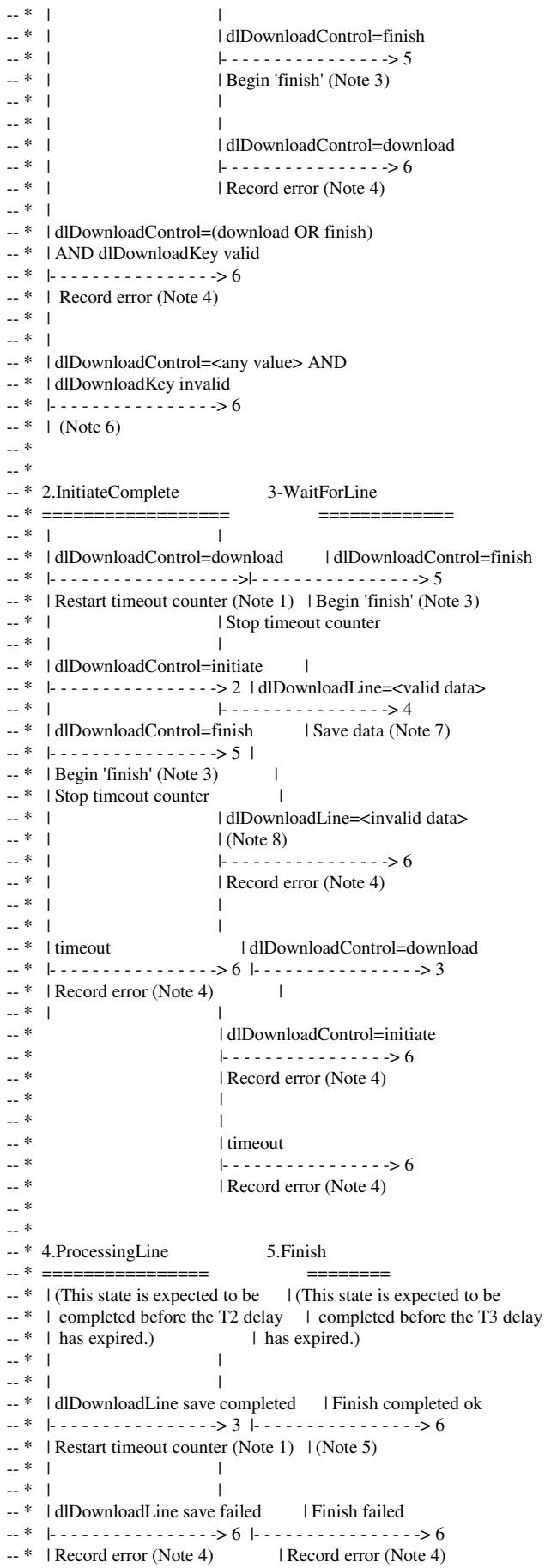
The syntax is given below.

```
-- ****************************************************************************
-- *
-- * Module Name: HMS063R6.MIB (SCTE 38-8)
-- *
-- * SCTE Status: ADOPTED JANUARY 11, 2002
-- *
-- *  Introduction:
-- *  ============
-- *  The Download MIB is used to maintain one or more loadable
-- *  firmware images on an HMS transponder. The MIB also has
-- *  support for downloading to sub-components managed by an
-- *  HMS transponder.
-- *
-- *  The firmware image for one vendor's transponder is not expected
-- *  to be interchangeable with another vendor's transponder. Care
-- *  must be used not to download the wrong image. The DEVICE-KEY
-- *  and dlDownloadKey are used to ensure that only the intended
-- *  transponders are modified.
-- *
-- *  The firmware may be transferred using broadcast, multicast,
-- *  or unicast addressing. When using broadcast or multicast
-- *  addressing, then suitable delays must be placed between
-- *  each transaction to allow the transponder time to process
-- *  the request. The value for these delays may be read from
-- *  the download distribution file.
-- *
-- *  Download Devices:
-- *  ================
-- *  The MIB is designed to support download to sub-components
-- *  attached to the transponder. These additional devices may
-- *  be located on the Transponder Interface Bus (TIB) or on a
-- *  vendor specific bus. The Transponder Interface Bus is defined
-- *  by SCTE 25-3 (formerly HMS022. )
-- *
-- *  Within this MIB, device number 1 specifies the transponder.
-- *  Device numbers 2 to 256 are reserved for TIB devices.  The
-- *  download device numbers 2 to 256 correspond to TIB addresses
-- *  1 to 255 respectively.  Download device numbers greater than
-- *  256 may be used for vendor specific non-TIB devices managed
-- *  by the transponder.
-- *
-- *  Not all transponders support TIB based devices.
-- *
-- *  The initial version of SCTE 25-3 (formerly HMS022) does not support firmware
-- *  download.
-- *
-- *
-- *  Distribution File Format:
-- *  ========================
-- *  The firmware update for a transponder is to be distributed in
-- *  a single file. The file has a header which describes information
-- *  used to configure a download. The header information consists
-- *  of a set of keywords with parameters, defined within comment lines.
-- *
-- *  Comment lines begin with a pair of back to back '-' characters.
-- *
-- *  The standard distribution file format for all vendors uses S-Records
-- *  to define the transponder firmware.  Using a common file format
-- *  permits standard download tools to be developed.
-- *
-- *  Each S-Record line may contains up to 255 bytes of data, which is
-- *  the maximum value for the S-Record length field.
-- *
-- *  Additional details on the S-Record format usage can be found
-- *  in the description of dlDownloadLine.
-- *
-- *
-- *  Distribution File Header:
-- *  ========================
-- *  The distribution file header will have the following keywords
-- *  located within successive comment lines:
-- *
```

```
-- *    VERS:<ASCII version, same as dlImageVersion>
-- *    DESC:<ASCII description, same as dlImageDescription>
-- *    T0:<decimal number>
-- *    T1:<decimal number>
-- *    T2:<decimal number>
-- *    T3:<decimal number>
-- *    DEVICE-KEY:<value for dlDownloadKey>
-- *    DEVICE:<1..n or PROMPT>
-- *    IMAGE:<1..n  or PROMPT>
-- *
-- *  All keywords begin one blank character after a comment introducer,
-- *  which simplifies parsing by the loader application. Comment lines
-- *  which do not have one of the keywords are ignored and can be used
-- *  to provide additional information.
-- *
-- *  The keywords must all be present before the first S-Record line.
-- *
-- *  Each parameter is expected to begin immediately following the ':'
-- *  character of each keyword.
-- *
-- *  All header keywords must be present. The list is not meant to
-- *  establish a specific order.
-- *
-- *  The VERS parameter reports the version of the firmware to the
-- *  loader. This string matches the value to be reported by the firmware
-- *  for dlImageVersion and dlActiveImageVersion. The version string
-- *  may be used as part of the dlDownloadKey. See dlDownloadKey for
-- *  additional information.
-- *
-- *  The DESC parameter reports the description of the firmware to the
-- *  loader. This string matches the value to be reported by the firmware
