Programmer Manual

Tektronix

MTM400
MPEG Transport Stream Monitor

071-1375-00

This document supports MTM400 Programmer Interface MIB, Version 2.0 and above.

www.tektronix.com

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

MTM400 MPEG Transport Stream Monitor Programmer Manual
TekCamb Document: SI443 Issue 1.0j

SNMP Version 1.0

Software Warranty

Tektronix warrants that the media on which this software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If a medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which this software product is furnished, this software product is provided 'as is' without warranty of any kind, either express or implied. Tektronix does not warrant that the functions contained in this software product will meet Customer's requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for this software product and return this software product and any associated materials for credit or refund.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER'S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

Web Server URLs

Appendix A – Event Identities

List of Figures

Preface

This document specifies the MTM400 MPEG Transport Stream Monitor remote control and status monitoring interfaces available to a management application.

The manual is divided into the following sections:

Introduction
This section describes the SNMP and MIB interfaces.

MTM400 MIB
This section describes the implementation of the MIB.

MIB Groups
This section defines the groups of MIB modules that make up the MTM400 SNMP interface.

Web Server URLs
This section describes the URLs supported by the MTM400 Web Server.

This manual is supplied in Adobe PDF format on a CD-ROM. Also supplied on the CD is the MTM400 Programmer Interface MIB file in ASCII format, which can be loaded directly into any suitable MIB browser.

Related Documentation

- MTM400 MPEG Transport Stream Monitor User Manual
  (Tektronix Part Number: 071-1224-xx)
- STD-15 (RFC1157) Simple Network Management Protocol
  (http://www.iso.ch)
- STD-16 (RFC1155 and RFC1212)
  Structure and Identification of Management Information for TCP/IP-based Internets (http://www.iso.ch)
- The Simple Book
  Marshall T. Rose (Prentice Hall, ISBN 0-13-451659-1)

Related Material

The following URLs access the Web sites for the standards organizations listed (the URLs listed were valid at the time or writing):

- MPEG-2 standards (International Organization for Standards)
  http://www.iso.ch/
- DVB standards (European Technical Standards Institute)
  http://www.etsi.org/
- ATSC standards (Advanced Television Systems Committee)
  http://www.atsc.org/
- The Internet Engineering Task Force
  http://www.ietf.org

Contacting Tektronix

Phone 1-800-833-9200*

Address Tektronix, Inc.
Department or name (if known)
14200 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

Web site www.tektronix.com

Support - N & S America

Sales support 1-800-833-9200, select option 1*

Service support 1-800-833-9200, select option 2*

Technical support Email: techsupport@tektronix.com

1-800-833-9200, select option 3*
* - toll free in North America.

6:00 a.m. -- 5:00 p.m. Pacific time
After office hours, please leave a voice mail message.

Sales Support - Europe & the Rest of the World

Telephone: +44 (0)1344 392000

Service & Technical Support - Europe & the Rest of the World

Telephone +44 (0)1223 200700

Fax +44 (0)1223 200701

Introduction

This document specifies the MTM400 MPEG Transport Stream Monitor remote control and status monitoring interfaces available to a Management application. Two interfaces are provided; SNMP and an HTTP Web-based interface.

IMPORTANT NOTES:

The MTM400 Programmer Interface MIB file accompanying this document contains entries not described in the manual. These entries should not be used.

This document should be read in conjunction with the MTM400 User Manual. The reader must be thoroughly familiar with the operation of the MTM400 and have detailed knowledge of SNMP and HTTP.

Do not use multiple variable binding SET requests. Only single variable binding SET requests should be used.

SNMP and MIBs

This document specifies the facilities provided by the MTM400 Simple Network Management Protocol (SNMP) agent, which allows various parameters within the MTM400 monitor to be viewed and set. This will allow you to develop management applications that can control the MTM400 units across a network using SNMP

The MTM400 SNMP agent has been implemented as an extensible agent under Nucleus, and as such conforms to SNMP v1.

The Simple Network Management Protocol (SNMP) is an Internet standard protocol for remote management of entities on a network. It is defined in Internet documents STD-15 (RFC1157) and STD-16 (RFC1155 and RFC1212). STD-15 defines the protocol operations; STD-16 defines the way in which information is structured under SNMP (SMI - Structure of Management Information).

SNMP defines a way of structuring information in a hierarchy of objects supporting both single objects and tables of objects, and making the information available through a network protocol.

Each object can be one of four types, namely:

- **Integer.** Represents numerical values.
- **OctetString.** Represents byte streams.
- **DisplayString.** Represents printable strings.
- **Object Identifier** (OID). References other objects within SNMP.

There are essentially three types of operations that can be performed on each object:

- **Get.** Retrieves the value of an object.
- **GetNext.** Retrieves the value of an object along with the OID of the next object available.
- **Set.** Sets the value of an object.

The complete set of objects accessible through an SNMP agent is called the Management Information Base (MIB). The MIB is a tree structure with MIB objects at the leaves of the tree. Every branch and leaf of the tree is numbered according to a scheme ultimately under the administration of either ISO or the CCITT (or the ITU-T as they are now called). (The root of the tree has three branches: branch 0 is owned by the CCITT, 1 by ISO and 2 is jointly owned by ISO and the CCITT.) These organizations have delegated various branches of this tree to other authorities. Everything of interest to SNMP is under the control of the IANA (the Internet Assigned Numbers Authority), who owns the branch named:

iso (1).org (3).dod (6).internet (1)

The strings of numbers identifying parts of the MIB tree are called Object Identifiers (OIDs).

The Internet standard management sub-trees are all under

iso (1).org (3).dod (6).internet (1).mgmt (2).

However the IANA also allocates numbers to other organizations. Companies can obtain their own sub-trees under

iso (1).org (3).dod (6).internet (1).private (4).enterprises (1).

This entire tree structure is called the MIB. A MIB module is a set of sub-sections of this tree that form some coherent function or set of functions, usually described in a single document and qualified with some other title, such as RMON MIB.

NOTE. *A MIB module is sometimes referred to as the MIB.*

A MIB Module is defined in a text file using ASN.1 (Abstract Syntax Notation One). This is a language defined by OSI for describing arbitrary data structures. (In fact SNMP mandates that a fairly narrow subset of ASN.1.)

For more detailed explanations of network management using SNMP, you can refer to 'The Simple Book' (Marshall T. Rose, Prentice Hall, ISBN 0-13-451659-1).

MTM400 Web Server

The MTM400 has a Web server interface on HTTP port 80. A number of URL's are supported and are used primarily for transferring bulk data, unsuited to SNMP, to and from the MTM400. A full list of Web server URLs supported is given on page 61.

MTM400 MIB

Tektronix has been assigned the following root OID:

iso.org.dod.internet.private.enterprises.128

Under this OID Tektronix can define its own MIB for various products.

The MIB subtree for MTM400 is under the following OID

iso.org.dod.internet.private.enterprises.tek(128).tvt(5).tvtproducts(1).

It is specified in the two ASN.1 text files: ADSYS.MIB defines the structure of device specific elements and ADMPEG.MIB defines the structure of the MPEG Interface specific elements.

MIB Types

The MTM400 MIB defines the following extra MIB types.

EvId

This type defines events that can occur within the MTM400. It is essentially a WORD, where values 0x1xxx represent events that are generated by the MTM400 box such as Fan and Battery errors. Values over and including 0x2000 represent events that are generated by specific MPEG Interfaces such as Sync Lock or Continuity errors. The full list of these events can be found in *Appendix A – Event Identities*.

EvState

This type represents the state of a given event which can be 'Green', 'Yellow' or 'Red'. Green indicates that there is no error, yellow indicates that there has been an error since this event was last reset, and red indicates that there is a persistent error.

This is essentially a WORD. Green is defined to be 0x1000, yellow as 0x2000 and red as 0x3xxx, where xxx is the specific error number. A value of 0x0000 means the state is unknown, and 0x4000 means the event is disabled. Two final values are also possible: 0x5000 is the maintenance state and 0x6000 is N/A.

AlmValue

This specifies which alarms are activated when an event occurs. It is an integer type and can take combinations of the following values:

0x00000001 = Audible Alarm

0x00000100, 0x00000200, .. , 0x00001000= Relay1, Relay2, .., Relay 5

0x00010000, 0x00020000, .. , 0x00040000= TTL1, TTL2, TTL3

0x00100000, 0x00200000 = Send Trap on; Raise, Clear

Time Stamp

Time stamps are used in several MIB items to specify the time of events. Each time stamp is stored as an eight-byte structure, which consists of an 11-bit signed integer representing the UTC offset, and a 53 bit signed integer representing the UTC time. The UTC offset is the number of minutes that must be added to UTC time to obtain the local time on the MTM400 unit. The UTC time is the number of microseconds since midnight Greenwich Mean Time (GMT) January 1, 1970.

Figure 1 shows that the timestamp is actually stored with the UTC offset, followed by the UTC Time in MSB format. However, the bytes are reversed when the timestamp is presented as part of an Octet String through SNMP so that the numbers are in LSB format. Care should be taken with byte 6 because it contains both the UTC offset and UTC time.

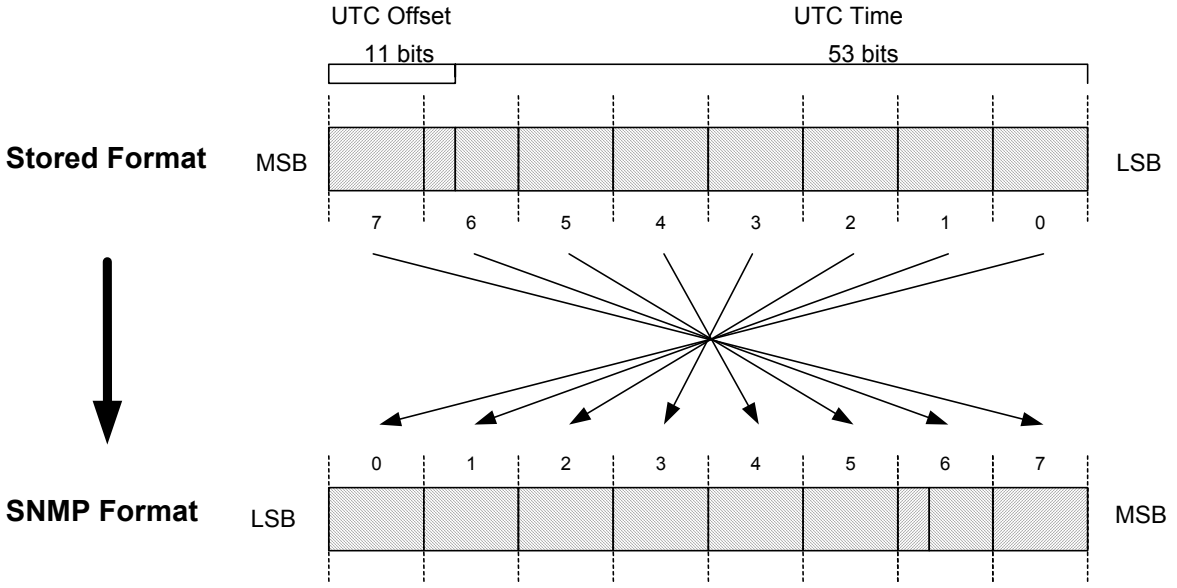

Figure 1: Time Stamp Storage

Accessing MIB Objects

This section describes how to access objects within the MTM400 MIB.

SNMP Access Operations

The MTM400 SNMP agent fully supports the standard SNMP GetRequest, GetNextRequest and SetRequest PDU operations. This document specifies the access permissions for each object within the MTM400 MIB using the following conventions:

- 'Get' indicates that the GetRequest and GetNextRequest can be used.
- 'Set' indicates that the SetRequest can be used.

Single Leaf Objects

Single Leaf Objects are single-value elements whose values can be accessed using the standard SNMP access operations by appending '0' to the appropriate OID specified in the MIB. For example, in order to access the program name within the System Information Group, use the following OID:

'…adsysProductName.0'.

Tables

The MTM400 MIB defines a number of tables. Tables normally contain objects that can have multiple values, each referenced by appending the required row number to the OID of the object specified in the MIB. Management applications typically access values of objects within tables by first performing a GetNextRequest-PDU on the OID object that will return the OID of the first value. Subsequent calls to the GetNextRequest operation will obtain the values for this object within the table. When the operation returns the 'No Such Name' error, this indicates that the last value has been reached.

Some tables within the MTM400 MIB are indexed by two or more values, so accessing object values becomes a little more complex. For example, the Event State Table is indexed by stream number and event id, so in order to reference a specific value, the OID should be created by appending the stream number and the event id to the OID specified for this object in the MIB. Consequently, in order to access the EventState for an event on a specific stream, use the following OID:

'…mivevtEventState.<interface_no>.<eventid>'.

The GetNextRequest-PDU operation will return the OID of the next eventid, until they have all been exhausted for that stream. At this point it will return the next interface_no, and the first event_id on that interface (or 'No Such Error' if no more interfaces exist to indicate that the end of the table has been reached).

When a table is defined within the MIB, each table leaf object is represented by the following OID:

'…<table_oid>.<table_entry_oid>.<table_leaf_object_oid>'.

The 'table_entry_oid's within the MTM400 MIB are always given the value 1, and are not shown on the structure charts within this document because it would only complicate the diagrams. However, it should be recognized that these must be included in the OIDs when referencing objects.

MIB Groups

The following sections define the groups of the MIB modules that make up the MTM400 SNMP interface. There is a split between MPEG-related and non-MPEG-related objects, and so the groups have been separated into two MIB modules. The System MIB module contains all non-MPEG-specific groups; MPEG-specific groups are found in the MPEG MIB module. Figure 2 to Figure 4 show the overall structure of the MTM400 MIB subtree.

