ETSI TS 103 197 V1.1.1 (2000-06)

Technical Specification

Digital Video Broadcasting (DVB); Head-end implementation of DVB SimulCrypt

European Broadcasting Union

Union Européenne de Radio-Télévision

EBU·UER

Reference

DTS/JTC-DVB-90

Keywords

broadcasting, digital, DVB, video

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by the Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

1 Scope

The present document of DVB-Simulcrypt addresses the requirements for interoperability between two or more conditional access systems at a head-end. It specifies the system architecture, timing relationships, messaging structures, extended interoperability and control.

The components within the system architecture represent functional units. The boundaries between physical units are not required to match the boundaries between functional units. It is possible that the SCS could be in the MUX or the SCS and MUX could be built independently. Neither architecture is mandated.

1.1 Common Scrambling Algorithm

The DVB-Simulcrypt group has looked at issues relating to the concepts of the common scrambling algorithm, within the DVB-Simulcrypt environment.

The DVB-Simulcrypt system is based on the concept of a shared scrambling and descrambling method. The group has looked at the possible constraints, which the DVB-Simulcrypt architecture might impose on the use of such a shared scrambling and descrambling method. No problems were noted.

1.2 Language

The word "shall" is used in a normative statement that can be verified and is mandatory. The word "should" is used in the context of a recommendation or a statement that cannot be verified or is not mandatory (it may be optional).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an ETSI EN with the same number.

[1] ETSI EN 300 468 (V1.3): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[2] ETSI ETR 154 (1997): "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications".

[3] ETSI ETR 162 (1995): "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) codes for DVB systems".

[4] ETSI ETR 211 (1997): "Digital Video Broadcasting (DVB); guidelines on implementation and usage of service information (SI)".

[5] ETSI ETR 289 (1997): "Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems".

[6] RFC 1213 (1991): "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", K. McCloghrie, M. Rose.

[7]	ISO/IEC 13818-1 (1994): "Information Technology - Generic coding of Moving Pictures and Associated Audio Recommendation H.222.0 (Systems)".
[8]	ITU-T Recommendation X.731 (1992) | ISO/IEC 10164-2 (1992): "Information technology - Open Systems Interconnection - Systems Management: State management function".
[9]	ITU-T Recommendation X.733 (1992) | ISO/IEC 10164-4 (1992): "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".
[10]	ITU-T Recommendation X.734 (1992) | ISO/IEC 10164-5 (1993): "Information technology - Open Systems Interconnection - Systems Management: Event report management function".
[11]	ITU-T Recommendation X.735 (1992) | ISO/IEC 10164-6 (1993): "Information technology - Open Systems Interconnection - Systems Management: Log control function".
[12]	RFC 768 (1980): "User Datagram Protocol, J. Postel".
[13]	RFC 791 (1981): "Internet protocol DARPA internet program protocol specification".
[14]	RFC 793 (1981): "Transmission control protocol DARPA internet program protocol specification".
[15]	RFC 1901 (1996): "Introduction to Community-based SNMPv2", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[16]	RFC 1902 (1996): "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[17]	RFC 1903 (1996): "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[18]	RFC 1904 (1996): "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[19]	RFC 1905 (1996): "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[20]	RFC 1906 (1996): "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[21]	RFC 1907 (1996): "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[22]	RFC 1908 (1996): "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", J. Case, K. McCloghrie, M. Rose, S. Waldbusser.
[23]	FIPS 46-1: "Specifications for the Data Encryption Standard".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

broadcaster (service provider): organization which assembles a sequence of events or services to be delivered to the viewer based upon a schedule.

CA_subsystem_id: CA_subsystem_id is defined in the present document to handle multiple connections to ECMGs with the same CA_system_id value. The combination of CA_system_id and CA_subsystem_id is called Super_CAS_id.

CA_system_id: CA system IDs are defined in table 3 'CA_system_id' of ETSI ETR 162 [3].

CA components: those components brought by a CA provider for integration into a host head-end system.

channel: application specific representation of an open TCP connection, allowing the association of application specific parameters with such a connection. Channels correspond on a one to one basis to TCP connections.

client: software entity on a host making use of one or more resources offered by a server.

Conditional Access (CA) system: system to control subscriber access to broadcast services and events.

Control Word (CW): data object used for scrambling.

Control Word Generator (CWG): this component receives a CW request from the SCS and returns a CW.

Crypto Period (CP): period when a particular Control Word is being used by the scrambler.

Entitlement Control Message (ECM): private Conditional Access information, which carries the control word in a secure manner and private entitlement information.

Entitlement Control Message Generator (ECMG): this generator produces the ECM messages but does not support ECM repetition. See subclause 4.2.3.

Entitlement Management Message (EMM): private Conditional Access information which, for example, specifies the authorization levels of subscribers or groups of subscribers for services or events.

Entitlement Management Message Generator (EMMG): produces the EMM messages and repeatedly plays them out at the appropriate times. See subclause 4.2.4.

forbidden: term "forbidden" when used in (the) clauses indicates that the value shall never be used.

generator: component producing data.

host: computer system uniquely identified by its IP address, and as such addressable in a computer network. It may take both client and server roles.

host head-end: system which is composed of those components required before a CA provider can be introduced into the head-end.

MPEG-2: refers to the standard ISO/IEC 13818-1 [7]. Systems coding is defined in part 1. Video coding is defined in part 2. Audio coding is defined in part 3.

multiplex: stream of all the digital data within a single physical channel carrying one or more services or events.

Multiplexer (MUX): see subclause 4.2.9.

Private Data Generator (PDG): see subclause 4.2.5.

proprietary: this term details the fact that the interface will be specified by the head-end provider, or by the CA provider. The interface can be commercially open but is not open within the present document. Its availability will be via commercial/technical agreement.

reserved: term "reserved" when used in the clause defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions. Unless otherwise specified within the present document all "reserved" bits shall be set to "1".

Rreserved future use: term "reserved_future_use", when used in the clause defining the coded bit stream, indicates that the value may be used in the future for ETSI defined extensions. Unless otherwise specified within the present document all"reserved_future_use " bits shall be set to "1".

Rresource: set of coherent functions, accessible through a server. More than one resource can reside on a single host.

Scrambler (SCR): see subclause 4.2.10.

server: software entity exporting a resource. More than one server may reside on a single host. A server is uniquely identified by an IP address and TCP port number.

service: sequence of events under the control of a broadcaster, which can be broadcast as part of a schedule.

Service Information (SI): information that is transmitted in the transport stream to aid navigation and event selection.

SI generator: see subclause 4.2.8.

Simulcrypt Integrated Management Framework (SIMF): addresses the requirements for interoperability between management components of multiple conditional access systems (CASs) at a head-end (see clause 7).

Simulcrypt Synchronizer (SCS): logical component that acquires Control Words, ECMs and synchronizes their playout for all the Conditional Access Systems connected.

stream: independent bi-directional data flow across a channel. Multiple streams may flow on a single channel. Stream_ids (e.g. ECM_stream_id, data_stream_id, ...) are used to tag messages belonging to a particular stream.

Super_CAS_id: super_CAS_id is a 32-bit identifier formed by the concatenation of the CA_system_id and the CA_subsystem_id.

Transport Stream: data structure defined in ISO/IEC 13818-1 [7]. It is the basis of the ETSI Digital Video Broadcasting (DVB) standards.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation One
bslbf	bit string, left bit first
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CIM	Common Information Model
Conf	Confirmation
CORBA	Common Object Request Broker Architecture
CP	Crypto Period
CW	Control Word
CWG	Control Word Generator
DAVIC	Digital Audio-Visual Council
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
ECM	Entitlement Control Message
ECMG	Entitlement Control Message Generator
EFD	Event forwarding discriminator
EGP	Exterior Gateway Protocol
EIS	Event Info Scheduler
EIT	Event Information Table
EMM	Entitlement Management Message
EMMG	Entitlement Management Message Generator
Id	Identifier
IDL	Interface Definition Language
Ind	Indication
IP	Internet protocol
ISO	International Organization for Standardization
JMAPI	Java Management API
LSB	Least Significant Bit
MIB	Management Information Base
MJD	Modified Julian Date
MPEG	Moving Pictures Expert Group
MUX	Multiplexer
NIT	Network Information Table
NM	Network Management
OSI	Open Systems Interconnection
PAT	Program Association Table
PDG	Private Data Generator

PID	Packet Identifier
PMT	Program Map Table
PSI	Program Specific Information
Req	Request
SCR	DVB Compliant Scrambler
SCS	Simulcrypt Synchronizer
SDT	Service Description Table
SI	Service Information
SIG	Service Information Generator
SIM	Simulcrypt Identification Module
SIMF	Simulcrypt Integrated Management Framework
SMI	Structure of management information
SMIB	Simulcrypt Management Information Base
SNMP	Simple network management protocol
ST	Stuffing Table
STB	Set Top Box
tcimsbf	two's complement integer msb (sign) bit first
TCP	Transport Control Protocol
TLV	Type, Length, Value
TMN	Telecommunications management network
UDP	User Datagram Protocol
uimsbf	unsigned integer most significant bit first
UTC	Universal Time, Co-ordinated

4 Architecture

4.1 System Architecture

Figure 1 shows the logical relationships between the components and which component-to-component interfaces are defined in the present document. Other components exist in a head-end, which are not illustrated in figure 1 (for example: SMS or Subscriber Management System…).

The DVB-Simulcrypt system architecture illustrated above is divided into 3 areas.

4.1.1 Host Head-end components

Host head-end components are those that will need to exist before Simulcrypt CA components can be introduced into a DVB-Simulcrypt head-end.