-- *  for dlImageDescription and dlActiveImageDescription.
-- *
-- *  The VERS and DESC keywords must match that which is to be reported
-- *  by the device. The corresponding strings are expected to be embedded
-- *  in the device by some means. The strings may be embedded in the
-- *  firmware image or possibly part of the S0 record which is then
-- *  stored. The S0 record format is vendor specific.
-- *
-- *  The T0, T1, T2, and T3 parameters specify the time delays described in
-- *  the download example for multiple transponders. The delays are specified
-- *  in units of milliseconds. These delays represent minimum values.
-- *
-- *  The DEVICE-KEY value is used to set dlDownloadKey. This value must
-- *  match dlDeviceKey of the active firmware image for the device specified
-- *  by dlDownloadDevice (and therefore the DEVICE keyword). Unless the
-- *  correct string is written to dlDownloadKey, the transponder will not
-- *  permit a download sequence to commence.
-- *
-- *  The DEVICE-KEY minimum string definition starts with the first 3
-- *  octets of physical address (the vendor's OUI) as a hexadecimal ASCII
-- *  string. This ensures no units will accidentally accept firmware from
-- *  the wrong vendor. Additional octets may be specified after this 6 byte
-- *  prefix field by each vendor. This string must match that which is
-- *  embedded in the device, and can be confirmed via dlDeviceKey.
-- *  It is expected that all firmware images for one device will have
-- *  the same dlDeviceKey. See the description of dlDownloadKey for
-- *  additional details.
-- *
-- *  The IMAGE value is used to specify the value of dlDownloadImage.
-- *  The parameter value range is 1 to 'N', where 'N' is vendor specific.
-- *  Rather than specifying a decimal number, the PROMPT string specifies
-- *  that the user must be prompted for the value.
-- *
-- *  The DEVICE value is used to specify the value of dlDownloadDevice.
-- *  The parameter value range is 1 to 'N', where 'N' is vendor specific.
-- *  Rather than specifying a decimal number, the PROMPT string specifies
-- *  that the user must be prompted for the value.
-- *
-- *
-- *  Download Sequence Examples:
-- *  ===========================
```

```
-- *  The typical sequence of events which occur when loading a
-- *  transponder firmware image are demonstrated by two examples.
-- *  The download process is not restricted to the sequence
-- *  shown in these examples.
-- *
-- *  The typical sequence of events which occur when loading a
-- *  transponder firmware image using broadcast or multicast
-- *  addressing are:
-- *
-- *    Read default download parameters from distribution file,
-- *    ... such as T0, T1, T2, T3, DEVICE-KEY, etc.
-- *
-- *    Write DEVICE-KEY to dlDownloadKey and wait T0.
-- *    Set dlDownloadDevice and wait T0.
-- *    Set dlDownloadImage and wait T0.
-- *    Optionally set dlDownloadTimeout and wait T0.
-- *
-- *    Set dlDownloadControl to initiate(1).
-- *    Wait T1 to allow sufficient time for dlDownloadStatus
-- *    ... to go to initiateComplete(2).
-- *
-- *    Set dlDownloadControl to download(2).
-- *
-- *    UNTIL all lines of firmware update transferred:
-- *        Write next dlDownloadLine.
-- *        Wait T2 to allow sufficient time for dlDownloadStatus
-- *        ... to go to waitingForLine(3).
-- *    END.
-- *
-- *    Set dlDownloadControl to finish(3).
-- *    Wait T3 to allow sufficient time for dlDownloadStatus
-- *    ... to go to done(6).
-- *
-- *    Use trap handling to detect errors.
-- *
-- *    Optionally, read dlDownloadErrorStatus from each transponder
-- *    ... affected to verify no errors occurred.
-- *
-- *  An alternate approach is to address each transponder via unicast
-- *  addressing. While the same sequence of events could be used,
-- *  it is also possible to query the transponder during each step to
-- *  monitor status. Here is an example for unicast addressing:
-- *
-- *    Write DEVICE-KEY to dlDownloadKey.
-- *    Set dlDownloadDevice.
-- *    Set dlDownloadImage.
-- *    Optionally set dlDownloadTimeout.
-- *
-- *    Set dlDownloadControl to initiate(1).
-- *    Wait for dlDownloadStatus to go to initiateComplete(2).
-- *    Set dlDownloadControl to download(2).
-- *    UNTIL all lines of firmware update transferred:
-- *        Set next dlDownloadLine.
-- *        Wait for dlDownloadStatus to go to waitingForLine(3).