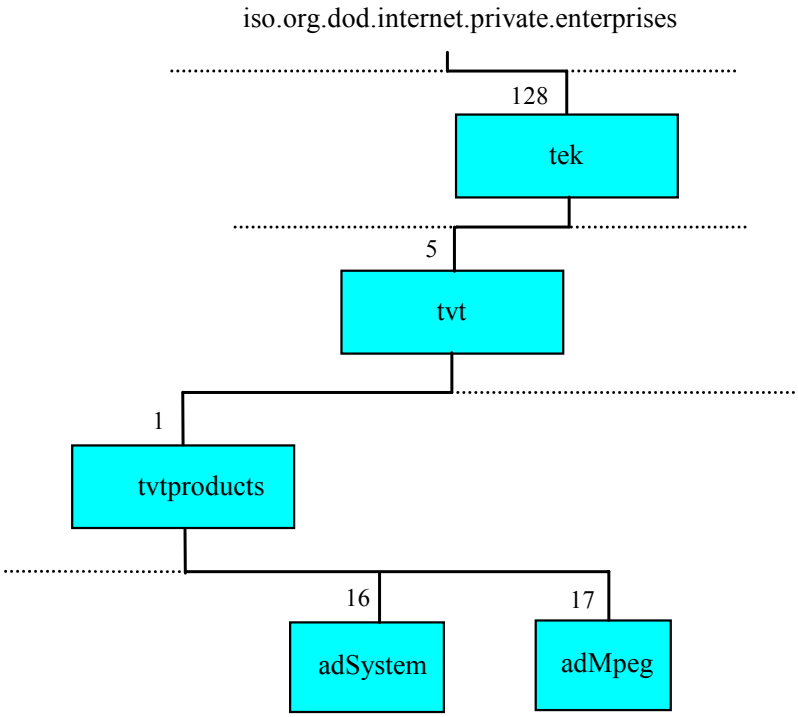

Figure 2: Overall MIB Structure

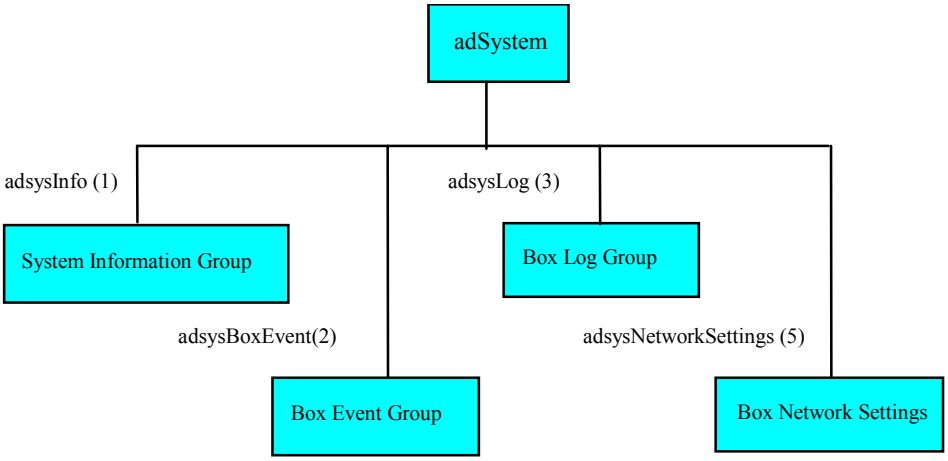

Figure 3: System Structure

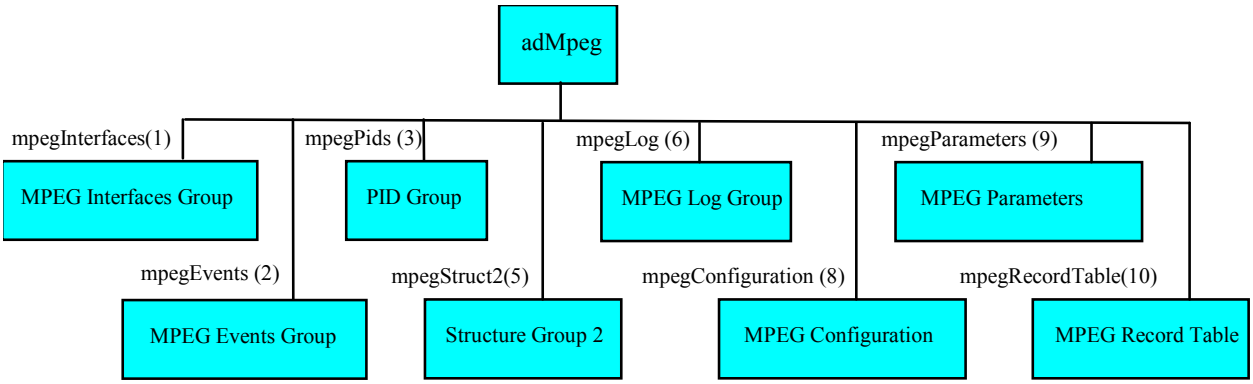

Figure 4: MPEG Structure

System Information Group

Figure 5 shows the structure of the System Information Group, which provides access to attributes of the most general nature, such as the product name and the software installed.

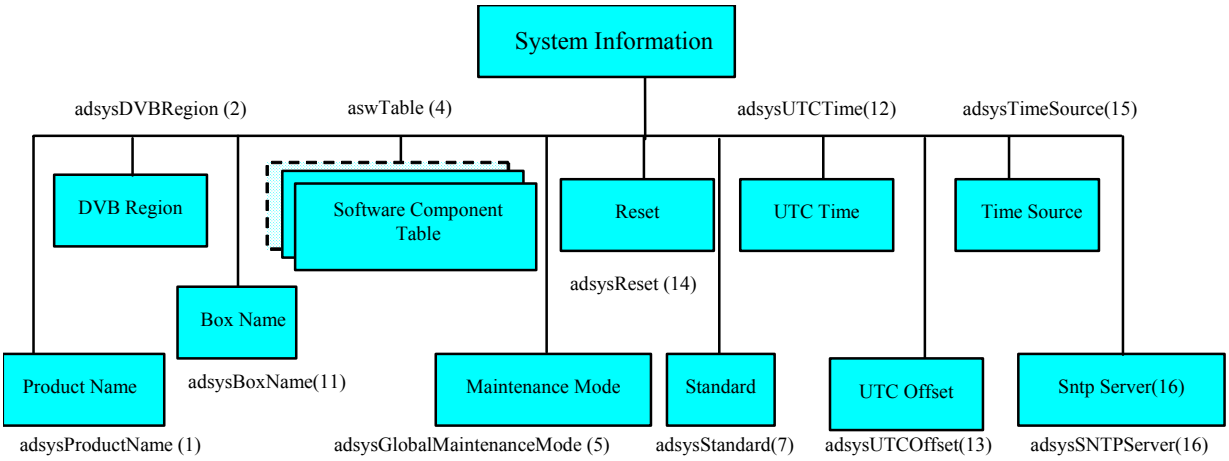

Figure 5: System Information Group Structure

Product Name

The Product Name contains the name of the product for this device. The format of this item is defined as:

Variable	Type	Use	Access
AdsysProductName (1)	DisplayString	The product name for this device.	Get

DVB Region

The DVB Region specifies the type of DVB streams the MTM400 has to monitor. The values are integers and have the following meanings:

0 = DVB (standard)

1 = DVB (DTG)

2 = DVB (Nordic)

3 = DVB (Reserved)

4 = DVB (Aus)

5 = Custom US satellite distributor

6 = DigicypherII

The format of this item is defined as:

Variable	Type	Use	Access
adsysDVBRegionalSetting (2)	INTEGER	The DVB region.	Get/Set

Software Component Table

The Software Component table contains information the following on the software used within the MTM400, and is defined as:

Variable	Type	Use	Access
aswIndex(1)	INTEGER	Table index	N/A
aswName(2)	DisplayString	The title for the software component in this table entry.	Get
aswVersion(3)	DisplayString	The version number for this software component.	Get

The table is indexed by an Integer, so for example in order to reference the name of the first software component, use the following OID: '…aswName'.

Standard

The standard specifies the MPEG Standard of the MTM400 unit. The format of this item is defined as:

Variable	Type	Use	Access
adsysStandard (7)	INTEGER	Specifies the MPEG Standard of the MTM400 unit 0 = DVB, 1 = ATSC, 2 = ISDB, 3 = China.	Get

Box Name

This value contains a configurable name for the box.

Variable	Type	Use	Access
adsysBoxName (11)	DisplayString	Contains a configurable name of the box.	Get/Set

UTC Time

The UTC time of the box i.e. the number of seconds since midnight 1st January 1970.

Variable	Type	Use	Access
adsysUTCTime (12)	INTEGER	The UTC time of the box.	Get/Set

UTC Offset

Number of minutes to add to UTC time to get to local time frame - this may be negative.

Variable	Type	Use	Access
adsysUTCOffset (13)	INTEGER	The UTC offset of the box.	Get/Set

Reset

Setting this value to a hex value DE5B12A resets the device

Variable	Type	Use	Access
adsysReset (14)	INTEGER	Device reset. Get has no meaning in this context.	Get/Set

Time Source

Specifies where the system derives the time from.

Variable	Type	Use	Access
adsysTimeSource (15)	INTEGER	0 = RTC (Real Time Clock on the device). 1 = LTC (Longitudinal Time Code). 2 = SNTP (Simple Network Time Protocol).	Get/Set

SNTP Server

The IP Address of an SNTP server.

Variable	Type	Use	Access
adsysSNTPServer (16)	IP Address	SNTP server IP Address.	Get/Set

Box Event Group

The MTM400 may generate several box-specific events. Normally, an event may be in one of five states:

- 'Red' (0x3xxx) indicates that there is currently an error condition.
- 'Yellow' (0x2000) indicates that there is currently no error condition, but that one has occurred since this event was last reset.
- 'Green' (0x1000) indicates that there is no error condition.
- 'Grey' (0x0000) indicates the state is unknown (or that the link is lost).
- 'White' (0x4000) indicates that the event is disabled.

Each event also has an alarm value associated with it, which indicates the type of alarm that will be triggered (such as audible or relay), if the event goes into error. The full list of box events is specified in *Appendix A – Event Identities*.

The following diagram shows the structure of the Box Event Group, which contains information on the states and alarm values for all box events that can be generated by the MTM400.

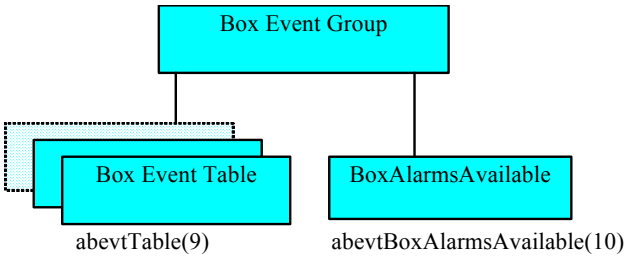

Figure 6: Box Event Group Structure

The following table describes the single leaf objects within the Box Event Group.

Variable	Type	Use	Access
abevtBoxAlarmsAvailable(10)	AlmValue	The result of 'ORing' the types of alarms that can be triggered for box events. This is determined by the hardware available on the addressed box.	Get

Box Event Table

The box alarm table contains the state and alarm value for each box-wide event as specified in *Appendix A – Event Identities* and is defined as:

Variable	Type	Use	Access
abevtIndex(1)	EvId	An index identifying the event id as defined in Appendix A.	N/A
abevtEventName(2)	DisplayString	A short name for this event.	Get
abevtEventDescription(3)	DisplayString	A brief description of the meaning of this alarm	Get
abevtEventState(4)	EvState	The state of this event.	Get/Set
abevtAlarmValue(5)	AlmValue	The alarms that will be triggered for this event.	Get/Set
abevtEventEnable(6)	INTEGER	Specifies whether the event is enabled (0 = disabled, 1 = enabled).	Get/Set

Indexing. The table is indexed by EvId; for example in order to reference the name of event 0x1000 (4096), use the following OID

'…abevtEventName.4096'.

Descriptions. An event name and description are included in this table. This is so that management applications using this MIB can report all events. (This table has been designed so that new event types can be added later. A management application could display all of the event types it knows about in a predetermined manner, but still be able to display events added after it was written.) These textual MIB variables would typically be downloaded once when the management application starts, or not at all if you only want to display some particular fixed set of events.

Unsupported Events. Box events that are not supported for this MTM400 unit will have an event state of 0x0000.

Setting Event States. Setting an event state for any value that is in the 'Yellow' (0x2000) state resets the event. Setting an event with a 'Red' state has no effect, because this indicates that there is a persistent error.

Setting Alarm Values. An alarm value specifies which alarms will be triggered when the corresponding event indicates an error. A value is a combination of those specified AlmValue (page 6), for example, 0x00020401 will set TTL2, Relay3 and Audible alarms to be triggered.

Box Log Group

Figure 7 shows the structure of the Box Log Group, which provides access to the box specific log items.

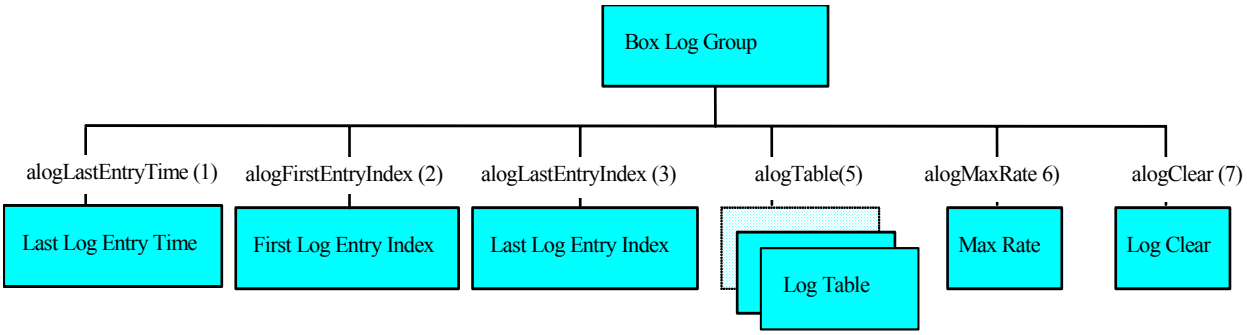

Figure 7: Box Log Group Structure

The following table describes the single leaf objects within the Box Log Group. It should be noted that some of these values also apply to the stream logs.

Variable	Type	Use	Access
alogLastEntryTime(1)	TimeTicks	The value of sysUpTime at which the most recent entry was added to a box log or any stream log.	Get
alogFirstEntryIndex(2)	INTEGER	The index of the oldest box log entry.	Get
alogLastEntryIndex(3)	INTEGER	The index of the most recent box log entry	Get
alogMaxRate(6)	INTEGER	This sets the maximum number of entries that will be logged (per second) for both box and stream logs. A value of 0 disables logging, and a value of 10000 specifies that there is no maximum limit.	Get/Set
alogClear(7)	INTEGER	Setting this value clears the box and stream logs.	Set

The first and last entry indices can be used to access the required elements from the Log Table, which is shown in the following table.