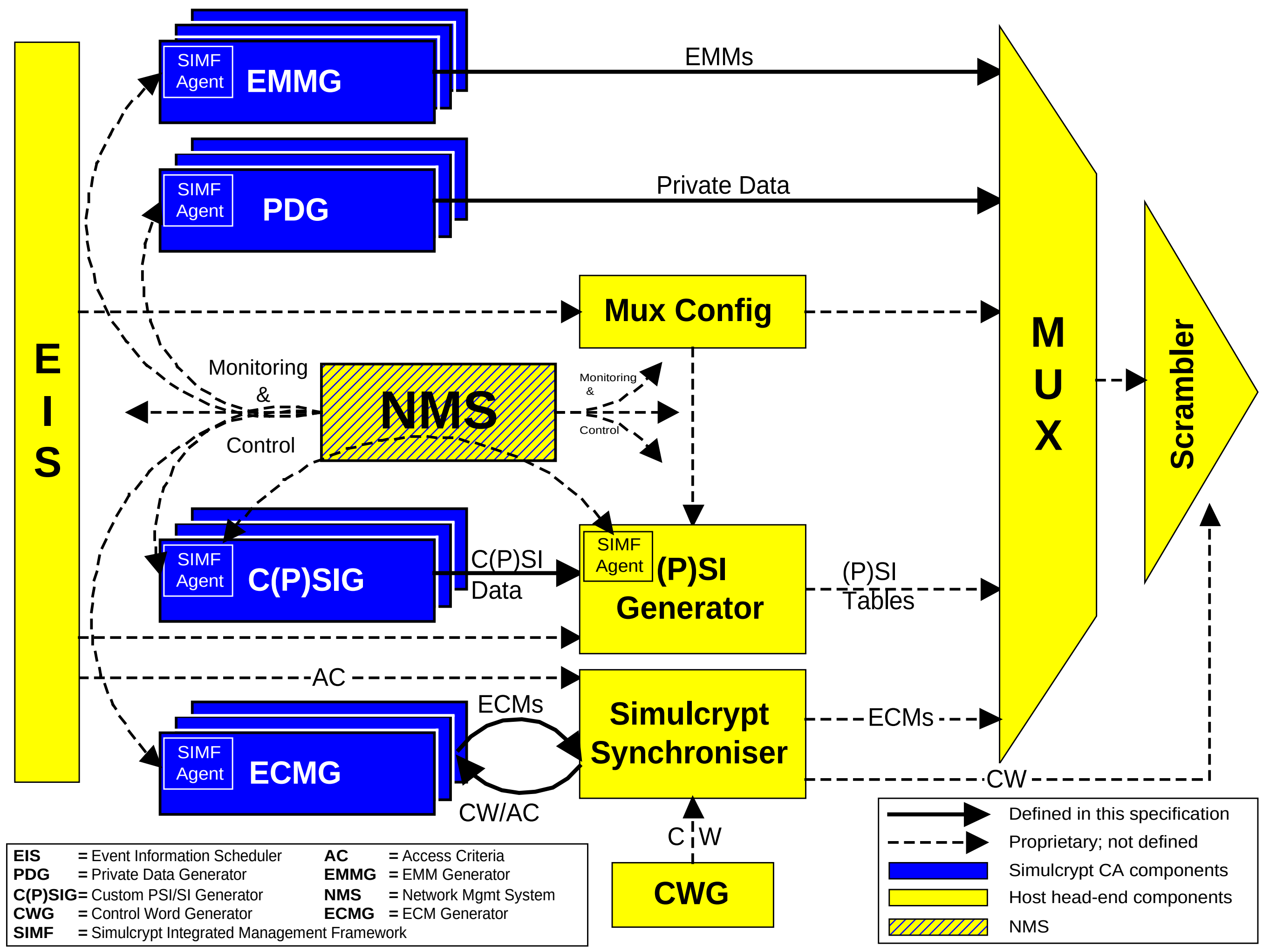
4.1.2 Simulcrypt CA components

Simulcrypt CA components are typically those, which are brought by a new CA provider to introduce his CA into a DVB-Simulcrypt head-end. Note that the EMMGs, PDGs and Custom SI generators are not necessarily required in a DVB-Simulcrypt system.

4.1.3 Simulcrypt Integrated Management Framework (SIMF)

The components of multiple conditional access systems (CASs) involved in a Simulcrypt architecture can be supported by a network management function existing in a head-end.

For ECMG, EMMG, PDG, C(P)SIG and (P)SIG, a Simulcrypt Integrated Management Framework (SIMF) is defined to address the requirements for interoperability between management components at a head-end. This framework does not address all of the issues relevant for a complete integrated Simulcrypt Management System, it specifies only the minimum set of components necessary to enable integration. SIMF is described in clause 7.



EIS	= Event Information Scheduler	**AC**	= Access Criteria
PDG	= Private Data Generator	**EMMG**	= EMM Generator
C(P)SIG	= Custom PSI/SI Generator	**NMS**	= Network Mgmt System
CWG	= Control Word Generator	**ECMG**	= ECM Generator
SIMF	= Simulcrypt Integrated Management Framework		

Figure 1: System Architecture

4.2 Description of Components

4.2.1 Event Info Scheduler (EIS)

In the DVB-Simulcrypt system architecture diagram (subclause 4.1), the EIS is the functional unit in charge of holding the entire schedule information, all the configurations and CA specific information required for the complete system. It is the overall database store for the whole head-end system. For instance, it is in charge of providing to the ECMGs (via the SCS) any information they need to generate their ECMs.

In reality this function might be distributed over several physical units, storage locations, and/or input terminals, and it may communicate with any other functional unit of the architecture diagram.

Concerning the CA provider components, the connections to the EIS and the data they carry will be agreed through the commercial arrangements made with the broadcaster. They are not defined in the present document.

4.2.2 Simulcrypt Synchronizer (SCS)

The role of the Simulcrypt Synchronizer is to:

- establish TCP connections with ECMGs and setup one channel per connection;
- setup streams within channels as needed and allocate ECM_stream_id values;
- get the control words from the CWG;

- supply the CWs to the relevant ECMGs on the relevant streams, as well as any CA specific information;
- acquire ECMs from the ECMGs;
- synchronize the ECMs with their associated Crypto periods according to channel parameters;
- submit these ECMs to the MUX and request their repetition according to the channel parameters;
- supply the CW to the scrambler for use in the specified Crypto Period.

4.2.3 ECM Generator (ECMG)

The ECMG shall receive CWs in a CW provision message as well as access criteria and shall reply with an ECM or an error message. The ECMG does not support ECM repetition.

4.2.4 EMM Generator (EMMG)

This component, supplied by the CA provider shall interface over a DVB-Simulcrypt specified interface to the MUX. The EMMG initiates connections to the MUX.

4.2.5 Private Data Generator (PDG)

This component is shown in the DVB-Simulcrypt System Architecture diagram to highlight the fact that the EMMG to MUX interface can be used for EMMs and other CA related private data. The PDG initiates connections to the MUX.

4.2.6 Custom (P)SI Generator (C(P)SIG)

This component is responsible for generating private PSI descriptors and/or private SI descriptors. It interfaces to the (P)SI Generator.

The generic term **C(P)SIG** refers to a head-end process that serves as a CPSIG, a CSIG, or both (CPSISIG).

Custom PSI Generator (CPSIG): the CA System (CAS) process(es) responsible for generating CAS-specific private data for insertion in selected MPEG-2 PSI tables.

Custom SI Generator (CSIG): the CAS process(es) responsible for generating CAS-specific private data for insertion in selected DVB SI tables.

Each CAS may (optionally) include one or more C(P)SIG.

4.2.7 MUX Config

This component is responsible for configuring the MUX and providing a link to the PSI generator for PSI construction and playout. The interfaces 'MUX Config and MUX' and 'MUX Config and PSI Generator' are not defined by the present document.

4.2.8 (P)SI Generator ((P)SIG)

This component is responsible for generating the PSI (ISO/IEC 13818-1 [7]) and/or the SI (EN 300 468 [1]) for the system. The PSI Generator and/or the SI Generator take their primary data from the EIS and supplementary data from the Custom (P)SI Generators supplied by the CA providers. The interfaces between the EIS and the (P)SI Generator and between the (P)SI Generator and the MUX are not defined by the present document.

The generic term **(P)SIG** refers to a head-end process that serves as a PSIG, an SIG, or both PSISIG.

PSI Generator (PSIG): the head-end process(es) responsible for generating MPEG-2 PSI (Program Specific Information) tables.

SI Generator (SIG): the head-end process(es) responsible for generating DVB SI (Service Information) tables.

NOTE: The NIT (Network Information Table) is considered a DVB SI table.

If the head-end supports the C(P)SIG ⇔ (P)SIG interface, it shall include one or more (P)SIG.

4.2.9 Multiplexer (MUX)

The role of this head-end component is to perform time multiplexing of input data, and to output an MPEG-2 transport stream. The input data can be transport packets, MPEG sections or raw data. The exact functionality of a MUX is implementer specific. For the purpose of the present document, the MUX shall be able to communicate with the SCS, the (P)SIG and to accept connections with EMMGs with the interface defined.

4.2.10 Scrambler (SCR)

This component is responsible for scrambling data in the MPEG2 Transport Ttream. The exact functionality of the Scrambler is implementer specific. For the purpose of the present document, the Scrambler shall be able to receive Control Words from the SCS.

4.2.11 Control Word Generator (CWG)

This component is responsible for generating control words used in scrambler initialization stream. The exact functionality of the Scrambler is implementer specific. For the purpose of the present document, the Control Word Generator shall be able to provide the SCS with control words.

4.2.12 Network Management System (NMS)

This component is responsible for monitoring and control of SIMF agents. The exact nature of this function depends on the type of host component the agent is situated in, i.e. ECMG, EMMG, PDG, etc, and the type of management function the NMS component is performing, i.e. fault, configuration, accounting, performance and security management.