-- *    END.
-- *
-- *    Set dlDownloadControl to finish(3).
-- *    Wait for dlDownloadStatus to go to done(6).
-- *
-- *    Read dlDownloadErrorStatus and verify no errors occurred.
-- *
-- *
-- *  Download State Diagram:
-- *  =======================
-- *  The detailed relationships between objects in this MIB are shown
-- *  in the following state diagram. The states are labelled based on
-- *  dlDownloadStatus.
-- *
-- *  For each state transition, the condition for the change is defined
-- *  immediately above the transition line, the actions are shown
-- *  immediately below the transition line. For this diagram the '=' symbol
-- *  indicates a SNMP Set operation.
```

```
-- *
-- *  Within the state diagram, the first transition line shows the normal
-- *  flow.
-- *
-- *  Any SNMP Set of dlDownloadControl to a value other than initiate(1),
-- *  download(2), or finish(3) value results in a SNMP badValue error,
-- *  and no change in the state of the download.
-- *
-- *  With a change to a new state, there is an implicit action in which
-- *  dlDownloadStatus is set to the new state.
-- *
-- *  The dlDownloadLine can only be written to when dlDownloadControl
-- *  is download(2) and the dlDownloadKey is valid. A write at any other
-- *  time results only in a badValue error and no change to the state
-- *  of the download.
-- *
-- *  The ability to record an error requires that dlDownloadErrorStatus
-- *  be clear, hence zero length. The dlDownloadErrorStatus is cleared
-- *  when dlDownloadControl is set to initiate(1) to begin a download.
-- *
-- *  The hmsDownloadStatus trap is generated at the same time that
-- *  dlDownloadErrorStatus is set internally to record an error. If an
-- *  error has already been recorded, then subsequent errors are not
-- *  recorded, and subsequent traps are dropped.
-- *
-- *  Only one hmsDownloadStatus trap is to be pending at any one time.
-- *  If the MAC TALK command is not used to retrieve traps, then it
-- *  is possible to loose old traps. When the dlDownloadControl is
-- *  set to initiate, a pending hmsDownloadStatus trap is to be dropped.
-- *
-- *  For a given implementation the transient states associated
-- *  with initiateInProgress, processingLine, and finishInProgress
-- *  may never be reported through dlDownloadStatus. If no SNMP
-- *  commands can be processed during these states, then the state
-- *  machine to be implemented by a transponder is greatly reduced.
-- *
-- *  The objects dlDownloadDevice, dlDownloadImage, and dlDeviceKey
-- *  should only to be written when dlDownloadStatus is done(6). Attempts
-- *  to change these objects during any other state results in termination
-- *  of the download in progress and an error being recorded. The actions
-- *  associated with detecting an error are described by Note 4 of the
-- *  download state diagram. The object value will not be cleared however,
-- *  and will accept the new value.
-- *
-- *  The initial state for the Download MIB is dlDownloadStatus is
-- *  done(6) and dlDownloadControl is finish(3). This is also the
-- *  state of the MIB at the end of a download, whether is was successful
-- *  or not.  A successful download is indicated by a zero length
-- *  dlDownloadErrorStatus.
-- *
-- *
-- *  6-Done                          1.InitiateInProgress
-- *  ======                           ====================
-- *   |                             | (This state is expected to be
-- *   |                             |  completed before the T1 delay
-- *   |                             |  has expired.)
-- *   | dlDownloadControl=initiate  |
-- *   | AND dlDownloadKey valid     |
-- *   | AND dlDownloadDevice valid  |
-- *   | AND dlDownloadImage valid   | Initiate has completed
-- *   |- - - - - - - - - - - - - - ->|- - - - - - - - - - - - - - -> 2
-- *   | Begin 'initiate' (Note 2)   | Start timeout counter (Note 1)
-- *   | Clear dlDownloadErrorStatus |
-- *   | Clear pending download status trap |
-- *   |                             | Initiate has failed
-- *   |                             |- - - - - - - - - - - - - - -> 6
-- *   | dlDownloadControl=initiate  | Record error (Note 4)
-- *   | AND dlDownloadKey valid     |
-- *   | AND (dlDownloadDevice invalid OR  |
-- *   |      dlDownloadImage invalid      | dlDownloadControl=initiate
-- *   |- - - - - - - - - - - - - - -> 6  |- - - - - - - - - - - - - - -> 1
-- *   |  Record error (Note 4)      |
```