Log Table

The log entry table contains information on the event log generated by the MTM400, and is defined as:

Variable	Type	Use	Access
alogIndex(1)	INTEGER	The index for the log entry.	N/A
alogText (2)	OCTET STRING	The OCTET STRING contains a coded representation of the log entry.	Get

Indexing. The table index is an integer, so it may wrap around if the number of entries in the log becomes very large. This implies that the element with the largest index is not necessarily the latest log entry. The index of the last entry can be obtained from the single leaf element alogLastEntryIndex. In order to obtain the required log text from the table, use the following OID

'…alogText.<index>'.

Log Text Formatting. The alogText will be empty if the index requested is not valid. This occurs if the management application requests an entry that no longer exists, for example, if the the log was full and the entry was deleted from the end of the list to make room for new entries. If the log is being filled rapidly, the index returned from alogFirstEntryIndex is likely to be invalid for a call to alogText.

If alogText is not empty, the format of the octet string is as follows:

Bytes 0..7 : The timestamp of the event as defined in Time Stamp, page 6.

Bytes 8..9 : The source of the event (stream number least significant bit (LSB) first)

Bytes 10..11 : The ID of the event (LSB first)

Bytes 12..13 : Extension ID (LSB first)

Bytes 14..15 : The state of the event (LSB first)

Bytes 16 - onwards : Textual Description (UTF8, not NULL terminated).

Network Settings Table

The network settings table provides information on the device's network settings. The information available is defined as:

Variable	Type	Use	Access
aNetIpAddress(1)	IP Address	The IP Address of the device.	Get/Set
aNetGatewayAddress (2)	IP Address	The IP Address of the gateway for the device.	Get/Set
aNetSubnetMask (3)	IP Address	The subnet mask.	Get/Set
aNetSubnetMask (3)	IP Address	The subnet mask.	Get/Set
aNetCommunityRead (4)	DisplayString	Alternate SNMP community string used to read.	Get/Set
aNetCommunityWrite (5)	DisplayString	Alternate SNMP community string used to write.	Get/Set
aNetCommunityTrap (6)	DisplayString	SNMP target community for all traps.	Get/Set

The read and write community strings in this table are alternates to support management systems with fixed communities. The default 'public' community will always work.

Changing the network information will have no effect until the MTM400 is reset.

License Table

The network settings table provides information on the device's network settings. The information available is defined as:

Variable	Type	Use	Access
alicCapabilities (1)	OCTET STRING	The licensed capabilities of the device.	Get

Licensed Capabilities

This field is an octet string containing a variable length bitfield enumerating the capabilities of the unit.

The current bit definitions are:

0	Structure View	10	Pid Variability
1	Repetition Graphs	11	Scheduling
2	Bitrate Limits	12	(Reserved)
3	Pid Groups	13	TMCC
4	Templates	14	RF Analysis
5	Template Tree View	15	Full I/O
6	Recording	16	Reduced I/O
7	PCR Graphs	17	QAM A Select *
8	SFN	18	QAM B Select *
9	Service Log	19	QAM C Select *

* - applicable to common design cards only

Each octet has bits numbered from zero for the least significant, to seven for the most significant. The first octet contains the values 0..7, the second contains 8..15, and so on up to the number of required octets.

MPEG Interfaces Group

Figure 8 shows the structure of the MPEG Interfaces Group, which contains information on each of the MPEG Interfaces connected to the MTM400 unit. The terms 'Stream' and 'Interface' are used interchangeably.

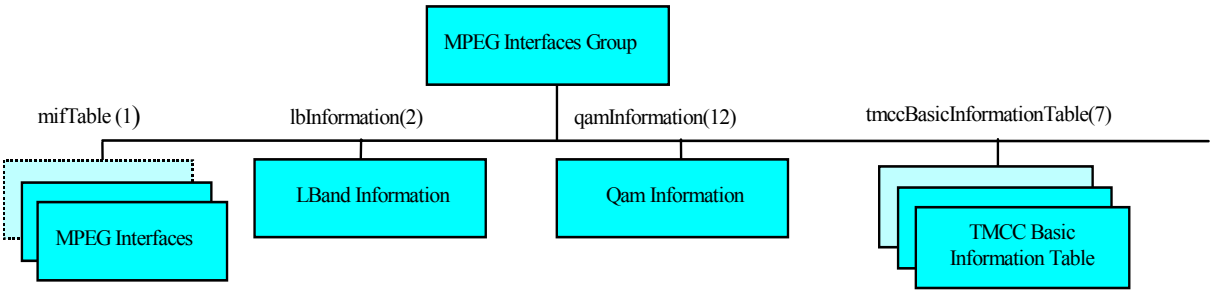

Figure 8: MPEG Interfaces Group Structure

MPEG Interfaces Table

The MPEG Interfaces table is similar in concept to the Interfaces Group (ifTable) defined in MIB-II (RFC1213), which provides a list of all network interfaces installed in a device supporting network management. As with the ifTable, it allows a common network management mechanism to be used to describe and control MPEG interfaces regardless of the application. Also as with the ifTable, the indices into the MPEG Interface Table can be used as cross references from other MIB modules, or even as indices for other tables, enabling these to extend the MPEG Interface table with application-specific information.

The table is defined as:

Variable	Type	Use	Access
mifIndex(1)	INTEGER	The MPEG Interface for which these readings apply. These are used to identify MPEG interfaces elsewhere in the MIB.	N/A
mifAvailableInterface(6)	INTEGER	The interface available (see below).	Get
mifActualMpegPacketSize(7)	INTEGER	The actual MPEG Packet size on this interface. This will be 0, 188, 204 or 208 where 0 indicates unknown.	Get
mifTransportStreamBitRate(10)	INTEGER	Transport rate of the stream in bits/s.	Get
mifNoPids(11)	INTEGER	Number of PIDs in the stream with non-zero bit rate and those that have had limits set.	Get
mifStreamName(12)	DisplayString	Configurable name for the stream.	Get/Set
mifChosenInterfaceType(13)	INTEGER	Specifies the type of interface to use for the stream (see *Available Interface*, page 25). Where more than one interface exists with this type, the first one is chosen by default. This can be changed with the 'migChosenInterfaceIndex' item.	Get/Set
mifChosenInterfaceIndex(14)	INTEGER	Specifies which interface of the type specified by mifChosenInterfaceType is to be used.	Get/Set

Variable	Type	Use	Access
mifPCRAccuracyMode(15)	INTEGER	Specifies algorithm to use of PCR Accuracy calculations. 0 = Previous Point Differential, 1 = Line Fit Differential.	Get/Set
mifStreamMaintenanceMode (16)	INTEGER	Specifies whether the stream is in maintenance mode. 0 = off, 1 = on.	Get/Set
mifDVBRegion(17)	INTEGER	Specifies the DVB Region of the stream. 0 = DVB, 1 = DTG, 2 = Nordic, 3 = ISDB, 4 = Aus, 5 = Custom US satellite distributor, 6 = DigiCypherII.	Get/Set
mifReset(18)	INTEGER	Setting this value has the effect of resetting the stream parameters to the factory defaults.	Set
mifStandard(19)	INTEGER	Specifies the MPEG Standard for the stream. 0 = DVB, 1 = ATSC, 2 = ISDB, 3 = China.	Get/Set
mifSchedulerEnabled(21)	INTEGER	Specifies whether scheduler is enabled. 0 = disabled, 1 = enabled.	Get/Set
mifScheduleName (23)	INTEGER	The name of the schedule file currently loaded.	Get

Indexing. The table is indexed by interface number, for example to reference the name for interface 1, use the following OID:

'…mifName.1'.

Available Interface. This field indicates which, if any, of the supported interface cards are connected to the MTM400 via the serializer port. The interpretation of the mifAvailableInterface values is as follows:

0 = Unknown

0x0800 = QAM_ANNEX_A

0x2800 = QAM_ANNEX_B

0x1800 = QAM_ANNEX_C

0x3800 = LBAND_II

0x6800 = GPSI_2

0xE000 = ASI

Standard and Region

There are a number of standards, the region field meaning depends on the standard chosen. For DVB, this field denotes a region, in other cases it is a specialization.

Standard	Region
DVB (0)	Std (0)
	DTG (1)
	Nordic (2)
	Reserved (3) (was ARIB)
	Aus (4)
ATSC (1)	Standard (0)
ISDB (2)	ISDB-S (0) (Japanese standard)
	ISDB-T (1) (Japanese standard)
Chinese (3)	GY/Z 174-2001 (0) (DVB with explicit GB2312 content)
	GB2312 (1) (DVB with implicit GB2312 content)

LBand Information Group

The following diagram shows the structure of the LBand Information Group, which contains information on the LBand Settings where appropriate.

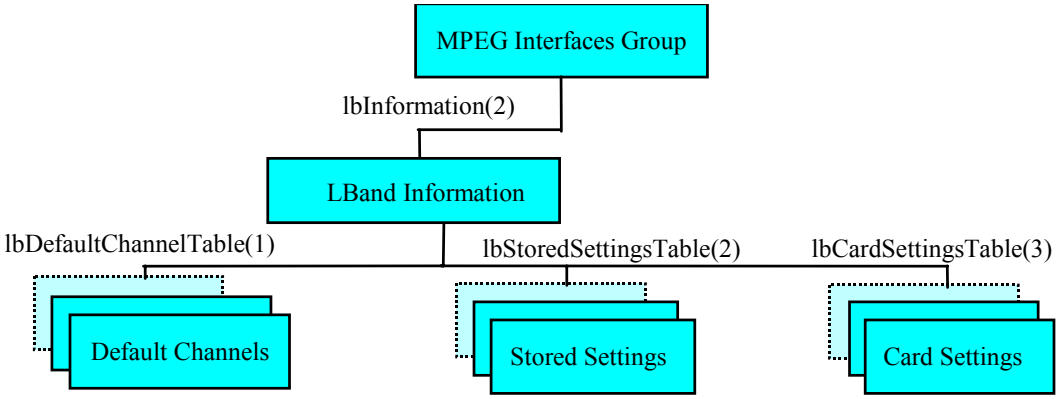

Figure 1-9: L-Band Information Group Structure

Default Channels Table. The default channels table contains the name of the stored LBand Settings to use for each interface, and is defined as:

Variable	Type	Use	Access
LbDefaultChannelmifIndex(1)	INTEGER	The MPEG interface for which this default channel applies.	N/A
LbDefaultChannelName(2)	DisplayString	The name of the selected stored settings channel. This is used to reference the required entry in the *Stored Settings* table (see following table).	Get/Set

The table is indexed by MPEG Interface, so in order to determine the name of the stored LBand settings for interface 1, use the following OID:

'…lbDefaultChannelName.1'.

Stored Settings Table. The stored settings table contains the available stored LBand settings that can be used for each interface, and is defined as:

Variable	Type	Use	Access
LbStoredmifIndex(1)	INTEGER	The MPEG interface for which these stored settings apply.	N/A
lbStoredChannelIndex(2)	INTEGER	Index to the stored channel settings used for this interface.	N/A
lbStoredName(3)	DisplayString	The name given to these stored settings.	Get
lbStoredLoFreq(5)	INTEGER	Local Oscillator Frequency in MHz.	Get/Set
ldStoredTrFreq(6)	INTEGER	Transponder Frequency in MHz.	Get/Set
lbStoredPolarization(7)	INTEGER	Polarization (Volts) 0 = 0, 1 = 13(V), 2 = 18(H)	Get/Set
lbStoredSymRate(8)	INTEGER	Symbol Rate	Get/Set
lbStoredViterbiRate(9)	INTEGER	0 = 1/2 , 1 = 2/3, 2 = 3/4, 3 = 4/5, 4 =5/6, 5 = 6/7	Get/Set
lbStoredViterbiRateAuto(10)	INTEGER	Sets whether ViterbiRateAuto is on (0 = off, 1 = on).	Get/Set
lbStoredTone22K(11)	INTEGER	Sets whether 22KHz tone is on (0 = off, 1 = on).	Get/Set
lbStoredInvertSpectrum(12)	INTEGER	Sets whether invert spectrum is set (0 = off, 1 = on).	Get/Set

The table is indexed by MPEG Interface followed by Channel Index. The stored LBand settings are persistent across all interfaces, so the Channel Index is used to reference which settings should be used from this global list. This has the consequence that if any of these values are changed on one interface, it will be changed across all interfaces. As an example, in order to reference the Transponder Frequency for interface 1, channel 2, use the following OID:

'….lbStoredTrFreq.1.2'.

Card Settings Table. The card settings table contains the current settings for the LBand card, and is defined as:

Variable	Type	Use	Access
lbCardmifIndex(1)	INTEGER	The MPEG interface for which these card settings apply.	N/A
lbCardValidSettings(2)	INTEGER	Determines whether the LBand settings for this interface are valid (if this interface supports an LBand card: 1 = true, 0 = false).	Get/Set
lbCardLoFreq(4)	INTEGER	Local Oscillator Frequency in MHz	Get/Set
ldCardTrFreq(5)	INTEGER	Transponder Frequency in MHz	Get/Set
lbCardPolarization(6)	INTEGER	Polarization (Volts) 0 = 0, 1 = 13(V), 2 = 18(H)	Get/Set
lbCardSymRate(7)	INTEGER	Symbol Rate	Get/Set
lbCardViterbiRate(8)	INTEGER	0 = 1/2 , 1 = 2/3, 2 = 3/4, 3 =5/6, 4 = 6/7, 5 = 7/8	Get/Set
lbCardViterbiRateAuto(9)	INTEGER	Sets whether ViterbiRateAuto is on (0 = off, 1 = on)	Get/Set
lbCardTone22K(10)	INTEGER	Sets whether 22KHz tone is on (0 = off, 1 = on)	Get/Set
lbCardFrontEndLock(11)	INTEGER	Determines whether Front End Lock is on	Get
lbCardBER(12)	INTEGER	The BER. See below for specific values.	Get
lbCardInvertSpectrum (13)	INTEGER	Sets whether invert spectrum is set (0 = off, 1 = on).	Get/Set
lbCardMER(14)	INTEGER	MER db * 10^6	Get
lbCardActualBER(15)	INTEGER	BER Ratio * 10^9	Get
lbCardEVM(16)	INTEGER	EVM % * 10^6	Get
lbCardTEFCount(17)	INTEGER	TEF count	Get
lbCardSignal(18)	INTEGER	Signal Strength % * 10^6	Get

The table is indexed on MPEG Interface. As an example, in order to reference the Viterbi Rate for interface 1, use the following OID:

'….lbCardViterbiRate.1'.