4.2.13 SIMF agent

This component supports network management protocol transactions on the Simulcrypt Management Information Base (SMIB), which it implements. It instruments the SMIB with monitoring and control functionality of the host component, i.e. ECMG, EMMG, PDG, etc.

4.3 Description of interfaces

4.3.1 ECMG ↔ SCS

The interface allowing a CAS to provide a SCS with ECMs under the control of this SCS. This interface is mandatory and shall be implemented as described in clause 5.

4.3.2 EMMG ↔ MUX

The interface allowing a CAS to provide a MUX with EMM under the control of the CAS. This interface is mandatory and shall be implemented as described in clause 6.

4.3.3 PDG ↔ MUX

The interface used is the EMMG ⇔ MUX interface.

4.3.4 Custom (P)SI Generator ↔ (P)SI Generator

The interface allowing a CAS to provide a (P)SIG with private data descriptors for the head-end to insert in (P)SI tables. This interface is mandatory and shall be implemented as described in clause 8.

4.3.5 EIS ↔ SI Generator

Proprietary, not defined by the present document.

4.3.6 SI Generator ↔ MUX

Proprietary, not defined by the present document.

4.3.7 EIS ↔ MUX Config

Proprietary, not defined by the present document.

4.3.8 MUX Config ↔ PSI Generator

Proprietary, not defined by the present document.

4.3.9 PSI Generator ↔ MUX

Proprietary, not defined by the present document.

4.3.10 MUX ↔ SCR

Proprietary, not defined by the present document.

4.3.11 SCR onward

Proprietary, not defined by the present document.

4.3.12 SCS ↔ MUX

Proprietary, not defined by the present document.

4.3.13 SCS ↔ SCR

Proprietary, not defined by the present document.

4.3.14 SCS ↔ CWG

Proprietary, not defined by the present document.

4.3.15 EIS ↔ SCS

Proprietary, not defined by the present document.

4.3.16 NMS Component ↔ SIMF Agent

This interface allows a network management function existing in a head-end to support Simulcrypt CA components. It is defined by the Simulcrypt MIB and the NM protocol used in the system.

This interface is optional; if used it shall be implemented as described in clause 7.

4.3.17 Mandatory or optional characteristics of the interfaces

The subclauses 4.3.1 to 4.3.16 give the global mandatory or optional characteristic of each interface described in the present document. Table 1a sums up these characteristics.

Table 1a: Mandatory or optional characteristics of the Simulcrypt interfaces

Interfaces	Global characteristic	Particular points	See subclauses
ECMG ⇔ SCS	mandatory	only TCP based implementation Security of control words: • CW security shall be supported; the means or methods to support this security are chosen by commercial agreement; one of them is the CW encryption in the protocol; • CW encryption in the protocol is optional if CWs are encrypted in the protocol, the method given in annex D is recommended.	5.1.7
EMMG ⇔ MUX	mandatory	a real implementation is chosen by the head-end operator among: • TCP based implementation for data provision and control; • UDP based implementation for data provision and TCP based implementation for control; • UDP based implementation for data provision and SIMF based implementation for control.	6.1
C(P)SIG ⇔ (P)SIG	mandatory	according to the same application protocol model, a real implementation is defined by commercial agreement among: • TCP based implementation; • SIMF based implementation.	8.2
NMS Component ⇔ SIMF Agent	optional	the Simulcrypt Network Management function implementation is optional and is defined by the head-end operator among: • SNMP v2 for agents and manager; this implementation is fully defined in the present document; • SNMP v2 for agents and CORBA for manager; in this case only the SNMP v2 compliant SMIB is described in the present document.	7.1

4.4 Protocol types

4.4.1 Connection-oriented protocols

For the connection-oriented protocols defined in the present document, the messages shall have the following generic structure:

Table 1b: Message-type values for command/response-based protocols

```
generic_message
{
    protocol_version    1 byte
    message_type    2 bytes
    message_length  2 bytes
    for (i=0; i < n; i++)
    {

        parameter_type  2 bytes
        parameter_length    2 bytes
        parameter_value <parameter_length> bytes
    }
}
```

NOTE 1: For parameters with a size two or more bytes the first byte to be transmitted will be the most significant byte.

NOTE 2: Parameters do not need to be ordered within the generic message.

protocol_version: A 8 bit field identifying the protocol version. It shall have the value 0x02.

message_type: A 16 bit field identifying the type of the message. The list of message type values is defined in table 2. Unknown message types shall be ignored by the receiving entity.

message_length: 16-bit field specifies the number of bytes in the message immediately following the message_length field.

parameter_type: 16-bit field specifies the type of the following parameter. The list of parameter type values is defined in the interface specific sections of the present document. Unknown parameters shall be ignored by the receiving entity. The data associated with that parameter will be discarded and the remaining message processed.

parameter_length: 16-bit field specifies the number of bytes of the following parameter_value field.

parameter_value: variable length field specifies the actual value of the parameter. Its semantics is specific to the parameter type value.

Table 2: Message-type values for command/response-based protocols

Relevant interface	Message_type value	Message type
DVB reserved	0x0000	DVB reserved
ECMG ⇔ SCS	0x0001 0x0002 0x0003 0x0004 0x0005	channel_setup channel_test channel_status channel_close channel_error
DVB reserved	0x0006 to 0x0010	DVB reserved
EMMG ⇔ MUX	0x0011 0x0012 0x0013 0x0014 0x0015	channel_setup channel_test channel_status channel_close channel_error
DVB reserved	0x0016 to 0x0100	DVB reserved
ECMG ⇔ SCS	0x0101 0x0102 0x0103 0x0104 0x0105 0x0106	stream_setup stream_test stream_status stream_close_request stream_close_response stream_error
DVB reserved	0x107 to 0x110	DVB reserved
EMMG ⇔ MUX	0x0111 0x0112 0x0113 0x0114 0x0115 0x0116 0x0117 0x0118	stream_setup stream_test stream_status stream_close_request stream_close_response stream_error stream_BW_request stream_BW_allocation
DVB reserved	0x0119 to 0x0200	DVB reserved
ECMG ⇔ SCS	0x0201 0x0202	CW_provision ECM_response
DVB reserved	0x0203 to 0x0210	DVB reserved
EMMG ⇔ MUX	0x0211	data_provision
DVB reserved	0x0212 to 0x0300	DVB reserved
C(P)SIG ⇔ (P)SIG	0x0301 0x0302 0x0303 0x0304 0x0305	channel_setup channel_status channel_test channel_close channel_error
DVB reserved	0x0306 to 0x0310	DVB reserved
C(P)SIG ⇔ (P)SIG	0x0311 0x0312 0x0313 0x0314 0x0315 0x0316 0x0317 0x0318 0x0319 0x031A 0x031B 0x031C 0x031D 0x031E 0x031F 0x0320 0x0321	stream_setup stream_status stream_test stream_close stream_close_request stream_close_response stream_error stream_service_change stream_trigger_enable_request stream_trigger_enable_response trigger table_request table_response descriptor_insert_request descriptor_insert_response PID_provision_request PID_provision_response
DVB reserved	0x0322 to 0x7FFF	DVB reserved
user defined	0x8000 to 0xFFFF	user defined

4.4.2 SIMF-based protocols

All Simulcrypt CAS/Head-end interface specifications such as ECMG ⇔ SCS, EMMG ⇔ Mux, PDG ⇔ Mux, and C(P)SIG ⇔ (P)SIG are based on a connection-oriented communications paradigm (i.e. channel/streams). Alternatively, all these interfaces can be implemented using a transaction-based communications paradigm using the SIMF. In the current Simulcrypt specification this alternative is only available:

- for the C(P)SIG ⇔ (P)SIG interface;
- for the control of UDP-based protocol on EMMG/PDG ⇔ Mux interface.

5 ECMG ↔ SCS interface

5.1 Interface principles

5.1.1 Channel and Stream specific messages

This interface shall carry the following channel messages defined further in subclause 5.4:

- Channel_setup
- Channel_test
- Channel_status
- Channel_close
- Channel_error

and the following stream messages defined further in subclause 5.5:

- Stream_setup
- Stream_test
- Stream_status
- Stream_close_request
- Stream_close_response
- Stream_error
- CW_provision
- ECM_response

For this interface, the SCS is the client and the ECMG is the server. The SCS has a prior knowledge of the mapping between Super_CAS_ids and the IP addresses and port numbers of the ECMGs. When a new ECM stream is requested by the EIS for a given Super_CAS_id value, the SCS will open a new stream with the appropriate ECMG. This might require the opening of a new channel (which involves the opening of a new TCP connection).

When a new ECM stream is created in a transport stream, a new ECM_id shall be assigned to it by the head-end, according to the operational context (ECM stream creation or ECMG replacement). The value of the ECM_id parameter remains unmodified as long as the ECM stream exists. The combination {« ECM » type + Super_CAS_id + ECM_id} identifies uniquely this new ECM stream in the whole system.

NOTE: There can be several ECMGs associated with the same Super_CAS_id value (e.g. for performance or redundancy reasons). In such a case the SCS should be able to choose with which ECMG the connection will be opened, based on either a redundancy policy or resource available.

5.1.2 Channel establishment

There is always one (and only one) channel per TCP connection. Once the TCP connection is established, the SCS sends a channel_setup message to the ECMG. In case of success the ECMG replies by sending back a channel_status message.

In case of a rejection or a failure during channel setup the ECMG replies with a channel_error message. This means that the channel has not been opened by the ECMG and the SCS shall close the TCP connection.

5.1.3 Stream establishment

The SCS sends a stream_setup message to the ECMG. In case of success the ECMG replies by sending back a stream_status message. In case of a rejection or a failure the ECMG replies with a stream_error message.