| | dlDownloadControl=finish
|- - - - - - - - - - - - - - - -> 5
| Begin 'finish' (Note 3)
| dlDownloadControl=download
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| dlDownloadControl=(download OR finish)
| AND dlDownloadKey valid
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| dlDownloadControl=<any value> AND
| dlDownloadKey invalid
|- - - - - - - - - - - - - - - -> 6
| (Note 6)
2.InitiateComplete
==========
3-WaitForLine
==========
| dlDownloadControl=download
|- - - - - - - - - - - - - - - - - - ->
| Restart timeout counter (Note 1)
| dlDownloadControl=finish
|- - - - - - - - - - - - - - - - - - ->
| Begin 'finish' (Note 3)
| dlDownloadControl=initiate
|- - - - - - - - - - - - - - - -> 2
| Stop timeout counter
| dlDownloadLine=<valid data>
|- - - - - - - - - - - - - - - -> 4
| Save data (Note 7)
| dlDownloadLine=<invalid data>
| (Note 8)
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| timeout
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| dlDownloadControl=download
|- - - - - - - - - - - - - - - -> 3
| dlDownloadControl=initiate
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| timeout
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
4.ProcessingLine
==========
5.Finish
========
| (This state is expected to be
| completed before the T2 delay
| has expired.)
| (This state is expected to be
| completed before the T3 delay
| has expired.)
| dlDownloadLine save completed
|- - - - - - - - - - - - - - - -> 3
| Finish completed ok
|- - - - - - - - - - - - - - - -> 6
| Restart timeout counter (Note 1)
| (Note 5)
| dlDownloadLine save failed
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)
| Finish failed
|- - - - - - - - - - - - - - - -> 6
| Record error (Note 4)

```
-- *   |                              |
-- *   |                              |
-- *   | dlDownloadControl=finish     |
-- *   |- - - - - - - - - - - - - - - ->| dlDownloadControl=finish
-- *   | Begin 'finish' (Note 3)      |- - - - - - - - - - - - - - - -> 5
-- *   | Stop timeout counter         |
-- *   |                              |
-- *   |                              | dlDownloadControl=download OR
-- *   |                              | dlDownloadControl=initiate
-- *   |                              |- - - - - - - - - - - - - - - -> 6
-- *   |                              | Record error (Note 4)
-- *   | dlDownloadControl=download OR |
-- *   | dlDownloadControl=initiate   |
-- *   |- - - - - - - - - - - - - - -> 6 |
-- *   | Record error (Note 4)        |
-- *
-- *
-- *
-- * Note 1: The timeout counter is used to ensure that the download does
-- *         not become accidentally locked in a specific state. There is
-- *         only one timeout counter referred to in the state diagram.
-- *         The value for the timeout is specified by dlDownloadTimeout
-- *         for the device selected by dlDownloadDevice.
-- *
-- * Note 2: The Initiate operations are vendor specific. During this state
-- *         any necessary preparation for download can be made, such as
-- *         erasing the region which will accept the download.
-- *
-- * Note 3: The Finish operations are vendor specific. A typical operation
-- *         performed during this state would be validation of the downloaded
-- *         image.
-- *
-- * Note 4: The dlDownloadDevice and dlDownloadImage objects are set to zero.
-- *         The dlDownloadKey is set to a zero length string.
-- *         The dlDownloadErrorStatus is set to a vendor defined message.
-- *         The dlDownloadControl is set to finish(3).
-- *         An hmsDownloadStatus trap is registered.
-- *         The timeout counter, described in note 1, must be stopped.
-- *         If a set of dlDownloadControl was the action which triggered the
-- *         error, then a badValue error results.
-- *
-- * Note 5: The dlDownloadDevice and dlDownloadImage objects are set to zero.
-- *         The dlDownloadKey is set to a zero length string.
-- *
-- * Note 6: Return SNMP badValue error, no change to download state
-- *         as dlDownloadKey is invalid.
-- *
-- * Note 7: The data from dlDownloadLine is saved in an appropriate manner
-- *         within the transponder. The transponder must be able to accept
-- *         a download image which is sent more than once during the same
-- *         download cycle, hence while dlDownloadControl is still set to
-- *         download(2). When using Broadcast or multicast addressing,
-- *         sending the image more than once may be done in case noise
-- *         corrupts one or more packets.
-- *
-- * Note 8: Invalid data written to dlDownloadLine refers to data which
-- *         is unexpected in a given address range, data which is outside
-- *         the expected address range, or an illegal type field. A vendor
-- *         may have additional qualifiers to detect invalid data.
-- *         Lines which do not begin with an 'S' are ignored, and are not
-- *         considered to be in error.
-- *
-- *
-- *  Additional Notes:
-- *  ================
-- *  A transponder is in the download state when dlDownloadStatus is
-- *  not done(6). While a transponder is in the download state:
-- *
-- *    1)All MAC layer commands operate the same as if a
-- *      download was not in progress. As an operational
-- *      issue it is advisable not to change the forward
-- *      or reverse frequencies while a download is in progress.
```

```
-- *
-- *    2)The only MIB objects guaranteed to be supported
-- *      while a download is in progress are those MIB objects
-- *      from HMS063, which is the Download MIB.
-- *
-- *    3)MIB objects not supported during a download will
-- *      report the error NoSuchName. This behaviour is
-- *      consistent with community profiles and MIB views
-- *      as defined in RFC1157.
-- *
-- *    4)Upon completion of a download, then the MIB objects
-- *      which had restricted access during the download are
-- *      again accessible.
-- *
-- *  The dlDownloadTimeout prevents a transponder from remaining in the
-- *  download state in the event that a download operation was interrupted.
-- *
-- *  Upon completion of a successful download the transponder may transfer
-- *  control to the new image, or continue to run the current. The behaviour
-- *  is determined by the object dlDownloadOption. This object is optional,
-- *  and if not implemented, the transponder is expected to transfer control
-- *  to the successfully downloaded image.
-- *
-- *  If dlDownloadOption is noAction, then control may be transferred to a
-- *  downloaded image at a later point by setting dlStartupImage followed
-- *  by a reset of the device. Setting dlStartupImage for the transponder
-- *  does not restart the transponder. The transponder can be reset remotely
-- *  using the commonReset object.
-- *
-- ******************************************************************************

SCTE-HMS-DOWNLOAD-MIB DEFINITIONS ::= BEGIN

IMPORTS
    OBJECT-TYPE        FROM RFC-1212
    TRAP-TYPE          FROM RFC-1215
    DisplayString      FROM RFC1213-MIB
    scteHmsTree        FROM SCTE-ROOT
    commonPhysAddress  FROM SCTE-HMS-COMMON-MIB
    commonLogicalID    FROM SCTE-HMS-COMMON-MIB
    downloadIdent      FROM SCTE-HMS-ROOTS
    ;

downLoad            OBJECT IDENTIFIER ::= { downloadIdent 1 }
transponderImage    OBJECT IDENTIFIER ::= { downloadIdent 2 }

-- *
-- *    ***************************************
-- *    Firmware Download Management Group
-- *    ***************************************
-- *    This portion of the Download MIB defines the core objects
-- *    used to manage the download of a firmware image or block
-- *    of data.
-- *

dlDownloadDevice  OBJECT-TYPE
   SYNTAX  INTEGER
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
      "This object defines the device which is to have firmware
      downloaded.

      A value of 1 specifies the transponder. A value of 2 to 256
      specifies TIB devices 1 to 255. Values of dlDownloadDevice
      greater than 256 may be used for non-TIB devices managed by
      the transponder.

      This indexing scheme is used to avoid using an index value of 0
      to identify the transponder. This indexing scheme matches that of
      dlTransponderTable.
```