The BER values returned have the following meanings:

{1.0e-1, 1},	{1.3e-2, 11},	{6.0e-5, 21},
{9.0e-2, 2},	{1.0e-2, 12},	{3.0e-5, 22},
{8.0e-2, 3},	{7.0e-3, 13},	{1.0e-5, 23},
{7.0e-2, 4},	{5.5e-3, 14},	{4.0e-6, 24},
{6.0e-2, 5},	{3.0e-3, 15},	{1.0e-6, 25},
{5.0e-2, 6},	{1.5e-3, 16},	{1.0e-7, 26},
{4.0e-2, 7},	{1.0e-3, 17},	{1.0e-8, 27},
{3.0e-2, 8},	{5.5e-4, 18},	{1.0e-9, 28}
{2.5e-2, 9},	{3.0e-4, 19},	
{1.7e-2, 10},	{1.5e-4, 20},	

QAM Information Group

Figure 10 shows the structure of the QAM Information Group, which contains information on the QAM Settings where appropriate.

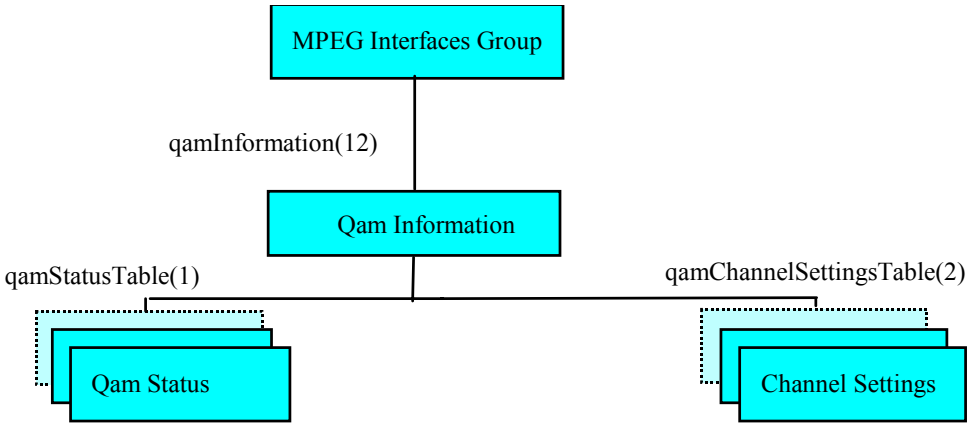

Figure 10: QAM Information Group Structure

Status Table. The status table contains the name of the selected channel settings and the status of the QAM card. The table is defined as:

Variable	Type	Use	Access
qamStatusmifIndex (1)	INTEGER	The MPEG interface for which this channel applies.	N/A
qamCurrentChannelName (2)	DisplayString	The name of the selected stored channel. This is used to reference the required entry in the QAM Channel Settings table (see following table).	Get/Set
qamFrontEndLock(3)	INTEGER	Boolean indicating the state of the front end lock (0 – no lock, 1 – in lock).	Get
qamSignalStrength (4)	INTEGER	The signal strength 0->255.	Get
qamBER (5)	INTEGER	The BER. (See BER values below for specific values.)	Get
qamMER(6)	INTEGER	MER db * 10^6.	Get
qamActualBER(7)	INTEGER	BER Ratio * 10^9.	Get
qamEVM(8)	INTEGER	EVM % * 10^6.	Get
qamTEFCount(9)	INTEGER	TEF count.	Get
qamSignal(10)	INTEGER	Signal Strength % * 10^6.	Get

The table is indexed by MPEG Interface, so in order to determine the name of the stored QAM channel settings for interface 1, use the following OID:

'…qamCurrentChannelName.1'.

The BER values returned have the following meanings:

{1.0e-1, 1},
{9.0e-2, 2},
{8.0e-2, 3},
{7.0e-2, 4},
{6.0e-2, 5},
{5.0e-2, 6},
{4.0e-2, 7},
{3.0e-2, 8},
{2.5e-2, 9},
{1.7e-2, 10},
{1.3e-2, 11},
{1.0e-2, 12},
{7.0e-3, 13},
{5.5e-3, 14},
{3.0e-3, 15},
{1.5e-3, 16},
{1.0e-3, 17},
{5.5e-4, 18},
{3.0e-4, 19},
{1.5e-4, 20},
{6.0e-5, 21},
{3.0e-5, 22},
{1.0e-5, 23},
{4.0e-6, 24},
{1.0e-6, 25},
{1.0e-7, 26},
{1.0e-8, 27},
{1.0e-9, 28}

QAM Channel Settings Table. The QAM channel settings table contains the stored QAM settings that can be used for each interface, and is defined as:

Variable	Type	Use	Access
qamChannelSettingsmifIndex (1)	INTEGER	The MPEG interface for which these channel settings apply.	N/A
qamChannelIndex (2)	INTEGER	Index to the stored channel settings used for this interface.	N/A
qamChannelName (3)	DisplayString	The name given to these channel settings.	Get
qamChannelRxFreq (4)	INTEGER	The rx frequency of the channel in Hz.	Get/Set
qamChannelSymRate (5)	INTEGER	The symbol rate of the channel.	Get/Set
qamChannel2LoFreq (6)	INTEGER	The 2nd Local Oscillator frequency of the channel in Hz.	Get/Set
qamChannelConstellation (7)	INTEGER	The constellation (trellis patters) of the channel. The MIB value is mapped to the constellation as follows : 0 = 4 1 = 16 2 = 64 3 = 256	Get/Set
qamChannelInversion (8)	INTEGER	Specifies inversion for the channel. 0 = not inverted, 1 = inverted.	Get/Set
qamChannelVControl (9)	INTEGER	Specifies V Control for the channel. 0 = off, 1 = on.	Get/Set
QamChannelLockConfidence (11)	INTEGER	Reserved.	Get/Set
QamChannelCorrectionConfidence (12)	INTEGER	Reserved.	Get/Set

The table is indexed by MPEG Interface followed by Channel Index. The QAM channel settings are persistent across all interfaces, so the Channel Index is used to reference which settings should be used from this global list. This has the consequence that if any of these values are changed on one interface, it will be changed across all interfaces.

TMCC Basic Information Table

The TMCC Basic Information Table contains the information stored in the first eight bytes of TMCC blocks for each interface. In order for the MTM400 to process the TMCC information, tmccAcquisition must be set to 1 for the appropriate stream.

Variable	Type	Use	Access
tmccmifIndex(1)	INTEGER	The MPEG interface for which these settings apply.	N/A
tmccAcquisition(2)	INTEGER	Specifies whether to process TMCC information. 0 = No, 1 = Yes.	Get/Set
tmccBufferReset(3)	INTEGER	1 = buffer reset, 0 = buffer not reset.	Get
tmccEmergencySignal(4)	INTEGER	1 = emergency signal, 0 = no emergency signal.	Get
tmccChangeIndication(5)	INTEGER	1 = change, 0 = no change.	Get
tmccBeginningOfFrame(6)	INTEGER	1 = beginning of frame, 0 = not beginning of frame.	Get
tmccBeginningOfSuperFrame(7)	INTEGER	1 = beginning of super frame, 0 = not beginning of super frame.	Get
tmccTransmissionMode1(8)	DisplayString	First transmission mode description.	Get
tmccSlotAllocation1 (9)	INTEGER	First slot allocation.	Get
tmccTransmissionMode2(10)	DisplayString	Second transmission mode description.	Get
tmccSlotAllocation2(11)	INTEGER	Second slot allocation.	Get
tmccTransmissionMode3(12)	DisplayString	Third transmission mode description.	Get
tmccSlotAllocation3(13)	INTEGER	Third slot allocation.	Get
tmccTransmissionMode4(14)	DisplayString	Fourth transmission mode description.	Get
tmccSlotAllocation4(15)	INTEGER	Fourth slot allocation.	Get
tmccTransportID(16)	INTEGER	The transport ID.	Get
tmccRawBytes(17)	Display String	8 Bytes of TMCC raw.	Get

The table is indexed by MPEG Interface. As an example, in order to reference the Emergency Signal for interface 1, use the following OID

'…tmccEmergencySignal.1'.

MPEG Events Group

The MTM400 may generate several events for each MPEG interface. Normally, an event may be in one of five states:

- 'Red' (0x3xxx) indicates that there is currently an error condition.
- 'Yellow' (0x2000) indicates that there is currently no error condition, but that one has occurred since this event was last reset.
- 'Green' (0x1000) indicates that there is no error condition.
- 'Grey' (0x0000) indicates the state is unknown (or link lost).
- 'White' (0x4000) indicates that the event is disabled.

Each event also has an alarm value associated with it, which indicates the type of alarm that will be triggered (for example, audible or relay), if an error occurs. The full list of box events is specified in Appendix A – Event Identities.

Figure 11 shows the structure of the MPEG Events Group, which contains information on the states and alarm values of events on each MPEG Interface.

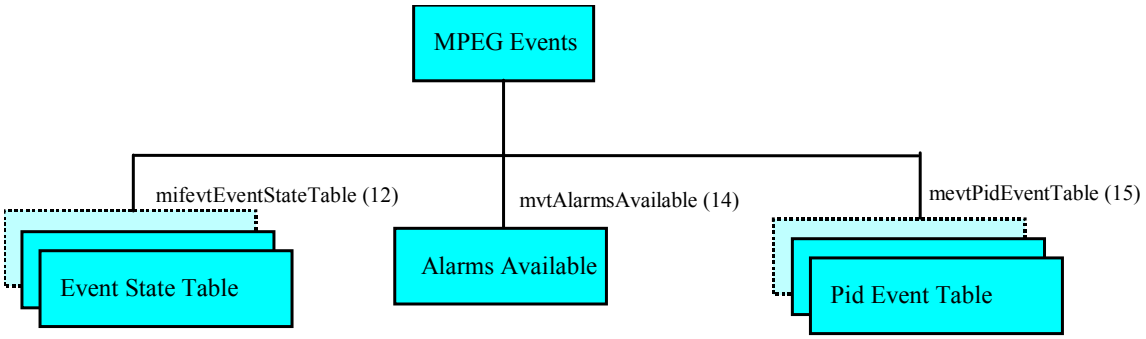

Figure 11: MPEG Events Group Structure

Single Leaf Objects. The following table describes the single leaf objects within the MPEG Events Group.

Variable	Type	Use	Access
mevtAlarmsAvailable(14)	INTEGER	Indicates the types of alarms that can be triggered for stream events.	Get

Event State Table. The Event State Table contains the state of each MPEG Interface event on every interface, and is defined as:

Variable	Type	Use	Access
mifevtMifIndex(1)	INTEGER	The MPEG interface for which these events apply.	N/A
mifevtEventIndex(2)	EvId	An index uniquely identifying the event.	N/A
mifevtEventName(3)	DisplayString	A short textual title for this event.	Get
mifevtEventDescription(4)	DisplayString	A brief description of this event.	Get/Set
mifevtEventState(5)	EvState	The state of this event. Writing any value will reset the event. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set
mifevtAlarmValue(6)	AlmValue	The alarms that will be triggered for this event.	Get/Set
mifevtEventEnable(7)	INTEGER	Specifies whether this event is enabled (0 = disabled, 1 = enabled).	Get/Set

Indexing. The table is indexed by MPEG Interface followed by EvId. As an example, in order to reference the alarm value of event 0x2000 (8192) on interface 1, use the following OID '…mifevtAlarmValue.1.8192'.

Unsupported Events. Events that are not supported on an interface will have an event state of 0x0000.

Setting Event States. Setting an event that is in the 'Yellow' (0x2000), to any value, resets the event. Setting an event with a 'Red' state has no effect, because this indicates that there is a persistent error.

Setting Alarm Values. An alarm value specifies which alarms will be triggered when an error occurs in the corresponding event.
The value is a combination of those specified in *AlmValue*, page 6 (for example, 0x00020401 will set TTL2, Relay3 and Audible alarms to be triggered).

PID Event Table. The PID Event Table contains a table of MPEG PID specific events on every interface, and is defined as:

Variable	Type	Use	Access
mevtPidMifIndex (1)	INTEGER	The MPEG interface for which these events apply.	N/A
mevtPidEventIndex (2)	INTEGER	A unique index identifying a particular type of PID event. The values for this index are prescribed, and can be found in Appendix A of the MIB Specification.	N/A
mevtPidPidIndex (3)	INTEGER	The PID number + 1.	N/A
mevtPidEventState (4)	INTEGER	Reading this returns the current event status for the PID as described for the EvState type. Writing any value will reset the event. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set
mevtPidEventEnable (5)	INTEGER	If a per PID event is disabled, the EvState will always be reported as 'disabled', no alarms will be generated for the event and the system does not need to perform any processing associated with the event.	Get/Set

MPEG PIDs Group

Figure 12 shows the structure of the PIDs Group, which contains PID, PID Group and Program limit and rate information:

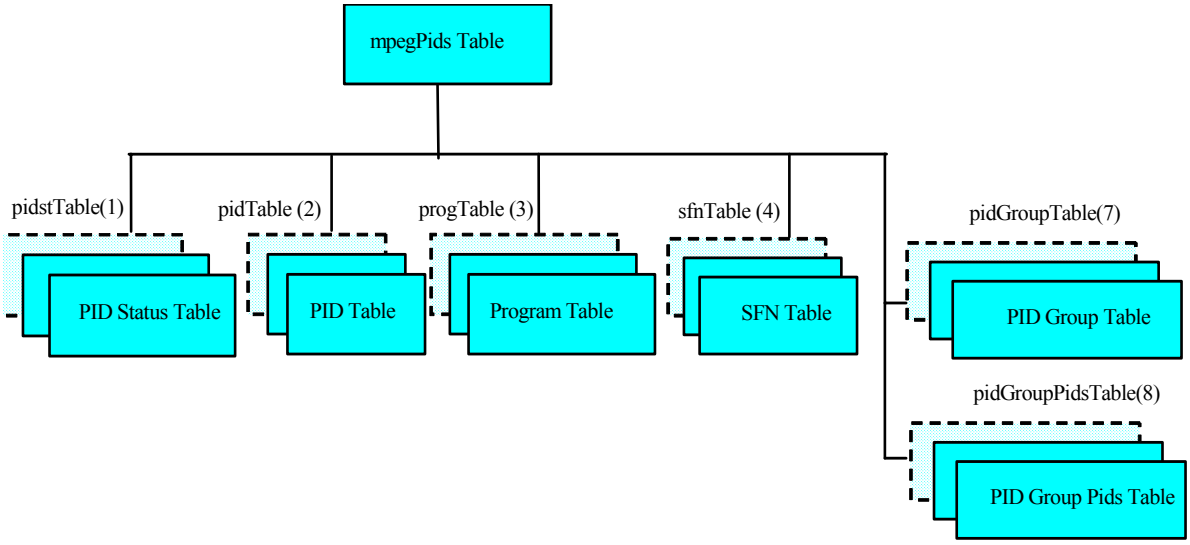

Figure 12: MPEG PIDs Group Structure

PID Status Table. The PID Status Table contains PID and Program status information for each interface, and is defined as:

Variable	Type	Use	Access
pidstMifIndex(1)	INTEGER	MPEG interface for which these elements apply.	N/A
pidstClearLimits (2)	INTEGER	Setting this clears all the PID rate limits for this interface. Reading this value has no meaning.	Set
progstClearLimits (3)	INTEGER	As above, but for programs.	Set
pidstResetRates(4)	INTEGER	Setting this resets all PID minimum and maximum rate measurements for this interface. Reading this value has no meaning.	Set
progstResetRates(5)	INTEGER	As above, but for programs.	Set
pidgroupstClearLimits (10)	INTEGER	Setting this clears all the PID group rate limits for this interface. Reading this value has no meaning.	Set
pidgroupstResetRates (11)	INTEGER	Writing any value to this object will reset the currently latched minimum and maximum bit rates for all PID groups. Reading this value has no meaning.	Set
pidgroupstNewPidGroupIndex(12)	INTEGER	Reading this will create a new pid group on the MTM400 device. The value returned is the group index. This is used to index this group in the pidGroupTable and pidGroupPidsTable.	Set
pidgroupstDeletePidGroupIndex (13)	INTEGER	Writing a value will delete the group with the index specified by the value set.	Set

The table is indexed by MPEG Interface. As an example, in order to reference pidsResetRates for interface 1, use the following OID:

'…pidstResetRates.1'.

PID Table. The PID Table contains information for each PID on each interface, and is defined as:

Variable	Type	Use	Access
pidsMifIndex(1)	INTEGER	The MPEG interface for which these readings apply.	N/A
pidsPidIndex(2)	INTEGER	The PID index - this is the PID number + 1 to avoid a 0 index.	N/A
pidsRate(3)	INTEGER	The most recently measured rate for this PID.	Get
pidsMinRate(4)	INTEGER	The minimum rate latched for this PID since last reset.	Get
pidsMaxRate(5)	INTEGER	The maximum rate latched for this PID since last reset.	Get
pidsMinLimit(6)	INTEGER	The minimum limit for this PID.	Get/Set
pidsMaxLimit(7)	INTEGER	The maximum limit for this PID.	Get/Set
pidsState(8)	EvState	The state of this PID.	Get
pidsScrambled (11)	INTEGER	0 = PID not scrambled, 1 = PID scrambled.	Get
pidsUnreferenced(13)	INTEGER	Indicates whether the PID is un-referenced. 1 = un-referenced, 0 = referenced.	Get
pidsForceListPresence (15)	INTEGER	Specifies whether the PID must exist in this list, event if it does not appear in the transport stream. 0 = PID not present, 1 = PID present.	Get/Set
pidsVariability (16)	OCTET STRING	Textual representation of variability (floating point number).	Get

- **Indexing.** The table is indexed by MPEG Interface, followed by PID Index. As an index of 0 is not allowed in SNMP tables, the PID Index is actually PID+1. Therefore, in order to reference the required PID item, for example pidsMinLimit, use the following OID:

  '…pidsMinLimit.<interface>.<pid+1>'.

- **Reading PID Information.** The list of PIDs for which readings are available can change fairly rapidly, so the management application must be notified that subsequent requests for PID elements may result in values for different set of PIDs. Consequently, if a client application requests all of the pidsMinRates followed by pidsMaxRates, it is not guaranteed that the values obtained will be for exactly the same set of PIDs. Therefore, in order to force the agent to include a PID in its list, the management application should set the corresponding Min and Max limits.

- **PID Limits.** By default, the limits for each PID are not defined; this is represented by the pidsMinLimit and pidsMaxLimit values being set to 0 and -1 respectively. When setting a limit, the management application must ensure that the value of pidsMaxLimit is always greater than pidsMinLimit, otherwise the new setting will not be accepted by the MTM400. The new and current values of pidsMinLimit and pidsMaxLimit will therefore affect the order in which the management application sets these limits.

  The limits for a PID can be cancelled at a later date by setting the pidsMinLimit to 0 and then setting pidsMaxLimit to -1. Although -1 is less than 0, this is a special case, which is accepted by the MTM400.

- **PID Occupancy Events.** The MPEG Interface event 0x2001 will be generated whenever any PID occupancy exceeds its limits. The management application can choose to poll this at the required interval.

Program Table. The following are the objects in the program table in the PID group:

Variable	Type	Use	Access
progsMifIndex(1)	INTEGER	The MPEG interface for which these readings apply.	N/A
progsProgIndex(2)	INTEGER	The program index - this is the program number + 1 to avoid a 0 index.	N/A
progsRate(3)	INTEGER	The most recently measured rate for this program.	Get
progsMinRate(4)	INTEGER	The minimum rate latched for this program since last reset.	Get
progsMaxRate(5)	INTEGER	The maximum rate latched for this program since last reset.	Get
progsMinLimit(6)	INTEGER	The minimum limit for this program.	Get/Set
progsMaxLimit(7)	INTEGER	The maximum limit for this program.	Get/Set
progsState(8)	EvState	The state of this program.	Get
progsPMTTestEnabled(9)	INTEGER	Determines whether PMT Test is enabled for this program. 0 = disabled, 1 = enabled.	Get/Set
progsPMTTestState (11)	INTEGER	Reading this returns the current state of the PMT Test for the program. Writing any value will reset the PMT Test for the program. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set

- **Indexing.** The table is indexed by MPEG Interface, followed by Program Index. An index of 0 is not allowed in SNMP tables, so the Program Index is actually Program + 1. Therefore, in order to reference the required Program item, for example progsMinLimit, use the following OID:

  '…pidsMinLimit.<interface>.<prog+1>'.

- **Program Limits.** By default, the limits for each Program are not defined; this is represented by the progsMinLimit and progsMaxLimit values being set to 0 and -1 respectively. When setting a limit, the management application must ensure that the value of progsMaxLimit is always greater than progsMinLimit, otherwise the new setting will not be accepted by the MTM400. The new and current values of progsMinLimit and progsMaxLimit will therefore affect the order in which the management application sets these limits.

  The limits for a Program can be cancelled at a later date by setting progsMinLimit to 0, and then setting progsMaxLimit to -1. Although -1 is less than 0, this is a special case, which is accepted by the MTM400.

- **Program Occupancy Events.** The MPEG Interface event 0x2002 will be generated whenever any Program occupancy limit is exceeded. The management application can choose to poll this at the required interval.

SFN Table. The SFN Table contains the Single Frequency Network Information for each interface, and is defined as:

Variable	Type	Use	Access
sfnMifIndex(1)	INTEGER	MPEG interface for which these elements apply.	N/A
sfnSynchronisation (2)	INTEGER	The SFN Synchronisation Scheme (usually 0).	Get
sfnSectionLength (3)	INTEGER	Number of bytes following the section_length field.	Get
sfnPointer(4)	INTEGER	Number of transport packets between the MIP and the first packet of the succeeding Mega Frame.	Get
sfnPeriodicFlag(5)	INTEGER	0 = aperiodic, 1 = periodic insertion of the MIP.	Get
sfnSynchronisationTimeStamp(6)	INTEGER	Time difference between the latest pulse of the 'one pulse per second' reference and the actual start of this Mega Frame in units of 100 ns.	Get
sfnMaximumDelay(7)	INTEGER	Delay between start of Mega Frame at the antenna, and the start of it at the SFN adapter in units of 100 ns.	Get
sfnTPSMip(8)	OCTET STRING	Four bytes containing bitstream P0-P31 of the Transport Parameter Signaling (TPS) information defined in TS 101 191 V1.2.1.	Get
SfnIndividualAddressing(9)	INTEGER	Total length of the individual addressing field in bytes.	Get
sfnMegaFrameSize(10)	INTEGER	Calculated Mega Frame Size.	Get
sfnDelay(11)	INTEGER	Calculated Delay.	Get
sfnInaccuracy(12)	INTEGER	Calculated Inaccuracy.	Get

Variable	Type	Use	Access
sfnFunctionBytes(13)	OCTET STRING	The bytes immediately following the individual_addressing_length field of the MIP up to the crc_32, which contains the function descriptors.	Get
sfnExists(14)	INTEGER	Indicates whether the SFN PID (0x15) exists in the transport stream. 0 = false, 1= true.	Get

The table is indexed by MPEG Interface. As an example, in order to reference sfnTPSMip for interface 1, use the following OID:

'…sfnTPSMip.1'.

PID Group Table. The PID Group Table provides access to PID group related information for each interface, and is defined as:

Variable	Type	Use	Access
pidGroupMifIndex (1)	INTEGER	MPEG interface for which these PID groups apply.	N/A
pidGroupIndex (2)	INTEGER	The index of this group.	N/A
pidGroupName (3)	DISPLAY STRING	The PID group name.	Get/Set
pidGroupRate (4)	INTEGER	The most recently measured bit rate of this PID group. Units are bit/s.	Get
pidGroupMinRate (5)	INTEGER	The lowest measured bit rate of this PID group since the minimum measured rate was last reset. Units are bit/s.	Get
pidGroupMaxRate (6)	INTEGER	The highest measured bit rate of this PID group since the maximum measured rate was last reset. Units are bit/s.	Get
pidGroupMinLimit (7)	INTEGER	The lower bit rate limit on this PID group. . Units are bit/s.	Get/Set
pidGroupMaxLimit (8)	INTEGER	The upper bit rate limit on this PID group. . Units are bit/s.	Get/Set
pidGroupState (9)	INTEGER	Reading this returns the current event status with respect to whether the PID group's bit rate has gone outside the bit rate limits. See the EvState type. Writing any value will reset the 'PID Group Occupancy' event.	Get/Set
pidGroupNewPid (11)	INTEGER	Setting this value adds the PID specified to the group. Reading this field has no meaning.	Set
pidGroupDeletePid (12)	INTEGER	Setting this value deletes the PID specified from the group. Reading this field has no meaning.	Set

PID Group PIDS Table. The PID Group PIDS Table provides access to the lists of PIDS defined for each group. It is defined as follows:

Variable	Type	Use	Access
pidGroupPidsMifIndex (1)	INTEGER	MPEG interface for which these PID group PIDs apply.	N/A
pidGroupPidsGroupIndex (2)	INTEGER	The index of the group of interest.	N/A
pidGroupPidsPidIndex (3)	INTEGER	The PID plus 1. This index is one greater than the number of the PID because PID 0 is valid, but an index of 0 into an SNMP table is not.	N/A
pidGroupPidsInGroup (4)	INTEGER	Specifies whether the PID (as specified by pidGroupPidsPidIndex - 1) belongs to the group. Setting this to 0 will remove the PID from the group.	Get

Structure Group 2

This provides access to the unformatted raw byte stream information stored in the MPEG Tables that describe the structure of MPEG transport streams.

There are two main problems with attempting to provide MPEG structure information through an SNMP interface. Firstly, the amount of information stored in MPEG Tables can grow to an arbitrarily large size, certainly more than the 484 bytes SNMP systems are required to support, and potentially larger than the maximum UDP packet size. Secondly, this information can change fairly rapidly.

In order to solve the first problem, the information for each MPEG Table is split up into manageable 'chunks' with a maximum size of 128 bytes. However, the second problem of potential rapid updates means that the MPEG table information can change between reading the separate chunks. Consequently, serial numbers are used to represent versions of MPEG Tables at particular times.

Figure 13 shows the way in which the MPEG transport stream information is represented within the MTM400 MIB. SNMP tables have been used to represent the data stored in MPEG Tables, so it is possible that some confusion may arise over terminology, so specific reference has been made as to whether MPEG or SNMP tables are being discussed in the descriptions below.

In the MPEG standard, each MPEG Table has an identifier, which is represented as a single byte value. For example, the Program Association Table has a table id of 0x00. The use of these MPEG Table identifiers within the Structure Group is consistent with this standard.

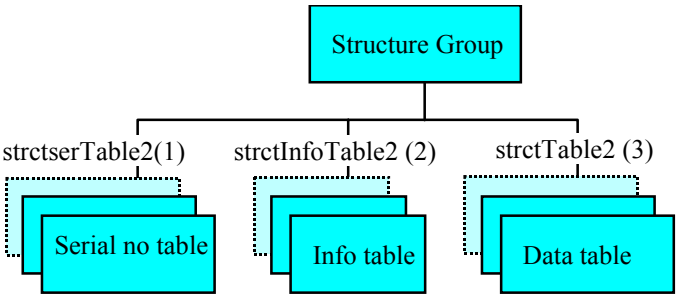

Figure 13: Structure Group 2 Structure

Serial Number Table. The SNMP Serial Number Table contains the serial numbers that should be used to index the SNMP Size and Data tables in order to obtain the most up to date information for each MPEG Table. Each serial number is incremented every time its MPEG Table changes. Management applications attempting to use out of date serial numbers to read the Size and Data SNMP tables will receive SNMP 'No Such Name' errors. If this happens, they should attempt to obtain the new serial number for this table and start again. (MPEG Tables can also disappear completely without being replaced by a more up to date version, in which case, the management application will need to abort the operation.)

The Serial Number table is defined as follows:

Variable	Type	Use	Access
strctserMifIndex2(1)	INTEGER	The MPEG interface for which these readings apply.	N/A
strctserTableIndex2(2)	INTEGER	The MPEG Table Id (+1).	N/A
strctserMajorExtensionIndex2(3)	INTEGER	Top 32 bits of the sub-table unique identifier (+ 1).	N/A
strctsetMinorExtensionIndex2(4)	INTEGER	Bottom 32 bits of the sub-table unique identifier (+ 1).	N/A
strctserNumber2(5)	INTEGER	The serial number of the most up to date version of this MPEG Table.	Get

As an example, the OID '…strctserNumber2.1.67.1081.54' would return the most recent serial number for the DVB table id 66 (Service Description Table) where:

.1. = Stream 1 (default)

.67. = table id + 1

.1081.54 = unique identifier of the subtable

Info Table. This contains the total number of bytes stored for a specified version (referenced by serial number) of each MPEG Table on each interface. The size should be used to cross check that the correct numbers of bytes are read from the SNMP Data table. The table also contains the PID number on which the table was transmitted.