Once the connection, channel and stream have been correctly established the ECM will be transferred. It can be transferred as sections or as TS packets. The ECMG indicates at channel setup which kind of data object will be used.

Once the connection, channel and stream have been correctly established, ECMs will be transferred to the SCS as a response to the CW_provision message.

5.1.4 Stream closure

Stream closure is always initiated by the SCS. This can occur when an ECM stream is no longer needed or in the case of an error. This is done by means of a stream_close_request message. A stream_close_response message indicates the stream has been closed.

5.1.5 Channel closure

Channel closure can occur when the channel is no longer needed or in case of error (detected by SCS or reported by ECMG). This is done by means of a channel_close message sent by the SCS. Subsequently, the connection shall be closed by both sides.

5.1.6 Channel/Stream testing and status

At any moment either component can send a channel_test/stream_test message to check the integrity of a channel/stream. In response to this message the receiving component shall reply with either a channel/stream status message or a channel/stream error message.

5.1.7 Unexpected communication loss

Both SCS and ECMG shall be able to handle an unexpected communication loss (either on the connection, channel or stream level).

Each component, when suspecting a possible communication loss (e.g. a 10 second silent period), should check the communication status by sending a test message and expecting to receive a status message. If the status message is not received in a given time (implementation specific) the communication path should be re-established.

5.1.8 Handling data inconsistencies

If the ECMG detects an inconsistency it shall send an error message to the SCS. If the SCS receives such a message or detects an inconsistency it may close the connection. The SCS (as the client) will then (re-)establish the connection, channel and (if applicable) streams.

NOTE: The occurrence of a user defined or unknown parameter_type or message_type shall not be considered as an inconsistency.

5.2 Parameter_type values

Table 3: ECMG protocol parameter_type values

Parameter_type Value	Parameter type	Type/units	Length (bytes)
0x0000	DVB Reserved	-	-
0x0001	Super_CAS_id	uimsbf	4
0x0002	section_TSpkt_flag	Boolean	1
0x0003	delay_start	tcimsbf/ms	2
0x0004	delay_stop	tcimsbf/ms	2
0x0005	transition_delay_start	tcimsbf/ms	2
0x0006	transition_delay_stop	tcimsbf/ms	2
0x0007	ECM_rep_period	uimsbf/ms	2
0x0008	max_streams	uimsbf	2
0x0009	min_CP_duration	uimsbf/n x 100ms	2
0x000A	lead_CW	uimsbf	1
0x000B	CW_per_msg	uimsbf	1
0x000C	max_comp_time	uimsbf/ms	2
0x000D	access_criteria	user defined	variable
0x000E	ECM_channel_id	uimsbf	2
0x000F	ECM_stream_id	uimsbf	2
0x0010	nominal_CP_duration	uimsbf/n x 100ms	2
0x0011	access_criteria_transfer_mode	Boolean	1
0x0012	CP_number	uimsbf	2
0x0013	CP_duration	uimsbf/n x 100ms	2
0x0014	CP_CW_Combination	---	variable
	CP	uimsbf	2
	CW	uimsbf	variable
0x0015	ECM_datagram	user defined	variable
0x0016	AC_delay_start	tcimsbf/ms	2
0x0017	AC_delay_stop	tcimsbf/ms	2
0x0018	CW_encryption	user defined	variable
0x0019	ECM_id	uimsbf	2
0x001A to 0x6FFF	DVB reserved	-	-
0x7000	Error_status	see subclause 5.6	2
0x7001	Error_information	user defined	variable
0x7002 to 0x7FFF	DVB reserved	-	-
0x8000 to 0xFFFF	User defined	-	-

5.3 Parameter semantics

AC_delay_start: this parameter shall be used in place of the delay start parameter for the first Crypto period following a change in AC.

AC_delay_stop: this parameter shall be used in place of the delay stop parameter for the last Crypto period preceding a change in AC.

access_criteria: this parameter contains CA system specific information of undefined length and format, needed by the ECMG to build an ECM. It can be, for example, a pointer to an access criterion in an ECMG database, or a list of relevant access criteria items in an encapsulated TLV format. This parameter contains the information related to the CP indicated in the CW_provision message. The presence and contents of the access criteria parameter are the result of CA system supplier requirements.

access_criteria_transfer_mode: this 1-byte parameter is a flag. If it equals 0, it indicates that the access_criteria parameter is required in the CW_provision message only when the contents of this parameter change. If it equals 1, it indicates that the ECMG requires the access_criteria parameter be present in each CW_provision message.

CP_CW_combination: this parameter is the concatenation of the Crypto period number the control word is attached to and the control word itself. The parity (odd or even) of the Crypto Period number is equal to the parity of the corresponding control word (see ETSI ETR 289 [5]). This parameter is typically 10 byte long.

CP_duration: this parameter indicates the actual duration of a particular Crypto period for a particular stream when it differs from the nominal_CP_duration value (see definition below).

CP_number: an identifier to a Crypto period. This parameter indicates the Crypto period number a message is attached to. This is relevant for the following messages: CW_provision, and ECM_response.

CW_encryption: this parameter enables encrypting of control words over the SCS ⇔ ECMG interface. If the parameter is included in the CW_provision message, control word scrambling is invoked; if omitted, CWs are being issued in the clear. This parameter may include sub-parameters according to the used encrypting method. It may be used by the CW security method described in annex D or by an equivalent method.

CW_per_msg: the number of control words needed by the ECMG per control word provision message. If this value is 'y' and lead_CW is 'x', each control word provision message attached to Crypto period 'n' will contain all control words from period (*n*+1+x-y) to period (*n*+x). Control words are carried with their Crypto period number by means of the CP_CW_combination parameter. In most existing CA systems CW_per_msg is 1 or 2. See also lead_CW.

For example, if an ECMG requires the current and next control word to generate an ECM, it shall by definition specify at least one lead_CW. However, since it may buffer its own control words, it can set CW_per_msg to one. By doing this, it always receives the control word for the *next* Crypto Period and accessing the control word for the current Crypto Period from memory (a previous provision message). Alternatively, it may specify 2 CW_per_msg and have both control words available at ECM generation time. This eliminates the need for ECMG buffering and can be advantageous for a hot backup to take over, since each provision message includes all control words required.

An SCS shall minimally support CW_per_msg values 1 and 2.

delay_start: this signed integer represents the amount of time between the start of a Crypto Period, and the start of the broadcasting of the ECM attached to this period. If it is positive, it means that the ECM shall be delayed with respect to the start of the Crypto Period. If negative, it means that the ECM shall be broadcast ahead of this time. This parameter is communicated by the ECMG to the SCS during the channel setup.

delay_stop: this signed integer represents the amount of time between the end of a Crypto Period, and the end of the broadcasting of the ECM attached to this period. If it is positive, it means that the end of the ECM broadcast shall be delayed with respect to the end of the Crypto Period. If negative, it means that the ECM broadcast shall be ended ahead of time. This parameter is communicated by the ECMG to the SCS during the channel setup.

ECM_channel_id: the ECM_channel_id is allocated by the SCS and uniquely identifies an ECM channel across all connected ECMGs.

ECM_id: the ECM_id is allocated by the head-end and uniquely identifies an ECM stream for a Super_CAS_id. The combination of the « ECM » type, the Super_CAS_id and the ECM_id identifies uniquely an ECM stream in the whole system. The unique identifier principle is described in subclause 8.2.7.

ECM_datagram: the actual ECM message to be passed by the SCS to the MUX. It can be either a series of transport packets (of 188 byte length) or an MPEG-2 section, according to the value of section_TSpkt_flag. The ECM datagram can have a zero length meaning that there is no ECM to be broadcast for the crypto period. The ECM datagram shall comply with ETSI ETR 289 [5].

ECM_rep_period: this integer represents the period in milliseconds for the repetition of data (e.g. ECMs).

ECM_stream_id: this identifier uniquely identifies an ECM stream within a channel. It is allocated by the SCS prior to stream setup.

error_status: see subclause 5.6.

error_information: this optional parameter contains user defined data completing the information provided by error_status. It can be an ASCII text or the parameter ID value of a faulty parameter for example.

lead_CW: the number of control words required in advance to build an ECM. If this value is 'x' the ECMG requires control words up to Crypto Period number '*n*+x' to build the ECM attached to Crypto period '*n*'. In most existing CA systems lead_CW is 0 or 1. See also CW_per_msg.

For example, if the ECMG requires the current and next control word to generate an ECM, lead_CW would be 1. In other words, it defines the most future control word required for ECM generation.

An SCS shall minimally support lead_CW values 0 and 1.

max_comp_time: this parameter is communicated by the ECMG to the SCS during channel setup. It is the worst case time needed by an ECMG to compute an ECM when all the streams in a channel are being used. This time is typically used by the SCS to decide when to time-out on the ECM_response message. This value shall be lower than the min_CP_duration parameter of the same channel_status message.

max_streams: maximum number of simultaneous opened streams supported by an ECMG on a channel. This parameter is communicated from the ECMG to the SCS during the channel setup. A value of 0 means that this maximum is not known.

min_CP_duration: this parameter is communicated at channel setup by the ECMG to the SCS to indicate the minimum supported amount of time a control word shall be active before it can be changed. This value shall be greater than the max_comp_time parameter of the same channel_status message.

nominal_CP_duration: this parameter indicates the nominal duration of Crypto periods for the particular stream. It means that all the Crypto periods related to this stream will have this duration, except for the purpose of event alignments and error handling. Even in these exceptional cases, all the actual Crypto periods shall have a duration greater than or equal to the nominal_CP_duration. In addition, the nominal Crypto period duration (chosen by SCS) shall be greater than or equal to:

- all the min_CP_duration specified by the ECMGs during channel_setup;
- all the max_comp_time values specified by the ECMGs during channel setup, plus typical network latencies.

section_TSpkt_flag: this parameter defines the format of the ECM carried on this interface:

- **0x00**: the ECMs carried on the interface are in MPEG-2 section format;
- **0x01**: the ECMs carried on the interface are in MPEG-2 transport stream packet format, all TS packets shall be 188 byte long, any other payload length being considered as an error; it is the head-end's responsibility to fill the PID field in TS packet header;
- other values: DVB reserved.