Not all transponders support TIB based devices.

There are no restrictions on the value of this object.
When dlDownloadControl is set to initiate(1), an invalid value
for dlDownloadImage results in the inability to set dlDownloadControl.

This value for this object reverts back to 0 after the end of
any download or download error. As valid indices begins at 1,
this provides some added security to prevent accidental erasure.

The value of this object should only be changed when
dlDownloadStatus is done(6). Attempts to change object during any other
state results in termination of the download in progress and an
error being recorded. The actions associated with detecting an error
are described by Note 4 of the download state diagram presented
earlier in the document. The object value will not be cleared and
the new value shall be accepted.

The dlDownloadKey does not affect this object.

The default value is 0. This object is volatile."
::= { downLoad 2 }

```
dlDownloadImage  OBJECT-TYPE
   SYNTAX  INTEGER
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
```

"The image number which to be modified during the next download.

There are no restrictions on the value of this object.
When dlDownloadControl is set to initiate(1), an invalid value
for dlDownloadImage results in the inability to set dlDownloadControl.

This value for this object reverts back to 0 after the end of
any download or download error. As valid image numbers begin at 1,
this provides some added security to prevent accidental erasure.

The value of this object should only be changed when
dlDownloadStatus is done(6). Attempts to change the object
during any other state results in termination of the download
in progress and an error being recorded. The actions associated
with detecting an error are described by Note 4 of the download
state diagram presented earlier in the document. The object
value will not be cleared and the new value shall be accepted.

The dlDownloadKey does not affect this object.

The default value is 0. This object is volatile."
::= { downLoad 3 }

```
dlDownloadKey  OBJECT-TYPE
   SYNTAX  DisplayString
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
```

"This string is used as a security key to enable downloads.
Unless this string is correctly set, the transponder will not
permit a download sequence to commence. This string is useful
for downloading to multiple transponders via broadcast or
multicast addressing.

The dlDownloadKey must begin with the dlDeviceKey string of the
active firmware image for the device specified by dlDownloadDevice
if a download is to be permitted. The dlDeviceKey value can also
be obtained from the DEVICE-KEY parameter in the distribution file
header. The dlDeviceKey value is embedded in the running firmware,
which permits the security check to be made. It is expected that
all firmware images for one device will have the same dlDeviceKey.