The table is defined as follows:

Variable	Type	Use	Access
strctInfoMifIndex2(1)	INTEGER	The MPEG interface for which these readings apply.	N/A
strctInfoTableIndex2(2)	INTEGER	The MPEG Table Id (+1).	N/A
strctInfoMajorExtensionIndex2(3)	INTEGER	Top 32 bits of the table unique identifier (+ 1)	N/A
strctInfoMinorExtensionIndex2(4)	INTEGER	Bottom 32 bits of the table unique identifier (+ 1)	N/A
stcrtInfoSerialIndex2(5)	INTEGER	The serial number of this table.	N/A
strctInfoSize2(6)	INTEGER	The number of bytes in this table.	Get
strctInfoPid2(7)	INTEGER	The PID this table was transmitted on	Get

As an example, the OID '…strctInfoSize2.1.67.1081.54.2' would return the size of the table id 66 (SDT) with the serial number 2.

Data Table. This table contains the data from each version (referenced by serial number) of each MPEG Table on each interface split which has been split into 'chunks'.

Variable	Type	Use	Access
strctMifIndex(1)	INTEGER	The MPEG interface for which these readings apply.	N/A
strctTableIndex2(2)	INTEGER	The MPEG Table Id (+1).	N/A
strctExtensionIndex2(3)	INTEGER	Top 32 bits of the table unique identifier (+ 1)	N/A
strctMinorExtensionIndex2(4)	INTEGER	Bottom 32 bits of the table unique identifier (+ 1)	N/A
stcrtSerialIndex2(5)	INTEGER	The serial number of this table.	N/A
strctChunkIndex2(6)	INTEGER	The chunk index of this table.	Get
strctTableData2(7)	OCTET STRING	The raw bytes in this chunk.	Get

The data from the MPEG Table is split into sequential 'chunks' of up to 128 bytes, and the Chunk Index is the 'chunk' number that this TableData item represents. Management applications must concatenate the appropriate 'chunks' together in order to reconstruct the data contained in the corresponding MPEG Table.

As an example, the following OIDs '…strctTableData2.1.67.1081.54.2.1', '…strctTableData2.1.67.1081.54.2.2' and '…strctTableData2.1.67.1081.54.2.3' would return all of the data for serial number 2 of MPEG Table 66 (SDT) , assuming it was split into 3 'chunks'.

MPEG Log Group

Figure 14 shows the structure of the MPEG Log Group, which provides access to the stream specific log items.

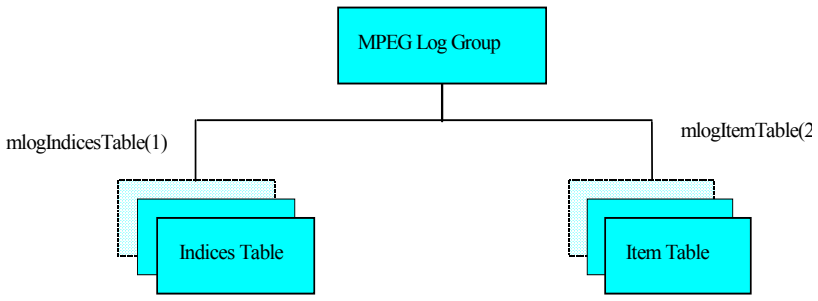

Figure 14: MPEG Log Group Structure

Indices Table. The Indices Table contains the most recent and oldest indices of the log entries for each stream, and is defined as:

Variable	Type	Use	Access
mlogIndecesMifIndex (1)	INTEGER	The MPEG Interface.	N/A
mlogRecentIndex(2)	INTEGER	The index of the most recent log entry on this interface.	Get
mlogOldestIndex(3)	INTEGER	The index of the most oldest log entry on this interface.	Get
mlogClear (4)	INTEGER	Writing any value to this variable will clear the stream log. Reading this field has no meaning.	Set

- **Indexing.** As the table index is an integer, this may wrap around if the number of entries in the log becomes significantly large. This means that the element with the largest index is not necessarily the latest log entry.

Item Table. The Item table contains the log entries for each interface, and is defined as:

Variable	Type	Use	Access
mlogItemMifIndex(1)	INTEGER	The MPEG Interface.	N/A
mlogItemIndex(2)	INTEGER	The log item index.	N/A
mlogItem(3)	OCTET STRING	Byte stream containing the log entry (see below for details).	Get

- **Log Entry.** The mlogItem entry will be empty if the index requested is not valid. This occurs if the management application requests an entry that no longer exists, for example, if the log was full and the entry was deleted from the end of the list to make room for new entries. If mlogItem is not empty, the format of the octet string is as follows:

  All numeric values are coded L.S.B. first:

  Bytes 0..7: Public Timestamp Structure as defined in 2.1.5

  Bytes 8..9: Stream number (1 for MTM400)

  Bytes 10..11: EvId event id for the event

  Bytes 12..13: Event id extension (zero if not applicable)

  Bytes 14..15: EvState for the event

  Bytes 16 onwards: Log text coded as UTF-8

MPEG Configuration Group

Figure 15 shows the structure of the MPEG Configuration Group that manages the stream configuration slots.

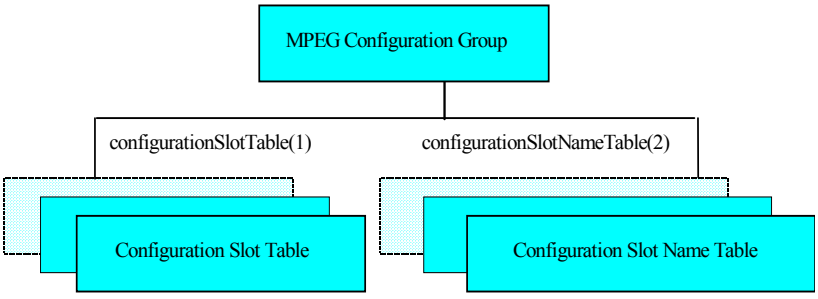

Figure 15: MPEG Configuration Group

Configuration Slot Table

The configuration slot table contains items for copying and storing stream configuration slots, and is defined as:

Variable	Type	Use	Access
configurationSlotMifIndex (1)	INTEGER	The MPEG interface for which these items apply.	N/A
copyStoredSlot(2)	INTEGER	Setting this copies the specified stored configuration slot to the active slot. Reading this returns the stored configuration slot last copied to the active slot. Valid values to read and set are 1-8. Reading a value of 0 implies that no stored slot has been copied to the active slot.	Get/Set
storeActiveSlot(3)	INTEGER	Setting this copies the current settings (held in the active slot) to the specified stored configuration slot.	Set
slotCopyTime (4)	OCTET STRING	The time at which a stored configuration was last copied to the active slot, or the active slot was copied to a stored slot. Time in time stamp format (see page 6).	Get
currentConfigurationSlotName(5)	INTEGER	This returns the name of the configuration last copied to the active slot.	Get
clearStoredSlot (6)	INTEGER	Setting this clears the contents of the specified stored configuration slot. Reading this value has no meaning.	Set

Configuration Slot Name Table

The configuration slot name table contains the name of the configuration stored in each slot, and is defined as:

Variable	Type	Use	Access
configurationSlotNameMifIndex (1)	INTEGER	The MPEG interface for which these slot names apply.	N/A
configurationSlotNameIndex(2)	INTEGER	The slot number of interest 1..8.	N/A
configurationSlotName (3)	Display String	The name of the slot.	Get

MPEG Parameters Group

Figure 16 shows the structure of the MPEG Parameters Group, which manages the Stream, PID, Program and PID Group parameters.

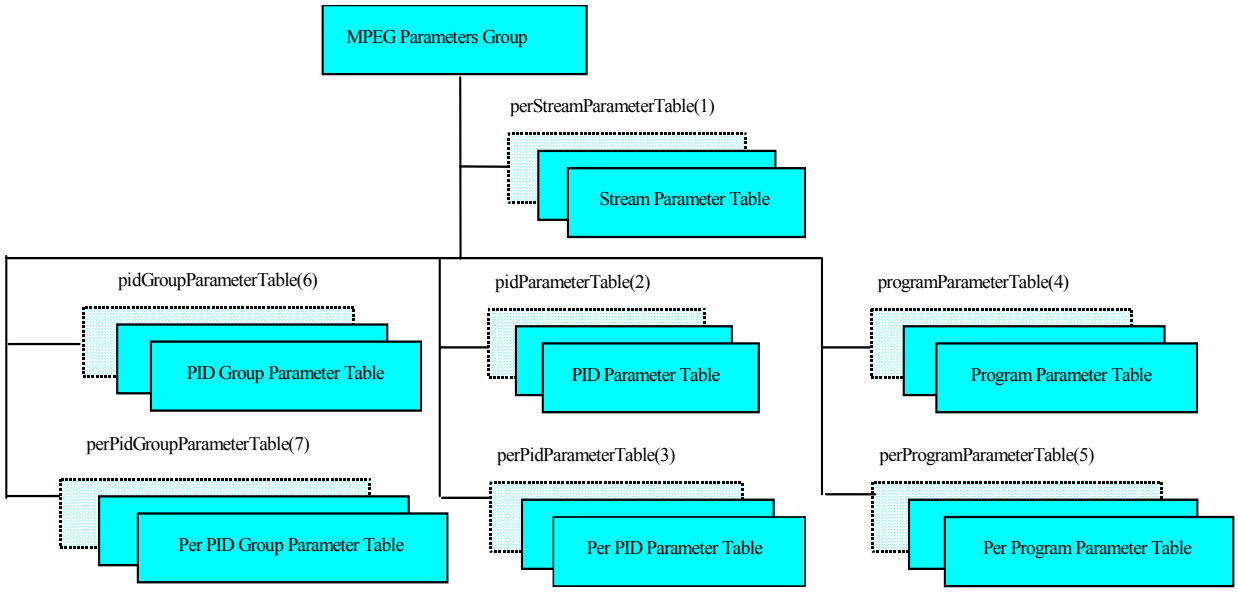

Figure 16: MPEG Parameters Group Structure

Stream Parameter Table. The stream parameter table provides access to the stream parameters, and is defined as:

Variable	Type	Use	Access
perStreamParameterMifIndex (1)	INTEGER	The MPEG interface associated with these parameters.	N/A
perStreamParameterIndex (2)	INTEGER	The unique Id of the required parameter.	N/A
perStreamParameterValue (3)	OCTET STRING	The value of this parameter as a string.	Get/Set

PID Parameter Table. The PID parameter table provides access to the default PID parameter values, and is defined as:

Variable	Type	Use	Access
pidParameterMifIndex (1)	INTEGER	The MPEG interface associated with these parameters.	N/A
pidParameterIndex (2)	INTEGER	The unique Id of the required PID parameter.	N/A
pidParameterDefaultValue (3)	OCTET STRING	The default value of this PID parameter as a string.	Get/Set

PID tests and events will inherit these values by default. Individual PID parameters can be customized using the Per PID Parameter table.

Per PID Parameter Table. The Per PID parameter table provides access to individual PID parameters, and is defined as:

Variable	Type	Use	Access
perPidParameterMifIndex (1)	INTEGER	The MPEG interface associated with these PID specific parameters.	N/A
perPidParameterIndex (2)	INTEGER	The unique Id of the required PID parameter.	N/A
perPidParameterPidIndex (3)	INTEGER	The PID of interest (Pid Number + 1).	N/A
perPidParameterValue (4)	OCTET STRING	The value of this specific PIDs parameter as a string.	Get/Set

Program Parameter Table. The Program parameter table provides access to the default Program parameter values, and is defined as:

Variable	Type	Use	Access
programParameterMifIndex (1)	INTEGER	The MPEG interface associated with these parameters.	N/A
programParameterIndex (2)	INTEGER	The unique Id of the required Program parameter.	N/A
programParameterDefaultValue (3)	OCTET STRING	The default value of this Program parameter as a string.	Get/Set

Program tests and events will inherit these values by default. Individual Program parameters can be customized using the Per Program Parameter table.

Per Program Parameter Table. The Per Program parameter table provides access to individual Program parameters, and is defined as:

Variable	Type	Use	Access
perProgramParameterMifIndex (1)	INTEGER	The MPEG interface associated with these Program specific parameters.	N/A
perProgramParameterIndex (2)	INTEGER	The unique Id of the required Program parameter.	N/A
perProgramParameterProgram Index (3)	INTEGER	The Program (program number +1) of interest.	N/A
perProgramParameterValue (4)	OCTET STRING	The value of this specific Programs parameter as a string.	Get/Set

PID Group Parameter Table. The PID Group parameter table provides access to the default PID Group parameter values, and is defined as:

Variable	Type	Use	Access
pidGroupParameterMifIndex (1)	INTEGER	The MPEG interface associated with these parameters.	N/A
pidGroupParameterIndex (2)	INTEGER	The unique Id of the required PID Group parameter.	N/A
pidGroupParameterDefaultValue (3)	OCTET STRING	The default value of this PID Group parameter as a string.	Get/Set

PID Group tests will inherit these values by default. Individual PID Group parameters can be customized using the Per PID Group Parameter table.