Super_CAS_id: the Super_CAS_id is a 32-bit identifier formed by the concatenation of the CA_system_id (16 bit) and the CA_subsystem_id (16 bit). It shall identify uniquely a (set of) ECMG(s) for a given SCS, see subclause 4.3.1. The CA_subsystem_id is defined by the user, it is private.

transition_delay_start: this parameter shall be used in place of the delay start parameter for the first crypto period following a clear to scrambled transition.

transition_delay_stop: this parameter shall be used in place of the delay stop parameter for the last crypto period preceeding a scrambled to clear transition.

5.4 Channel specific Messages

5.4.1 Channel_setup message: ECMG ← SCS

Parameter	Number of instances in message
ECM_channel_id	1
Super_CAS_id	1

The channel_setup message (message_type = 0x0001) is sent by the SCS to setup a channel once the TCP connection has been established, as described in subclause 5.1.2. It shall contain the Super_CAS_id parameter, to indicate to the ECMG to which CA system and subsystem the channel is intended (indeed, there could be several Super_CAS_ids handled by a single ECMG host).

5.4.2 Channel_test message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1

The channel_test message (message_type = 0x0002) can be sent at any moment by either side to check:

- the channel is in an error free situation;
- the TCP connection is still alive.

The peer shall reply with a channel_status message if the channel is free of errors, or a channel_error message if errors occurred.

5.4.3 Channel_status message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1
section_TSpkt_flag	1
AC_delay_start	0/1
AC_delay_stop	0/1
delay_start	1
delay_stop	1
transition_delay_start	0/1
transition_delay_stop	0/1
ECM_rep_period	1
max_streams	1
min_CP_duration	1
lead_CW	1
CW_per_msg	1
max_comp_time	1

The channel_status message (message_type = 0x0003) is a reply to the channel_setup message or the channel_test message (see subclauses 5.1.2 and 5.1.6).

When the message is a response to a setup, the values of the parameters are those requested by the ECMG. All these parameter values will be valid during the whole lifetime of the channel, for all the streams running on it.

When the message is a response to a test, the values of the parameters shall be those currently valid in the channel.

5.4.4 Channel_close message: ECMG ← SCS

Parameter	Number of instances in message
ECM_channel_id	1

The channel_close message (message_type = 0x0004) is sent by the SCS to indicate the channel is to be closed.

5.4.5 Channel_error message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1
error_status	1 to n
error_information	0 to n

A channel_error message (message type = 0x0005) is sent by the recipient of a channel_test message or by the ECMG at any time to indicate that an unrecoverable channel level error occurred. A table of possible error conditions can be found in subclause 5.6.

5.5 Stream specific messages

5.5.1 Stream_setup message: ECMG ← SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
ECM_id	1
nominal_CP_duration	1

The stream_setup message (message type = 0x0101) is sent by the SCS to setup a stream once the channel has been established, as described in subclause 5.1.3.

5.5.2 Stream_test message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1

The stream_test message (message_type = 0x0102) is used to request a stream_status message for the given ECM_channel_id and ECM_stream_id. The stream_test message can be sent at any moment by either entity. The peer shall reply with a stream_status message if the stream is free of errors, or a stream_error message if errors occurred.

5.5.3 Stream_status message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
ECM_id	1
access_criteria_transfer_mode	1

The stream_status message (message_type = 0x0103) is a reply to the stream_setup message or the stream_test message.

When the message is a response to a setup, the value of the access_criteria_transfer_mode parameter is the one requested by the ECMG.

When the message is a response to a test, the values of the parameters shall be those currently valid in the stream.

5.5.4 Stream_close_request message: ECMG ← SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1

The ECM_stream_id is sent by the SCS in the stream_close_request message (message type = 0x0104) to indicate which of the streams in a channel is due for closure.

5.5.5 Stream_close_response message: ECMG → SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1

The ECM_stream_id is sent by the ECMG in the stream_close_response message (message type = 0x0105) to indicate which of the streams in a channel is closing.

5.5.6 Stream_error message: ECMG ↔ SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
error_status	1 to n
error_information	0 to n

A stream_error message (message type = 0x0106) is sent by the recipient of a stream_test message or by the ECMG at any time to indicate that an unrecoverable stream level error occurred. A table of possible error conditions can be found in subclause 5.6.

5.5.7 CW_provision message: ECMG ← SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
CP_number	1
CW_encryption	0 to 1
CP_CW_combination	CW_per_msg.
CP_duration	0 to 1
access_criteria	0 to 1

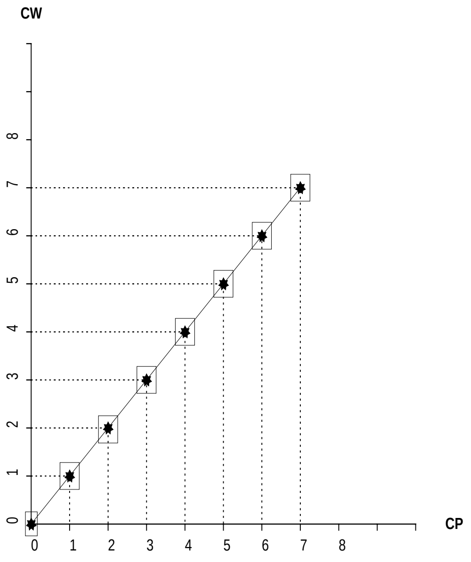
CW_provision message (message_type 0x201) is sent by the SCS to the ECMG and serves as a request to compute an ECM. The value of the CP_number parameter is the Crypto period number of the requested ECM. The control words are carried by this message with their associated Crypto period numbers in the CP_CW_combination parameter, according to the value of lead_CW and CW_per_msg as defined during the channel setup. For instance, if lead_CW = 1 and CW_per_msg = 2, the CW_provision message for Crypto period N shall contain control words for Crypto periods N and N+1.

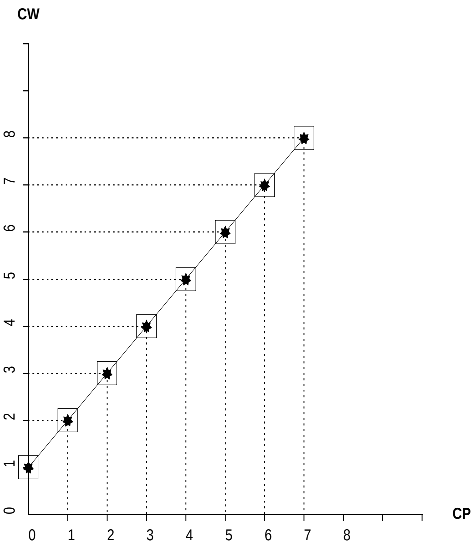
The SCS is not allowed to send a CW_provision message before having received the ECM_response message for the previous Crypto periods, except if there has been a time-out expiration, or an error message (in which case the way this error is handled is left to the discretion of the SCS manufacturer).

The specific CWs that are passed in the CP_CW_combination to the ECMG via the CW_provision message are derived from the values of lead_CW and CW_per_msg. The following table shows a number of different values these parameters can take to achieve different ECMG requirements.

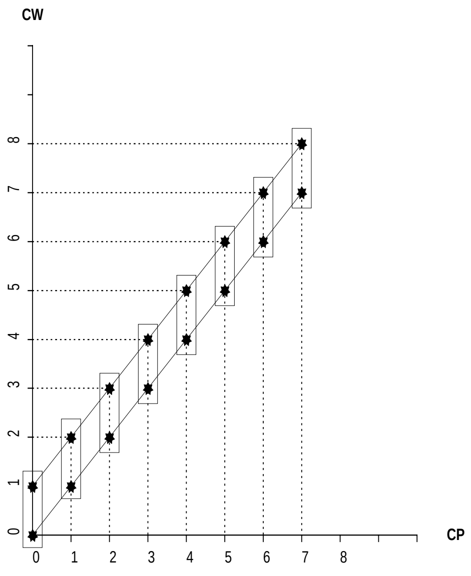
Example	Requirements	lead_CW	CW_per_msg
1	1 CW per ECM per CP	0	1
2	the CWs for the current and next CP per ECM and the ECMG buffers the current CW from the previous CW Provision message	1	1
3	the CWs for the current and next CP per ECM and the ECMG receives both CWs from the SCS in each CW Provision message	1	2
4	3 CWs per ECM per CP	1	3

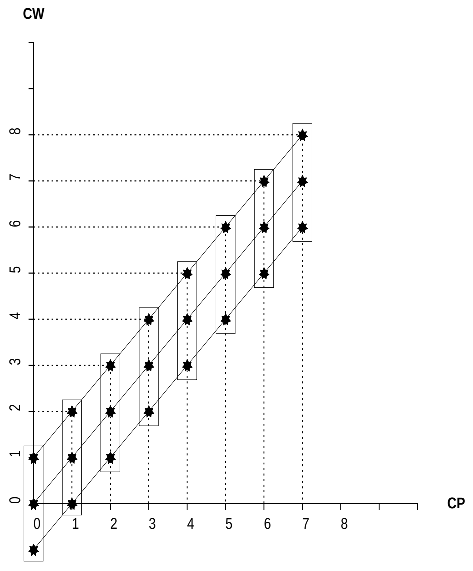
These graphs depict which CWs has to be passed for a specific CP, based on the different methods listed above. For any given CP, X-axis, the corresponding CW is portrayed on the Y-axis. In the CW_provision message, the boxed CWs are the set of CP_CW_combination that has to be passed.