The value written to dlDownloadKey may have the version string of
the new firmware appended to dlDeviceKey. The key will only

```
        permit the download if the version of the download firmware is
        newer than that of the firmware being replaced. If the version
        string is not present, then the version check is not made.

        As inclusion of the version string in dlDownloadKey is optional,
        this should not be relied upon as the  means by which the version
        string is transferred to the device.

        This value for this object reverts back to a zero length
        string after the end of any download or download error.

        The value of this object should only be changed when
        dlDownloadStatus is done(6). Attempts to change the object
        during any other state results in termination of the download
        in progress and an error being recorded. The actions associated
        with detecting an error are described by Note 4 of the download
        state diagram presented earlier in the document. The object
        value will not be cleared and the new value shall be accepted.

        The default value is a 0 length string. This object is volatile."
    ::= { downLoad 4 }

dlDownloadControl  OBJECT-TYPE
    SYNTAX  INTEGER {
        initiate (1),
        download (2),
        finish   (3)
    }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The dlDownloadControl manages the different stages of
        firmware download.

        The dlDownloadControl can be set to initiate(1) only
        when dlDownloadStatus is done(6) and dlDownloadKey
        is valid.

        The dlDownloadControl can be set to download(2) only
        when dlDownloadStatus is initiateComplete(2).

        The dlDownloadControl can be set to finish(3) at any time
        that dlDownloadStatus is not done(6).

        The default value is finish(3). This object is volatile.

        Details of the dlDownloadControl usage and interaction with
        other objects from this MIB are defined in the download
        state diagram presented earlier in the document."
    ::= { downLoad 5 }

dlDownloadStatus  OBJECT-TYPE
    SYNTAX  INTEGER {
        initiateInProgress (1),
        initiateComplete (2),
        waitingForLine (3),
        processingLine (4),
        finishInProgress (5),
        done (6)
    }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The dlDownloadStatus reports the different stages
        of firmware download.

        Upon dlDownloadControl being set to initiate(1), the
        dlDownloadStatus changes to initiateInProgress(1).
        Once initiateInProgress(1) has completed, the state
        changes to initiateComplete(2). During initiateInProgress(1)
        any necessary preparation for download can be made,
        such as erasing the region which will accept the
```

```
        download data.

        Upon dlDownloadControl being set to download(2), the
        dlDownloadStatus changes to waitingForLine(3). Upon
        a line of information being written to dlDownloadLine,
        the dlDownloadStatus changes to processingLine(4).

        Once the line of information has been saved, the
        dlDownloadStatus returns to waitingForLine(3) to
        indicate that the next record may be sent.

        Upon dlDownloadControl being set to finish(3), the
        dlDownloadStatus changes to finishInPogress(5).
        A typical operation performed during this state
        would be validation of the downloaded image. Upon
        completion of this state, dlDownloadStatus changes
        to done(6).

        Details of the dlDownloadStatus usage and interaction with
        other objects from this MIB are defined in the download
        state diagram presented earlier in the document."
    ::= { downLoad 6 }

dlDownloadErrorStatus  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE(0..128))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The error description message for the most recent download
        attempt. If no error occurs, then the string is zero length.

        The dlDownloadErrorStatus captures the first error to occur
        after setting dlDownloadControl to initiate. Subsequent errors
        are not recorded. This is accomplished by only permitting
        errors to be recorded when dlDownloadErrorStatus is a zero
        length string.

        The dlDownloadErrorStatus is cleared when dlDownloadControl is
        set to initiate(1) to begin a download."
    ::= { downLoad 7 }

dlDownloadLine  OBJECT-TYPE
    SYNTAX  OCTET STRING
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The dlDownloadLine is used to transfer a portion of the
        firmware image for a device. Each portion transferred
        by this object is referred to as a line.

        As the firmware image is unique to each vendor's device,
        care must be taken not to transfer the wrong image. The use
        of the dlDownloadKey and DEVICE-KEY resolves this problem.

        Each line begins with the first two characters from a
        S-Record line. The first character is therefore expected
        to be an ASCII 'S'. The second character is the ASCII
        character which defines the record type. The record type
        values range from and include ASCII '0' to '9'.

        Lines which do not begin with an 'S' are ignored.

        The remainder of the S-Record line is then presented in
        the download line in the binary form of the data. This
        includes all fields from the length to the checksum.

        The loader must not transform one data record type into
        another (e.g., S1 to S3), in the event the other type