Per PID Group Parameter Table. The Per PID Group parameter table provides access to individual PID Group parameters, and is defined as:

Variable	Type	Use	Access
perPidGroupParameterMifIndex (1)	INTEGER	The MPEG interface associated with these PID specific parameters.	N/A
perPidGroupParameterIndex (2)	INTEGER	The unique Id of the required PID Group parameter.	N/A
perPidGroupParameterPidGroup Index (3)	INTEGER	The PID Group of interest (Group Number)	N/A
perPidGroupParameterValue (4)	OCTET STRING	The value of this specific PID Groups parameter as a string.	Get/Set

MPEG Record Group

The MPEG Record Table provides the control and monitoring interface for the MTM400 triggered recording function, and is defined as follows:

Variable	Type	Use	Access
mpegRecordMifIndex (1)	INTEGER	The MPEG interface.	N/A
mpegRecordState (2)	INTEGER	State of recording: 0 = Idle 1 = Waiting for Trigger 2 = Recording in Progress 3 = Recording Complete	Get
mpegRecordTriggerType (3)	INTEGER	Type of recording trigger: 0 = Immediate (default) 1 = External Rising Edge 2 = External Falling Edge 3 = Event Alarm	Get/Set
mpegRecordLargestAllowed (4)	INTEGER	Largest number of packets allowed to record.	Get
mpegRecordPreTrigger (5)	INTEGER	Percentage of stream prepended to the recording before the trigger set off.	Get
mpegRecordActualSize (6)	INTEGER	Actual size of recording in packets.	Get
mpegRecordTotalMemorySize (7)	INTEGER	Total system memory size (in Megabytes) available for recording.	Get
mpegRecordActivate (8)	INTEGER	Setting this starts the wait for the trigger.	Set
mpegRecordClear (9)	INTEGER	Setting this clears the recording.	Set
mpegRecordTimestampAvailable (10)	INTEGER	Specifies whether device is capable of time-stamping recorded packets.	Get
mpegRecordUseTimestamp (11)	INTEGER	Specifies whether to	Get/Set

Variable	Type	Use	Access
		timestamp packets.	
mpegRecordProgress (12)	INTEGER	The percentage of the recording completed.	Get
mpegRecordDesiredSize (13)	INTEGER	Desired size of recording in packets.	Get/Set
mpegRecordTriggerTime (14)	INTEGER	This returns the time at which the trigger for the current recording occurred, or zero if not currently meaningful.	Get

MPEG Trap Control

The trap control group provides the variables to support the traps sent and the configuration items to control trap generation.

Clients subscribe to traps by writing their IP address into trapSink; they are automatically deleted from the notification list after trapSinkTimeout minutes. So a client should resubscribe every few minutes. TrapSinkTimeout may be 0, which means infinite.

Trapthrottle limits how many Traps per second may be generated, this is to stop the network being overloaded with traps.

There is a single trap type, this has a payload that defines the event and associated data.

Name	Type	Access	Comment
TrapSink	IpAddress	WO	Clients write their IP address into this variable to register that they want to receive traps.
TrapThrottle	Integer	RW	Specify the maximum number of traps issued per second.
TrapEventID	Integer	RO	Data for last trap fired.
TrapStatus	Integer	RO	
TrapTransportID	Integer	RO	
TrapNetworkID	Integer	RO	
TrapServiceID	Integer	RO	
TrapServiceType	Integer	RO	
TrapPID	Integer	RO	
TrapTimeStamp	String	RO	
TrapThresholdValue	String	RO	
TrapActualValue	String	RO	
TrapDuration	Integer	RO	
TrapStream	Integer	RO	Stream number, fixed at one in MTM400.
TrapSinkTimeout	Integer	RW	Minutes before unsubscribing trap client, 0 is infinite.

Web Server URLs

The following sections define the URLs supported by the MTM400 Web Server.

Upload Configuration

`http::://<MTM400 IP Address>/cgi-bin/uploadconfiguration?stream=x&slot=y`

The ‘stream’ parameter is always 1 for MTM400. The slot corresponds to the configuration slot into which the new configuration parameters will be loaded. This is in the range 1…8.

Download Configuration

`http::://<MTM400 IP Address>/cgi-bin/downloadconfiguration?stream=x&slot=y`

The ‘stream’ parameter is always 1 for MTM400. The slot corresponds to the configuration slot from which the parameters will be downloaded. This is in the range 1…8.

Upload Schedule

`http::://<MTM400 IP Address>/cgi-bin/uploadschedule?stream=x`

This URL is used to upload a schedule file for the specified stream interface. The ‘stream’ parameter is always 1 for MTM400.

Download Schedule

`http::://<MTM400 IP Address>/cgi-bin/downloadschedule?stream=x`

This URL is used to download a schedule file from the specified stream interface. The ‘stream’ parameter is always 1 for MTM400.

Download Recording

`http::://<MTM400 IP Address>/data/recording?start=x&end=y`

This URL is used to download a stream recording. The ‘start’ and ‘end’ parameters define the range of packets of interest.

Download Stream Log

```
http::://<MTM400 IP Address>/cgi-bin/streamlog?start=x&end=y
```

This URL is used to download the stream log. The ‘start’ and ‘end’ parameters define the range of log entries of interest. The available range of log entries can be determined from the SNMP MIB table ‘mpegLog’.

The stream log may be very large, so the URL allows for sections of the log to be downloaded. An XML format log file is downloaded in response to the invocation of this URL. There is no acknowledgement required because the log entries are only destroyed by the wrapping of its circular buffer.

Download Device Log

```
http::://<MTM400 IP Address>/cgi-bin/ devicelog?start=x&end=y
```

This URL is used to download the device log. The ‘start’ and ‘end’ parameters define the range of log entries of interest. The range of log entries available can be determined from the SNMP MIB table ‘adsysLog’.

An XML format log file is downloaded in response to the invocation of this URL. There is no acknowledgement required because the log entries are only destroyed by the wrapping of its circular buffer.

The Stream and Device Log downloads can also take a language parameter. For example,

```
http::://<MTM400 IP Address>/cgi-bin/devicelog?start=x&end=y&lang=x
```

where x=24 (English), 52 (Japanese) or 134 (Chinese).

Download Service Log

```
http::://<Despina IP Address>/service/log?reqid=ID
```

This URL results in a CSV file of service Log results being downloaded. Each request has a unique user-defined ID that is used in the acknowledge phase.

It must be acknowledged that data has been processed fully before it can be destroyed, so the client must make a request with a matching 'reqid' to clear the data as follows:

```
http::://<Despina IP Address>/service/logack?reqid=ID
```

The PIDs involved in the service log may be specified in the configuration file or configured by HTTP requests. The following two URLs will introduce or remove a PID from the service log.

```
http::://<Despina IP Address>/service/addpid?pid=1234
http::://<Despina IP Address>/service/delpid?pid=1234
```

Use the following URL to return a list of current PIDs in XML format.

```
http://<Despina IP Address>//service/current.xml
```

Appendix A – Event Identities

This appendix defines the event ids currently defined within MTM400.

Event Id	Event Name	Event Description
0x0000	EVID_ANY_ERR	Any error
0x0010	EVID_ANY_ALARM_ON	
0x1000	Any Box Error	The state of any box error
0x1001	Fan State	The current state of the fan
0x1002	EVID_HARD_DISK	
0x1003	EVID_FLOPPY_DISK	
0x1004	Temperature	Indicates whether temperature is within operating range
0x1005	EVID_DCVOLTAGE	
0x1020	EVID_PLUS_FIVE_VOLTS	
0x1021	EVID_PLUS_TWELVE_VOLTS	
0x1022	EVID_MINUS_FIVE_VOLTS	
0x1023	EVID_MINUS_TWELVE_VOLTS	
0x1030	Local Temperature	Local Temperature
0x1031	Remote Temperature	Remote Temperature
0x1032	System Card Temperature	System Card Temperature
0x1040	Fan 1	The current state of fan 1
0x1041	Fan 2	The current state of fan 2
0x1042	Fan 3	The current state of fan 3
0x1043	Fan 4	The current state of fan 4
0x1044	Fan Monitor	The current state of the fan monitor
0x1100	EVID_SV_START	
0x1101	EVID_SV_STOP	
0x1102	EVID_SV_INIT_FAIL	
0x1103	EVID_ALARM_RESET	
0x1104	EVID_EVENT_RESET	
0x1105	EVID_CLEAR_LOG	
0x1200	EVID_SV_DEBUG	
0x1201	I2C	I2C communications state
0x1202	Battery	Battery state
0x1203	EVID_BOX_CONFIG	
0x1204	EVID_INTERFACE_FIRMWARE	
0x1205	EVID_ASSERTION	
0x1206	Real Time Clock	The current state of the RTC

Event Id	Event Name	Event Description
0x1207	EVID_FIRMWARE_UPLOAD	
0x1208	EVID_NETWORK	
0x1209	EVID_LOGIC	
0x1210	EVID_MISC_HARDWARE	
0x1211	LTC Clock	Longitudinal Time Clock
0x1212	Network Clock	Simple Network Time Protocol Clock
0x1213	Time Source	Time Source state
0x1214	Front Panel	Front Panel state
0x1215	EVID_FRONT_PANEL_ITEM	
0x1216	System Card	System Card state
0x2000	Any Stream Error	Any stream error has occured
0x2001	Any PID Occupancy	PID occupancy exceeds limits
0x2002	Any Program Occupancy	Program occupancy exceeds limits
0x2003	Any PID Group Occupancy	PID Group occupancy exceeds limits
0x2004	EVID_INDIVIDUAL_PID_OCC_LIM	
0x2005	EVID_ANY_PIDGROUP_OCC_LIM	
0x2007	EVID_ANY_PID_ERR	
0x2008	EVID_ANY_PROG_ERR	
0x2010	PID Bit rate Variability	PID bit rate variability exceeds limits
0x2011	EVID_INDIVIDUAL_PID_BITRATE_VARIABILITY	
0x2100	EVID_STREAM_MISC_INF	
0x2101	EVID_STREAM_MISC_WARN	
0x2102	EVID_STREAM_MISC_ERR	
0x3011	Sync	Sync (DVB test 1.1)
0x3012	Sync Byte	Sync Byte (DVB test 1.2)
0x3014	Continuity	Continuity Error (DVB test 1.4)
0x3016	PID	PID (DVB test 1.6)
0x3017	EVID_INDIVIDUAL_PID_ERROR	
0x3018	PAT Table	PAT Table (DVB test 1.3)
0x3019	PMT	PMT (DVB test 1.5)
0x3021	Transport	Transport (DVB test 2.1)
0x3022	CRC	CRC (DVB test 2.2)
0x3023	PCR	PCR (DVB test 2.3)
0x3024	PCR Accuracy	PCR Accuracy (DVB test 2.4)
0x3025	PTS	PTS (DVB test 2.5)

Event Id	Event Name	Event Description
0x3026	CAT	CAT (DVB test 2.6)
0x3027	EVID_INDIVIDUAL_PCR_ERR	
0x3028	EVID_INDIVIDUAL_PCR_ACCURACY_ERR	
0x3031	NIT	NIT (DVB test 3.1)
0x3032	SI Repetition	SI Repetition (DVB test 3.2)
0x3035	SDT	SDT Error (DVB test 3.5)
0x3036	EIT	EIT Error (DVB test 3.6)
0x3037	RST	RST Error (DVB test 3.7)
0x3038	TDT	TDT Error (DVB test 3.8)
0x3039	EVID_PROG_OCC_LIM	
0x303A	EVID_INDIVIDUAL_UNREF_PID_ERR	
0x303B	Unref PID	Unreferenced PID (DVB test 3.4)
0x3040	EVID_PCR_OJ_ERR	PCR overall jitter
0x3041	EVID_PCR_FO_ERR	PCR frequency offset
0x3042	EVID_PCR_DR_ERR	PCR drift rate
0x3043	EVID_INDIVIDUAL_PCR_OJ_ERR	
0x3044	EVID_INDIVIDUAL_PCR_FO_ERR	
0x3045	EVID_INDIVIDUAL_PCR_DR_ERR	
0x3050	EVID_TS_SYNC_STATE	
0x3051	EVID_INDIVIDUAL_DVBEIT_PF_PRESENCE	
0x3052	EVID_DVBEIT_PF_PRESENCE_ERROR_ EVENT	
0x3100	PAT Err Timer	PAT Error Timer
0x3100	EVID_PAT_ERR_TIMER	
0x3101	PAT Err Table Id	PAT Error Table Id
0x3102	PAT Err Scr	PAT Error Scrambling
0x3103	PMT Err Timer	PAT Error Timer
0x3104	PMT Err Scr	PMT Error Scrambling
0x3105	SDT Err Timer	SDT Error Timer
0x3106	SDT Err Table Id	SDT Error Table Id
0x3107	CAT Err Table Id	CAT Error Table Id
0x3108	NIT Err Timer	NIT Error Timer
0x3109	NIT Err Table Id	NIT Error Table Id
0x3110	EIT Err Timer	EIT Error Timer
0x3111	EIT Err Table Id	EIT Error Table Id
0x3112	TDT Err Timer	TDT Error Timer
0x3113	TDT Err Table Id	TDT Error Table Id

Event Id	Event Name	Event Description
0x3114	CAT Err Scrambling	CAT Error Scrambling
0x3115	PCR Err Timer	PCR Error Timer
0x3116	PCR Err Disc	PCR Error Disc
0x3117	EVID_INDIVIDUAL_PCR_ERR_TIMER	
0x3118	EVID_INDIVIDUAL_PCR_ERR_DISC	
0x3120	EVID_RST_ERR_TABLE_ID	
0x3124	EVID_ANY_INDIVIDUAL_ERROR	
0x3130	EVID_CONTINUITY_COUNT_ERROR	
0x3131	EVID_DISCONTINUITY_ERROR	
0x3132	EVID_INDIVIDUAL_CONTINUITY_ERR	
0x3133	EVID_INDIVIDUAL_DISCONTINUITY_ERROR	
0x3140	EVID_NIT_ACTUAL_ERR	Any NIT actual error
0x3142	EVID_SDT_ACTUAL_ERR	Any SDT actual error
0x3143	EVID_EIT_ACTUAL_ERR	Any EIT actual error
0x3144	EVID_EIT_OTHER_ERR	Any EIT other error
0x3150	EVID_PCR_ERR_TIMER_2	PCR Error Timer
0x3151	EVID_PCR_ERR_DISC_2	PCR Error Disc
0x3160	EVID_INDIVIDUAL_PMT_ERR_TIMER	
0x3201	MGT Err	A/65 MGT Error
0x3202	STT Err	A/65 STT Error
0x3203	RRT Err	A/65 RRT Error
0x3204	ATSC EIT Err	A/65 EIT Error
0x3205	VCT Err	A/65 VCT Error
0x3206	ETT Err	ETT Error
0x3207	EIT Mode Err	EIT Mode Error
0x3208	Base PID Err	A/65 Base PID Error
0x3210	VCT Timer Err	VCT Timer Error
0x3211	MGT Timer Err	MGT Timer Error
0x3212	STT Timer Err	STT Timer Error
0x3213	RRT Timer Err	RRT Timer Error
0x3214	ATSCEIT Timer Err	ATSCEIT Timer Error
0x3230	EVID_MGT_EITK_PRESENCE	
0x3300	EVID_TEMPLATE_MATCH	
0x3320	Program Paradigm	A/53 Program Paradigm
0x3330	EVID_PROGRAM_NUMBER_CONSISTENCY	PAT/PMT program_number consistency
0x3331	EVID_PAT_SDT_CONSISTENCY	PAT/SDT consistency