Example 1 Example 2





Example 3 Example 4

Figure 2: Lead_CW to CW_per_msg relationship

5.5.8 ECM_response message: ECMG → SCS

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
CP_number	1
ECM_datagram	1

The ECM_response message (message_type 0x202) is a reply to the CW_provision message. It carries the ECM datagram, computed by the ECMG, from the information provided by the CW_provision message (and possibly from other CA specific information). The value of the CP_number parameter shall be the same in the replied ECM_response message as in the previous incoming CW_provision message (on that stream).

The time-out for the ECM_response message shall be computed by the SCS from the max_comp_time value defined during channel setup, and the typical network delays.

5.6 Error status

NOTE: TCP connection level errors are beyond the scope of the present document. Only channel, stream and application level errors are dealt with. These errors occur during the lifetime of a TCP connection.

There are two different error messages on these interfaces. The channel_error message for channel wide errors and the stream_error message for stream specific errors. These messages are sent by the ECMG to the SCS. When the ECMG reports an error to the SCS, it is up to the SCS to decide the most appropriate step to be taken. However 'unrecoverable error' explicitly means that the channel or stream (depending on the message used) has to be closed. Most of the error status listed in the table 4 cannot occur in normal operation. They are mainly provided to facilitate the integration and debugging phase.

Table 4: ECMG protocol error values

error_status value	Error type
0x0000	DVB Reserved
0x0001	invalid message
0x0002	unsupported protocol version
0x0003	unknown message_type value
0x0004	message too long
0x0005	unknown Super_CAS_id value
0x0006	unknown ECM_channel_id value
0x0007	unknown ECM_stream_id value
0x0008	too many channels on this ECMG
0x0009	too many ECM streams on this channel
0x000A	too many ECM streams on this ECMG
0x000B	not enough control words to compute ECM
0x000C	ECMG out of storage capacity
0x000D	ECMG out of computational resources
0x000E	unknown parameter_type value
0x000F	inconsistent length for DVB parameter
0x0010	missing mandatory DVB parameter
0x0011	invalid value for DVB parameter
0x0012	unknown ECM_id value
0x0013	ECM_channel_id value already in use
0x0014	ECM_stream_id value already in use
0x0015	ECM_id value already in use
0x0016 to 0x6FFF	DVB Reserved
0x7000	unknown error
0x7001	unrecoverable error
0x7002 to 0x7FFF	DVB Reserved
0x8000 to 0xFFFF	ECMG specific / CA system specific / User defined

5.7 Security in ECMG ↔ SCS protocol

The control words conveyed in the CP_CW_combination parameter within the CW_provision message constitute the clear cryptographic keys that are used to directly scramble content. Knowledge of these keys by unauthorized agents can result in the compromise of the security of the broadcast service. Therefore it is incumbent upon all Simulcrypt participants to employ effective and appropriate methods to preserve the confidentiality of the control words traversing this interface. One approach is to use only an inherently secure network for the ECMG ⇔ SCS interface. Another is to use a control word encryption scheme such as the one recommended in annex D of the present document to deliver the CW to the ECMG in a secure manner. In any case, the security of the CW on this interface shall be maintained so that unauthorized interception is prevented.

6 EMMG ↔ MUX and PDG ↔ MUX interfaces

6.1 Transport layer protocols for EMMG/PDG ↔ MUX interfaces

To facilitate co-existence the DVB Simulcrypt Specification provides both TCP and UDP protocols for the EMMG/PDG ⇔ MUX interface. This co-existence is not required within the same head-end. In certain head-ends the UDP protocol may be more suitable (e.g. for network performance reasons) and in other head-ends the TCP protocol may be more suitable (e.g. for network reliability reasons). Therefore it is the head-end operator that decides which protocol is more appropriate and should be followed.

6.2 TCP-based protocol

6.2.1 Interface principles

6.2.1.1 Channel and Stream specific messages

The interface shall carry the following channel messages defined further in subclause 6.2.4:

- Channel_setup
- Channel_test
- Channel_status
- Channel_close
- Channel_error

and the following stream messages defined further in section 6.2.5:

- Stream_setup
- Stream_test
- Stream_status
- Stream_close_request
- Stream_close_response
- Stream_error
- Data_provision

For this interface, the EMMG/PDG is the client and the MUX is the server. In this TCP-based protocol, all messages are sent using TCP.

6.2.1.2 Channel establishment

The EMMG/PDG sends a channel_setup message to the MUX. In case of success the MUX replies by sending back a channel_status message.

In case of a rejection or a failure during channel setup the MUX replies with a channel_error message. This means that the channel has not been opened by the MUX and the EMMG/PDG shall close the TCP connection.

6.2.1.3 Stream establishment

The EMMG/PDG sends a stream_setup message to the MUX. In case of success the MUX replies by sending back a stream_status message. In case of a rejection or a failure the MUX replies with a stream_error message.

When a new EMM/private data stream is created in a transport stream, a new data_id shall be assigned to it by the CAS, according to the operational context (EMM/private data stream creation or EMMG/PDG replacement). The value of the data_id parameter remains unmodified as long as the EMM/private data stream exists. The combination {stream type + client_id + data_id} identifies uniquely this new EMM/private data stream in the whole system.

Once the connection, channel and stream have been correctly established the EMMs/private data will be transferred. They can be transferred as sections or as TS packets. The EMMG/PDG indicates at channel setup which kind of data object will be used. The EMMs/private data shall be inserted in the transport stream in the same order as they are provided by the EMMG/PDG.

6.2.1.4 Bandwidth allocation

The interface allows bandwidth negotiation between the EMMG/PDG and the MUX. This is not mandatory. During stream setup the EMMG/PDG will request the optimal bandwidth for that stream. The MUX will then respond with the bandwidth that has been allocated for that stream. The EMMG/PDG can, at a later time, request an adjustment in the bandwidth allocation. The MUX could also initiate an allocation change, without any request from the EMMG/PDG.

6.2.1.5 Stream closure

Stream closure is always initiated by the EMMG/PDG. This can occur when an EMM/private data stream is no longer needed. This is done by means of a stream_close_request message. A stream_close_response message indicates the stream has been closed.

6.2.1.6 Channel closure

Channel closure can occur when the channel is no longer needed or in case of error (detected by EMMG/PDG or reported by MUX). This is done by means of a channel_close message sent by the EMMG/PDG. Subsequently, the connection shall be closed by both sides.

6.2.1.7 Channel/Stream testing and status

At any moment either component can send a channel_test/stream_test message to check the integrity of a channel/stream. In response to this message the receiving component shall reply with either a channel/stream status message or a channel/stream error message.

6.2.1.8 Unexpected connection loss

Both EMMG/PDG and MUX shall be able to handle an unexpected communication loss (either on the connection, channel or stream level).

Each component, when suspecting a possible communication loss (e.g. a 10 second silent period), should check the communication status by sending a test message and expecting to receive a status message. If the status message is not received in a given time (implementation specific) the communication path should be re-established.

6.2.1.9 Handling data inconsistencies

If the MUX detects an inconsistency it shall send an error message to the EMMG/PDG. If the EMMG/PDG receives such a message or detects an inconsistency it should close the connection. The EMMG/PDG (as the client) will then (re-) establish the connection, channel and (if applicable) streams.

NOTE: The occurrence of a user defined or unknown parameter_type or message_type shall not be considered as an inconsistency.

6.2.2 Parameter Type Values

Table 5: EMMG/PDG protocol parameter_type values

Parameter_type value	Parameter type	Type / Units	Length (bytes)
0x0000	DVB Reserved	-	-
0x0001	client_id	uimsbf	4
0x0002	section_TSpkt_flag	boolean	1
0x0003	data_channel_id	uimsbf	2
0x0004	data_stream_id	uimsbf	2
0x0005	datagram	user defined	variable
0x0006	bandwidth	uimsbf / kbit/s	2
0x0007	data_type	uimsbf	1
0x0008	data_id	uimsbf	2
0x0009 to 0x6FFF	DVB Reserved	-	-
0x7000	error_status	see subclause 6.2.6	2
0x7001	error_information	user defined	variable
0x7002 to 0x7FFF	DVB reserved	-	-
0x8000 to 0xFFFF	user defined	-	-

6.2.3 Parameter semantics

bandwidth: this parameter is used in stream_BW_request and stream_BW_allocation messages to indicate the requested bit rate or the allocated bit rate respectively. It is the responsibility of the EMMG/PDG to maintain the bit rate generated within the limits specified by the MUX when the bandwidth allocation method is used (optional). It should be noted that the EMMG/PDG will operate from 0 kbit/s to the negotiated rate. The EMMG/PDG will not exceed the negotiated rate. If the bandwidth allocation method is not used the responsibility of bit rate control is not defined in the present document.

client_id: the client_id is a 32-bit identifier. It shall identify uniquely an EMMG/PDG across all the EMMGs/PDGs connected to a given MUX. To facilitate uniqueness of this value, the following rules apply:

- in the case of EMMs or other CA related data, the two first bytes of the client_id should be equal to the two bytes of the corresponding CA_system_id;

- in other cases a value allocated by DVB for this purpose should be used.

data_stream_id: this identifier uniquely identifies an EMM/private data stream within a channel.

data_channel_id: this identifier uniquely identifies an EMM/private data channel within a client_id.

data_id: the data_id is allocated by the CAS and uniquely identifies an EMM/private data stream of a client_id. The combination of the client_id and the data_id identifies uniquely an EMM/private data stream in the whole system. The unique identifier principle is described in subclause 8.2.7.

data_type: type of data carried in the datagram in the stream:

- **0x00**: EMM;

- **0x01:** private data;

- **0x02**: DVB reserved (ECM);

- other values: DVB reserved.

datagram: this is the EMM/private data. The Datagram can be transferred in either section or TS packet format according to the value of section_TSpkt_flag.

error_status: see subclause 6.2.6.

error_information: this optional parameter contains user defined data completing the information provided by error_status. It can be an ASCII text or the parameter ID value of a faulty parameter for example.

section_TSpkt_flag: this parameter defines the format of the EMMs or of the private datagrams carried on this interface:

- **0x00**: the EMMs or private datagrams are in MPEG-2 section format;
- **0x01**: the EMMs or private datagrams are in MPEG-2 transport stream packet format; all TS packets shall be 188 byte long, any other payload length being considered as an error; it is the head-end's responsibility to fill the PID field in TS packet header;
- other values: DVB reserved.