        is not supported by the transponder.

        The loader must present one S-Record line as one dlDownloadLine,
        in the event the file has been optimised for the target.
```

```
        The dlDownloadLine can only be written to when dlDownloadControl
        is download(2). A write at any other time results only in a
        BadValue error and no change to the state of the download.

        When the object is read, a zero length octet string is
        returned.

        The following examples demonstrate how the download application
        converts an S-Record line into an instance of dlDownloadLine:

         S1050260EA812D     -> 53 31 05 02 60 EA 81 2D

         S30700000260EA812B -> 53 33 07 00 00 02 60 EA 81 2B

        In the example characters to the left of '->' represent the S-Record
        line. The characters to the right of '->' indicate the dlDownloadLine
        data in hexadecimal notation.

        Both examples represent the same two bytes of data. The address field
        for both examples contains the same value of 260 hexadecimal.

        The device must be able to accept a download image which is
        sent more than once during the same download cycle, hence while
        dlDownloadControl is still set to download(2). When using Broadcast
        or multicast addressing, sending the image more than once may be
        done to protect against noise corruption of one or more packets."
    ::= { downLoad 8 }

-- *
-- *    ******************************
-- *    Transponder Image Group
-- *    ******************************
-- *    This portion of the Download MIB presents details of the
-- *    transponder firmware image(s) and the ability to download
-- *    images.
-- *

transponderTable  OBJECT-TYPE
    SYNTAX SEQUENCE OF TransponderEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "The table containing the status of firmware images for all
        components of a transponder."
    ::= { transponderImage 1 }

transponderEntry  OBJECT-TYPE
    SYNTAX  TransponderEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A list of firmware image information about a component of
        the transponder."
    INDEX   { dlTransponderDevice }
    ::= { transponderTable 1 }

TransponderEntry ::= SEQUENCE {
    dlTransponderDevice
        INTEGER,
    dlNumberImages
        INTEGER,
    dlActiveImage
        INTEGER,
    dlActiveImageVersion
        DisplayString,
    dlActiveImageDescription
        DisplayString,
    dlActiveImageAccess
        INTEGER,
    dlStartupImage
        INTEGER,
```

```
    dlDeviceKey
        DisplayString,
    dlDownloadOption
        INTEGER,
    dlDownloadTimeout
        INTEGER
}

dlTransponderDevice  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The device index into the table to select a specific transponder
         component.

         A value of 1 specifies the transponder. A value of 2 to 256
         specifies TIB devices 1 to 255. Values of dlTransponderDevice
         greater than 256 may be used for non-TIB devices managed by
         the transponder.

         This indexing scheme is used to avoid using an index value of 0
         to identify the transponder."
    ::= { transponderEntry 1 }

dlNumberImages  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The maximum number of firmware images supported by the unit."
    ::= { transponderEntry 2 }

dlActiveImage  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The image number which is currently running.
         The value of dlActiveImage cannot exceed dlNumberImages."
    ::= { transponderEntry 3 }

dlActiveImageVersion  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE(0..32))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The version string for the active image."
    ::= { transponderEntry 4 }

dlActiveImageDescription  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE(0..64))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The description string for the active image."
    ::= { transponderEntry 5 }

dlActiveImageAccess  OBJECT-TYPE
    SYNTAX  INTEGER {
        overwriteAllowed (1),
        overwriteNotAllowed (2)
    }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Report if the image number corresponding to the active
         image can be downloaded. System which support only 1 for
         dlNumberImages must only report overwriteAllowed(1).

         The ability to overwrite the active image may not be possible
         on some systems due to the type of storage device used."
```

```
    ::= { transponderEntry 6 }

dlStartupImage  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The image to be used after the next reset.
         If the image specified is not present, then the transponder
         must determine which alternate image should be used.
         The algorithm used is vendor specific.

         This object is non-volatile.

         The value of dlStartupImage cannot exceed dlNumberImages.
         The dlImageStatus for the image used must be validApplication(2)."
    ::= { transponderEntry 7 }

dlDeviceKey  OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The device identification string is a read-only string
         provided by the vendor for a given type of transponder.
         This value is expected to be read from the distribution
         header file and is specified by the DEVICE-KEY keyword.
         As the 'device key' is embedded in the firmware, the
         DEVICE-KEY value must be the same as dlDeviceKey.

         This string is used as a security key to enable firmware
         downloads to occur. Unless this string is written to
         dlDownloadKey, the transponder will not permit