Event Id	Event Name	Event Description
0x3332	EVID_PAT_VCT_CONSISTENCY	PAT/xVCT consistency
0x3333	EVID_DVBEIT_PF_PRESENCE	DVB EIT P/F presence consistency
0x3400	SFN Err	SFN Error
0x3401	SFN No MIP Err	No SFN Mip
0x3401	EVID_SFN_NOMIP_ERR	
0x3402	SFN Packet Cout Err	Error in SFN Packet Count
0x3403	SFN Invalid MIP Err	Invalid SFN Mip
0x3404	SFN Timer Err	Error in SFN Timer
0x3410	EVID_SFN_ONE_MIP_PER_MF	
0x3411	EVID_SFN_REPETITION	
0x3412	EVID_SFN_MIP_LENGTH	
0x3413	EVID_SFN_CRC_ERROR	
0x3414	EVID_SFN_MIP_CODING	
0x3415	EVID_SFN_PERIODICITY_CONSISTENT	
0x3416	EVID_SFN_POINTER_CONSISTENT	
0x3500	EVID_SYNTAX_PAT	
0x3501	EVID_SYNTAX_PMT	
0x3502	EVID_SYNTAX_CAT	
0x3503	EVID_SYNTAX_NIT	
0x3504	EVID_SYNTAX_BAT	
0x3505	EVID_SYNTAX_SDT	
0x3506	EVID_SYNTAX_DVBEIT	
0x3507	EVID_SYNTAX_TDT	
0x3508	EVID_SYNTAX_TOT	
0x3509	EVID_SYNTAX_RST	
0x350A	EVID_SYNTAX_MGT	
0x350B	EVID_SYNTAX_RRT	
0x350C	EVID_SYNTAX_VCT	
0x350D	EVID_SYNTAX_ATSCEIT	
0x350E	EVID_SYNTAX_STT	
0x350F	EVID_SYNTAX_ETT	
0x3510	EVID_SYNTAX_DCCT	Syntax error in the Directed Channel Change Table (DCCT)
0x3511	EVID_SYNTAX_DCCSCT	Syntax error in the Directed Channel Change Selection Code Table (DCCSCT)
0x3520	EVID_ANY_SYNTAX	Section Syntax Error

Event Id	Event Name	Event Description
0x3600	EVID_TRANSPORT_STREAM_BITRATE	Transport Stream bit rate within limits
0x3703	EVID_NIT_ACTUAL_MIN_SECTION_RI	
0x3704	EVID_NIT_ACTUAL_MAX_SUBTABLE_RI	
0x3705	EVID_NIT_ACTUAL_MAX_SECTION_RI	
0x3706	EVID_NIT_OTHER_MIN_SECTION_RI	
0x3707	EVID_NIT_OTHER_MAX_SUBTABLE_RI	
0x3708	EVID_NIT_OTHER_MAX_SECTION_RI	NIT other max section RI
0x3709	EVID_SDT_ACTUAL_MIN_SECTION_RI	
0x370A	EVID_SDT_ACTUAL_MAX_SUBTABLE_RI	
0x370B	EVID_SDT_ACTUAL_MAX_SECTION_RI	
0x370C	EVID_SDT_OTHER_MIN_SECTION_RI	
0x370D	EVID_SDT_OTHER_MAX_SUBTABLE_RI	
0x370E	EVID_SDT_OTHER_MAX_SECTION_RI	SDT other max section RI
0x370F	EVID_DVBEIT_ACTUAL_PF_MIN_SECTION_RI	
0x3710	EVID_DVBEIT_ACTUAL_PF_MAX_SUBTABLE_RI	
0x3711	EVID_DVBEIT_ACTUAL_P_MAX_SECTION_RI	
0x3712	EVID_DVBEIT_ACTUAL_F_MAX_SECTION_RI	
0x3713	EVID_DVBEIT_OTHER_PF_MIN_SECTION_RI	
0x3714	EVID_DVBEIT_OTHER_PF_MAX_SUBTABLE_RI	
0x3715	EVID_DVBEIT_OTHER_P_MAX_SECTION_RI	
0x3716	EVID_DVBEIT_OTHER_F_MAX_SECTION_RI	
0x3717	EVID_DVBEIT_ACTUAL_S_MIN_SECTION_RI	
0x3718	EVID_DVBEIT_ACTUAL_S_MAX_SUBTABLE_RI	
0x3719	EVID_DVBEIT_OTHER_S_MIN_SECTION_RI	
0x371A	EVID_DVBEIT_OTHER_S_MAX_SUBTABLE_RI	
0x371B	EVID_RST_MIN_SECTION_RI	
0x371C	EVID_TDT_MIN_SECTION_RI	
0x371E	EVID_TOT_MIN_SECTION_RI	
0x371F	EVID_TOT_MAX_SECTION_RI	
0x3720	EVID_BAT_MIN_SECTION_RI	
0x3721	EVID_BAT_MAX_SUBTABLE_RI	
0x3726	EVID_ATSCEIT0_MAX_INSTANCE_RI	
0x3727	EVID_ATSCEIT0123_MAX_SECTION_RI	
0x3728	EVID_PTS_RI	

Event Id	Event Name	Event Description
0x3729	EVID_INDIVIDUAL_NIT_ACTUAL_MIN_SECTION_RI	
0x372A	EVID_INDIVIDUAL_NIT_ACTUAL_MAX_SUBTABLE_RI	
0x372B	EVID_INDIVIDUAL_NIT_OTHER_MIN_SECTION_RI	
0x372C	EVID_INDIVIDUAL_NIT_OTHER_MAX_SUBTABLE_RI	
0x372D	EVID_INDIVIDUAL_NIT_OTHER_MAX_SECTION_RI	
0x372E	EVID_INDIVIDUAL_SDT_ACTUAL_MIN_SECTION_RI	
0x372F	EVID_INDIVIDUAL_SDT_ACTUAL_MAX_SUBTABLE_RI	
0x3730	EVID_INDIVIDUAL_SDT_OTHER_MIN_SECTION_RI	
0x3731	EVID_INDIVIDUAL_SDT_OTHER_MAX_SUBTABLE_RI	
0x3732	EVID_INDIVIDUAL_SDT_OTHER_MAX_SECTION_RI	
0x3733	EVID_INDIVIDUAL_DVBEIT_ACTUAL_PF_MIN_SECTION_RI	
0x3734	EVID_INDIVIDUAL_DVBEIT_ACTUAL_PF_MAX_SUBTABLE_RI	
0x3735	EVID_INDIVIDUAL_DVBEIT_OTHER_PF_MIN_SECTION_RI	
0x3736	EVID_INDIVIDUAL_DVBEIT_OTHER_PF_MAX_SUBTABLE_RI	
0x3737	EVID_INDIVIDUAL_DVBEIT_ACTUAL_S_MIN_SECTION_RI	
0x3738	EVID_INDIVIDUAL_DVBEIT_ACTUAL_S_MAX_SUBTABLE_RI	
0x3739	EVID_INDIVIDUAL_DVBEIT_OTHER_S_MIN_SECTION_RI	
0x373A	EVID_INDIVIDUAL_DVBEIT_OTHER_S_MAX_SUBTABLE_RI	
0x373B	EVID_INDIVIDUAL_RST_MIN_SECTION_RI	
0x373C	EVID_INDIVIDUAL_TDT_MIN_SECTION_RI	
0x373D	EVID_INDIVIDUAL_TOT_MIN_SECTION_RI	
0x373E	EVID_INDIVIDUAL_BAT_MIN_SECTION_RI	
0x373F	EVID_INDIVIDUAL_BAT_MAX_SUBTABLE_RI	

Event Id	Event Name	Event Description
0x3740	EVID_INDIVIDUAL_ATSCEIT0_MAX_INSTANCE_RI	
0x3741	EVID_INDIVIDUAL_SDTT_MIN_SECTION_RI	
0x3742	EVID_INDIVIDUAL_SDTT_MAX_SUBTABLE_RI	
0x3743	EVID_INDIVIDUAL_BIT_MIN_SECTION_RI	
0x3744	EVID_INDIVIDUAL_BIT_MAX_SUBTABLE_RI	
0x3745	EVID_INDIVIDUAL_NBIT_MIN_SECTION_RI	
0x3746	EVID_INDIVIDUAL_NBIT_MAX_SUBTABLE_RI	
0x3747	EVID_INDIVIDUAL_NBIT_LINK_INFO_MIN_SECTION_RI	
0x3748	EVID_INDIVIDUAL_NBIT_LINK_INFO_MAX_SUBTABLE_RI	
0x3749	EVID_INDIVIDUAL_LDT_MIN_SECTION_RI	
0x374A	EVID_INDIVIDUAL_LDT_MAX_SUBTABLE_RI	
0x374B	EVID_INDIVIDUAL_VCT_MAX_SUBTABLE_RI	
0x374C	EVID_VCT_MAX_SECTION_RI	
0x374D	EVID_INDIVIDUAL_CDT_MAX_SUBTABLE_RI	
0x374E	EVID_INDIVIDUAL_CDT_MIN_SECTION_RI	
0x374F	EVID_DCCSCT_MAX_SECTION_RI	Maximum Section repetition interval error for DCCSCT
0x3750	EVID_PAT_MISSING	
0x3751	EVID_NIT_ACTUAL_MISSING	
0x3752	EVID_SDT_ACTUAL_MISSING	
0x3753	EVID_EIT_ACTUAL_PF_MISSING	
0x3754	EVID_TDT_MISSING	
0x3755	EVID_MGT_MISSING	
0x3756	EVID_VCT_MISSING	
0x3757	EVID_RRT_MISSING	
0x3758	EVID_STT_MISSING	
0x3759	EVID_ATSCEIT0123_MISSING	
0x3760	EVID_SDTT_MIN_SECTION_RI	
0x3761	EVID_SDTT_MAX_SUBTABLE_RI	
0x3762	EVID_BIT_MIN_SECTION_RI	
0x3763	EVID_BIT_MAX_SUBTABLE_RI	
0x3764	EVID_NBIT_MIN_SECTION_RI	
0x3765	EVID_NBIT_MAX_SUBTABLE_RI	
0x3766	EVID_NBIT_LINK_INFO_MIN_SECTION_RI	
0x3767	EVID_NBIT_LINK_INFO_MAX_SUBTABLE_RI	
0x3768	EVID_LDT_MIN_SECTION_RI	
0x3769	EVID_LDT_MAX_SUBTABLE_RI	

Event Id	Event Name	Event Description
0x3770	EVID_SDTT_ERR	Any SDTT error
0x3771	EVID_BIT_ERR	Any BIT error
0x3772	EVID_NBIT_ERR	Any NBIT error
0x3773	EVID_NBIT_LINK_INFO_ERR	Any NBIT_LINK_INFO error
0x3774	EVID_LDT_ERR	Any LDT error
0x3775	EVID_CDT_MIN_SECTION_RI	
0x3776	EVID_CDT_MAX_SUBTABLE_RI	
0x3777	EVID_CDT_ERR	
0x3778	EVID_DCCT_MAX_SUBTABLE_RI	Maximum Subtable repetition interval error for DCCT
0x3780	EVID_SDTT_ERR_TABLE_ID	
0x3781	EVID_BIT_ERR_TABLE_ID	
0x3782	EVID_NBIT_ERR_TABLE_ID	
0x3783	EVID_CDT_ERR_TABLE_ID	
0x3790	EVID_SYNTAX_PCAT	
0x3791	EVID_SYNTAX_SDTT	
0x3792	EVID_SYNTAX_BIT	
0x3793	EVID_SYNTAX_NBIT	
0x3794	EVID_SYNTAX_NBIT_LINK_INFO	
0x3795	EVID_SYNTAX_LDT	
0x3796	EVID_SYNTAX_LIT	
0x3797	EVID_SYNTAX_ERT	
0x3798	EVID_SYNTAX_ITT	
0x3799	EVID_SYNTAX_CDT	
0x5000	Stream Mode	Stream Maintenance Mode
0x5001	EVID_GLOBAL_MMODE	
0x5200	EVID_SERVICE_LOG_OVERFLOW	
0x5201	EVID_PROCESSING_STRAINED	Processing strained
0x5202	EVID_PROCESSING_OVERWHELMED	Processing overwhelmed
0x5203	EVID_SI_STRAINED	SI storage strained
0x5204	EVID_TRAPS_THROTTLED	
0x6000	Template Header Error	Error in header block of template
0x6010	Template TS Err	Template Error in Transport Stream
0x6010	EVID_TEMPLATE_TS_ERR	
0x6020	Template NetID Err	Template Error in Network ID
0x6030	Template Orig NetID Err	Template Error in Original Network ID
0x6040	Template Service Number Err	Template Error in Program Number

Event Id	Event Name	Event Description
0x6100	Template Service Error	Error in any service header block of template
0x6110	Template Service PCR Err	Template Error with Service PCR Pid
0x6120	Template Service Type Err	Template Error with Service Type
0x6130	Template Service Name Err	Template Error with Service Name
0x6140	Template Service Constraint Err	Template Error with Service Constraint
0x6150	Template Service Pid Number Err	Template Error with Service PID Number
0x6200	Template PID Error	Error in any pid block of template
0x6210	Template Pid CA Err	Template Error with PID CA Descriptor
0x6220	Template PID Scramble Err	Template Error with PID Scrambled
0x6230	Template PID Stream Type Err	Template Error with PID Stream Type
0x6240	Template PID Constraint Err	Template Error with PID constraint
0x6240	EVID_TEMPLATE_PID_CONSTRAINT_ERR	
0x6300	Template Rating Error	Error in any rating block of template
0x6301	EVID_INDIVIDUAL_TEMPLATE_RATING_ERR	
0x6310	Template Rating Region Err	Template Error with Rating Region
0x6320	Template Rating Guidance Err	Template Error with Rating Guidance
0x6FFF	Template Master Error	Any Template Error
0xB000	EVID_STREAM_CONFIG	
0xB010	EVID_SCHEDULER	
0xB100	EVID_RECORDING_COMPLETE	