6.2.4 Channel specific messages

6.2.4.1 Channel_setup message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
section_TSpkt_flag	1

The channel_setup message (message_type = 0x0011) is sent by the EMMG/PDG to the MUX to setup a channel once the TCP connection has been established, as described in subclause 6.2.1.2. It shall contain the client_id parameter indicating to the MUX the EMMG/PDG that is opening the channel.

6.2.4.2 Channel_test message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1

The channel_test message (message_type = 0x0012) can be sent at any moment by either side to check:

- the channel is in an error free situation;
- the TCP connection is alive.

The peer shall reply with a channel_status message if the channel is free of errors, or a channel_error message if errors occurred.

6.2.4.3 Channel_status message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
section_TSpkt_flag	1

The channel_status message (message_type = 0x0013) is a reply to the channel_setup message or the channel_test message (see subclauses 6.2.1.2 and 6.2.1.7). All the parameters listed above are mandatory.

When the message is a response to a setup, the values of the parameters are those requested by the MUX. All these parameter values will be valid during the whole lifetime of the channel, for all the streams running on it.

When the message is a response to a test, the values of the parameters shall be those currently valid in the channel.

6.2.4.4 Channel_close message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1

The channel_close message (message_type = 0x0014) is sent by the EMMG/PDG to indicate the channel is to be closed.

6.2.4.5 Channel_error message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
error_status	1 to n
error_information	0 to n

A channel_error message (message type = 0x0015) is sent by the recipient of a channel_test message or by the MUX at any time to indicate that an unrecoverable channel level error occurred. A table of possible error conditions can be found in subclause 6.2.6.

6.2.5 Stream specific messages

6.2.5.1 Stream_setup message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1
data_id	1
data_type	1

The stream_setup message (message_type = 0x0111) is sent by the EMMG/PDG to setup a stream once the channel has been established, as described in subclause 6.2.1.3.

6.2.5.2 Stream_test message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1

The stream_test message (message_type = 0x0112) is used to request a stream_status message for the given client_id, data_channel_id and data_stream_id. The stream_test message can be sent at any moment by either entity. The peer shall reply with a stream_status message if the stream is free of errors, or a stream_error message if errors occurred.

6.2.5.3 Stream_status message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1
data_id	1
data_type	1

The stream_status message (message_type = 0x0113) is a reply to the stream_setup message or the stream_test message.

The values of the parameters shall be those currently valid in the stream.

6.2.5.4 Stream_close_request message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1

The stream_close_request message (message_type = 0x0114) is sent by the EMMG/PDG to indicate the stream is to be closed.

6.2.5.5 Stream_close_response message: EMMG/PDG ← MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1

The stream_close_response message (message_type = 0x0115) is sent by the MUX indicating the stream that is being closed.

6.2.5.6 Stream_error message: EMMG/PDG ↔ MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1
error_status	1 to n
error_information	0 to n

A stream_error message (message type = 0x0116) is sent by the recipient of a stream_test message or by the MUX at any time to indicate that an unrecoverable stream level error occurred. A table of possible error conditions can be found in subclause 6.2.6.

6.2.5.7 Stream_BW_request message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1
bandwidth	0 to 1

The stream_BW_request message (message type = 0x0117) is always sent by the EMMG/PDG and can be used in two ways.

If the bandwidth parameter is present the message is a request for the indicated amount of bandwidth.

If the bandwidth parameter is not present the message is just a request for information about the currently allocated bandwidth. The MUX shall always reply to this message with a stream_BW_allocation message.

6.2.5.8 Stream_BW_allocation message: EMMG/PDG ← MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	1
data_stream_id	1
bandwidth	0 to 1

The stream_BW_allocation message (message type = 0x0118) is used to inform the EMMG/PDG about the bandwidth allocated. This can be a response to a stream_BW_request message or as a notification of a change in bandwidth initiated by the MUX. The message is always sent by the MUX.

If the bandwidth parameter is not present it means that the allocated bandwidth is not known.

NOTE: The bandwidth allocation message may indicate a different bandwidth than was requested (this could be less).

6.2.5.9 Data_provision message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	0 to 1
data_stream_id	0 to 1
data_id	1
datagram	1 to n

The data_provision message is used by the EMMG/PDG to send, on a given data_stream_id, the datagram (in the case of EMMG this is EMM data).

In the TCP-based protocol, the data_provision message shall include the data_channel_id and data_stream_id parameters. In the UDP-based protocol (see subclause 6.3), the data_provision message shall not include the data_channel_id and data_stream_id parameters.

6.2.6 Error status

NOTE: TCP connection level errors are beyond the scope of the present document. Only channel, stream and application level errors are dealt with. These errors occur during the lifetime of a TCP connection.

There are two different error messages on that interface. The channel_error message for channel wide errors, and the stream_error message for stream specific errors. These messages are sent by the MUX to the EMMG/PDG. When the MUX reports an error to the EMMG/PDG, it is up to the EMMG/PDG to decide the most appropriate step to be taken. However 'unrecoverable error' explicitly means that the channel or stream (depending on the message used) has to be closed. Most error status listed in table 6 cannot occur in normal operation. They are mainly provided to facilitate the integration and debugging phase.

Table 6: EMMG/PDG protocol error values

error_status value	Error type
0x0000	DVB Reserved
0x0001	invalid message
0x0002	unsupported protocol version
0x0003	uUnknown message_type value
0x0004	message too long
0x0005	unknown data_stream_id value
0x0006	unknown data_channel_id value
0x0007	too many channels on this MUX
0x0008	too many data streams on this channel
0x0009	too many data streams on this MUX
0x000A	unknown parameter_type
0x000B	inconsistent length for DVB parameter
0x000C	missing mandatory DVB parameter
0x000D	invalid value for DVB parameter
0x000E	unknown client_id value
0x000F	exceeded bandwidth
0x0010	unknown data_id value
0x0011	data_channel_id value already in use
0x0012	data_stream_id value already in use
0x0013	data_id value already in use
0x0014	client_id value already in use
0x0015 to 0x6FFF	DVB Reserved
0x7000	unknown error
0x7001	unrecoverable error
0x7002 to 0x7FFF	DVB Reserved
0x8000 to 0xFFFF	MUX specific / CA system specific / User defined

6.3 UDP-based protocol

The EMMG/PDG UDP-based protocol is using the same message format as described in subclause 6.2. Thanks to UDP functionality, it offers the unique advantage of broadcasting packets over a network and requires less network overhead; however, the EMMG/PDG UDP-based protocol does not provide additional feature to increase the intrinsic reliability of UDP.

6.3.1 Interface principles

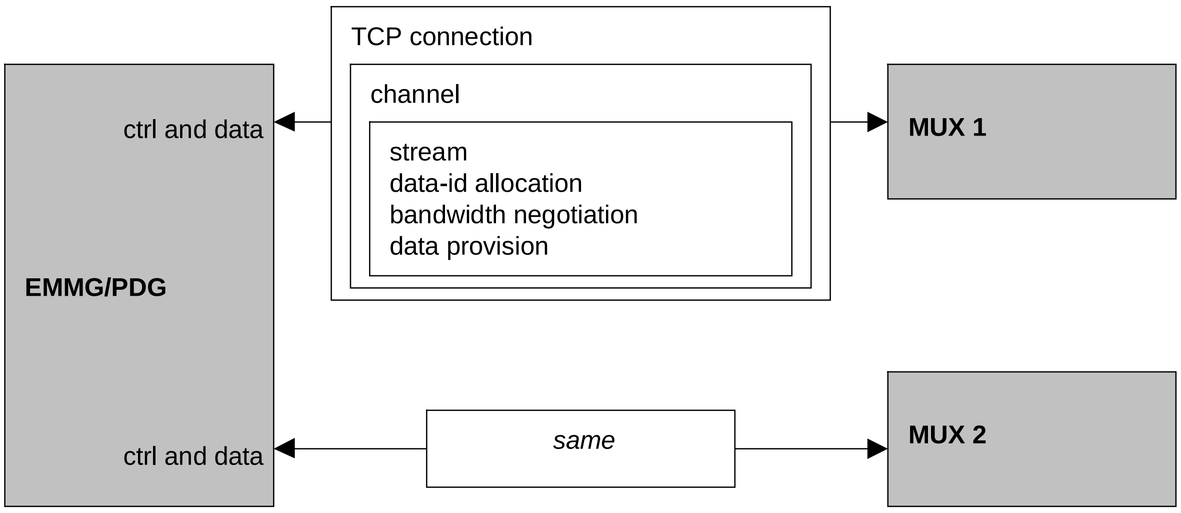
In this protocol, only the data_provision message is sent using UDP/IP; it is the same message as the one used in the TCP-based protocol described in subclause 6.2.5.9.