         a download sequence to commence. See the description
         of dlDownloadKey for additional details.

         The minimum string definition starts with the first 3
         octets of physical address (the vendor's OUI) as a hexadecimal
         ASCII string. This ensures no units will accidentally accept
         firmware from the wrong vendor. Additional octets may be
         specified after this 6 byte prefix field by each vendor."
    ::= { transponderEntry 8 }

dlDownloadOption  OBJECT-TYPE
    SYNTAX  INTEGER {
        setStartupAndReset(1),
        noAction (2)
    }
    ACCESS  read-write
    STATUS  optional
    DESCRIPTION
        "The object specifies the action to be taken once a
         firmware image has been successfully downloaded.

         The setStartupAndReset(1) will copy dlDownloadImage to
         dlStartupImage and reset the unit. The reset is the same
         action as if commonReset was set to 1.

         The noAction(2) results in the transponder continuing
         to run the current image. If the transponder only supports
         one image, then new image must take over.

         The default is setStartupAndReset. If dlDownloadOption
         is not implemented, then the transponder implicitly
         operates as if setStartupAndReset was set.

         This object is volatile, hence the value always returns
         to setStartupAndReset(1) when the unit powers up."
    ::= { transponderEntry 9 }

dlDownloadTimeout  OBJECT-TYPE
    SYNTAX  INTEGER (60..300)
```

```
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The object specifies the maximum amount of time which may occur
         between successive writes to dlDownloadLine. This is accomplished
         by limiting the amount of time dlDownloadStatus can be in
         the waitingForLine state.

         This timeout also limits the amount of time while dlDownloadStatus
         is in the initiateComplete(2) state.

         The timeout is specified in units of seconds.

         The default value is 60. The object is volatile.

         The value of dlDownloadTimeout can only be changed when
         dlDownloadStatus is done(6). Attempts to change the object
         during any other state results only in a badValue error."
    ::= { transponderEntry 10 }



dlImageTable  OBJECT-TYPE
    SYNTAX SEQUENCE OF DlImageEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "The table containing the status of firmware images."
    ::= { transponderImage 2 }

dlImageEntry  OBJECT-TYPE
    SYNTAX  DlImageEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A list of information for the status of firmware images."
    INDEX   { dlImageDevice, dlImageIndex }
    ::= { dlImageTable 1 }

DlImageEntry ::= SEQUENCE {
    dlImageDevice
        INTEGER,
    dlImageIndex
        INTEGER,
    dlImageStatus
        INTEGER,
    dlImageAccess
        INTEGER,
    dlImageVersion
        DisplayString,
    dlImageDescription
        DisplayString
}

dlImageDevice  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A value of 1 specifies the transponder. A value of 2 to 256
         specifies TIB devices 1 to 255. Values of dlImageIndex
         greater than 256 may be used for non-TIB devices managed by
         the transponder.

         This indexing scheme is used to avoid using an index value of 0
         to identify the transponder. This indexing scheme matches that of
         dlTransponderTable."
    ::= { dlImageEntry 1 }

dlImageIndex  OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
```

```
    STATUS  mandatory
    DESCRIPTION
        "Index into dlImageTable.
         The value of dlDownloadImage cannot exceed dlNumberImages for
         the respective device."
    ::= { dlImageEntry 2 }

dlImageStatus  OBJECT-TYPE
    SYNTAX  INTEGER {
        invalid (1),
        validApplication (2),
        validData (3)
    }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Report the validity of the download image.

         The mechanism by which an image is determined to be an
         application (executable image) or simply data is vendor specific.
         Only an image which is validApplication(2) can be selected via
         dlStartupImage.

         The error checking scheme used for an image is vendor
         specific."
    ::= { dlImageEntry 3 }

dlImageAccess  OBJECT-TYPE
    SYNTAX  INTEGER {
        imageAccessReadWrite (1),
        imageAccessReadOnly (2)
    }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Report the access rights to the download image. It is
         expected that most images will be read-write. At least
         one image may be read-only if a permanent default image
         is implemented."
    ::= { dlImageEntry 4 }

dlImageVersion  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE(0..32))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The version string for the image."
    ::= { dlImageEntry 5 }

dlImageDescription  OBJECT-TYPE
    SYNTAX  DisplayString (SIZE(0..64))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The description string for the image."
    ::= { dlImageEntry 6 }

-- *
-- *    ******************************
-- *    Download MIB Traps
-- *    ******************************
-- *
-- * The following definitions use the TRAP-TYPE macro as
-- * defined in RFC1215.
-- *
-- * The community string is defined by commonTrapCommunityString.
-- *

hmsDownloadStatus  TRAP-TYPE
    ENTERPRISE scteHmsTree
    VARIABLES { commonPhysAddress, commonLogicalID, dlDownloadErrorStatus,
                dlDownloadImage, dlDownloadDevice }
```

```
    DESCRIPTION
        "A hmsDownloadStatus trap is generated when dlDownloadErrorStatus
        is set in response to an error."
    ::= 3
END
```