For the control part of this protocol, two methods can be used. The implementation of at least one of them is mandatory:

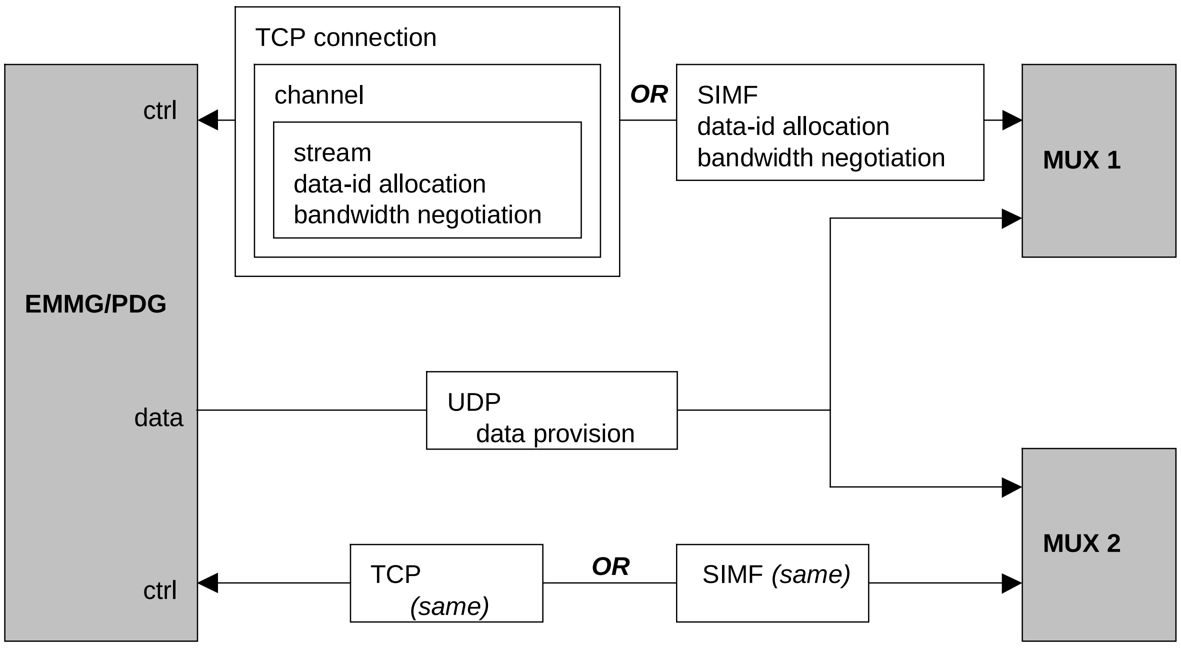
- the first method consists of using a TCP/IP connection supporting channel and stream management, as described in suclause 6.2.1. This connection carries also test, status, error and bandwidth negotiation messages. When UDP/IP broadcast is used to send EMMs or Private Data, it is the responsibility of the EMMG/PDG to open a TCP/IP connection with each multiplexer that needs to receive the UDP packets produced by the EMMG/PDG;

- the second method consists of using the Simulcrypt Management Framework described in clause 7. With this method, the Network Management System (NMS) is responsible for ensuring that the head-end components are configured properly to receive and process the EMM/Private data produced by the EMMG/PDG.

The figure 3 illustrates the TCP-based version and the UDP-based version of the EMMG/PDG protocol.



TCP-based protocol



UDP-based protocol

Figure 3: TCP-based and UDP-based EMMG/PDG protocols

6.3.1.1 Data_provision message: EMMG/PDG → MUX

Parameter	Number of instances in message
client_id	1
data_channel_id	0 to 1
data_stream_id	0 to 1
data-id	1
datagram	1 to n

The data provision message is used by the EMMG/PDG to send, on a given data_id the datagram (in the case of EMMG this is EMM data). Data_channel_id and data_stream_id are optional parameters. The client_id/data_id pair shall identify in a unique manner an EMM/private data stream across the system. For example, if two EMMGs send an EMM stream with the same data_id to the same multiplexer port, the multiplexer shall be able to distinguish between the two streams by looking at the client_id field in the data_provision message.

This message is the only message sent over UDP/IP and can be broadcast to several multiplexers.

6.3.1.2 Channel and stream configuration messages

Except for the data_provision message, all messages described in subclauses 6.2.4 and 6.2.5 can be used on a separate TCP/IP connection to manage channels and streams. If a broadcast mechanism is used to send data_provision messages, the client_id, data_stream_id and data_id parameters will be the same for all the Multiplexers processing the EMM/Private data stream.

For this interface, the EMMG is the client, and the MUX is the server. Please refer to subclause 6.2 (TCP-based protocol) for syntax details.

6.3.2 Bandwidth management

When using channel and stream configuration messages with UDP/IP broadcast data_provision messages, the same bandwidth negociation messages need to be sent to all Multiplexers processing the EMM/private data stream:

- where TCP connections are used to manage configurations, it is the responsibility of the EMMG/PDG to ensure that all Multiplexers accept a new bandwidth configuration before changing the actual EMM/private data bandwidth. Multiplexers cannot be held responsible for overflow conditions on UDP/IP sockets;

- where SIMF is being used to manage configurations, it is the responsibility of the Network Management System (NMS) to ensure that all configuration changes are synchronized properly among the network components. For example, bandwidth increases shall first occur on the Multiplexers, while bandwidth decreases shall first occur on the EMMG/PDG to avoid loss of data.

7 Network management

7.1 SIMF overview

The Simulcrypt Integrated Management Framework (SIMF) addresses the requirements for interoperability between management components of multiple conditional access systems (CASs) at a head-end. The framework does not address all of the issues relevant for a complete integrated Simulcrypt Management System. It specifies only the minimum set of components necessary to enable integration.

All Simulcrypt CAS/head-end interface specifications such as ECMG ⇔ SCS, EMMG ⇔ Mux, PDG ⇔ Mux, and C(P)SIG ⇔ (P)SIG are defined based on a connection-oriented communications paradigm (i.e. channel/streams). Alternatively, all these interfaces can be implemented using a transaction-based communications paradigm using the SIMF. In the current Simulcrypt specification this alternative is only available for the C(P)SIG ⇔ (P)SIG interface.

The basic specification principle of the SIMF is:

- to define a common information model for management of all Simulcrypt conditional access components within the head-end, which are directly related to the Simulcrypt protocol. Based on the Simulcrypt Common Information Model (CIM) the Simulcrypt Management Information Base (SMIB) is defined. The CIM defines management information specific to Simulcrypt conditional access components of a head-end such as the EMM Generators (EMMGs), ECM Generators (ECMGs), Custom Service Information Generators (CSIG) and Custom Program Specific Information Generators (CPSIG). The CIM also defines generic information enabling management facilitating functions such as event and alarm specification and logging;

- to define a management protocol for manager/agent communication.

7.1.1 Introduction to the Common Information Model (CIM)

The CIM provides a sufficiently large common denominator to ease the integration of basic management functions fault, configuration, and performance management into a head-end network manager or a conditional access system manager. Specifically, the CIM consists of the following information:

1) configuration and status information of the following Simulcrypt conditional access components:
   - EMM Generators (EMMG);
   - Private Data Generators (PDG);
   - ECM Generators (ECMG);
   - (P)SI Generator ((P)SIG);
   - C(P)SI Generator (C(P)SIG).
2) generic event reporting information including:
   - event specification information;
   - alarm, state change, and value change notifications;
   - event forwarding information.
3) generic log control mechanism including:
   - log specification information;
   - alarm, state change, value change logs;
   - log filtering information.

The CIM is implemented in the Simulcrypt Management Information Base (SMIB), which within the framework is realized in an open-ended fashion. This is necessary to not constrain a particular head-end in the choice of implementation technologies but to also simultaneously enable integration by providing a standard MIB.

The management functions themselves are not mandated but are only enabled.

Thus, the DVB Simulcrypt Integrated Management Framework (SIMF) standardizes management information access via a management protocol but does not specify how the management information is to be used. The framework consists of the following.

1) The DVB Simulcrypt Management Information Base (SMIB)- using the basic concepts and vocabulary of the ITU-T TMN Information Model and the standard Internet SNMPv2 SMI, the SMIB consists of four modules:
   - MIB II as defined in RFC 1213 [6];
   - the Simulcrypt Identification Module (SIM);
   - the Simulcrypt Events Module (SEM);
   - the Simulcrypt Logs Module (SLM).
2) The management protocol to be used.

A CIM description is given in subclauses 7.2 and 7.3.

7.1.2 SIMF specialization options

The framework enables the design and implementation of a complete integrated management system by specifying the common management information in the SMIB and a management protocol. To enable maximum flexibility in choosing the most appropriate technology for individual head-ends and facilitate open-endedness the following two specializations of the SIMF are defined of which one shall be chosen if a management system is to be supported by a head-end (it is the head-end operator who decides which options are more appropriate):

Option 1 - SNMP:

- the SMIB is realized as an SNMPv2 SMI;
- the management protocol is SNMPv2.

Option 2 - CORBA (not available for agents) (note):

- the SMIB is realized as an IDL translation of the Option 1 SMI SMIB using the Joint X/Open/NMF (JIDM) specification;
- the management protocol is defined as the SNMPv2 equivalent based on the JIDM specification.

NOTE: There is no obligation on a head-end management system to support CORBA agents. Only SNMP agents are always supported if the Simulcrypt Integrated Management Framework is implemented.

The two options facilitate a variety of possible DVB Simulcrypt Management Systems including all current major approaches as follows:

1) CORBA, TMN, TINA-C - see the OMG White Paper "CORBA-BASED Telecommunication Network Management Systems" which relies on the JIDM specification;

2) JMAPI - JMAPI supports both SNMP and CORBA;

3) WBEM - WBEM builds on top of existing frameworks.

The present document fully defines only the SNMPv2 and SNMPv2 SMI based SMIB.

The Simulcrypt Identification Module (SIM), the Simulcrypt Events Module (SEM) and the Simulcrypt Logs Module (SLM) can be used to support configuration, performance, and fault management of Simulcrypt components such as EMMG, ECMG, etc. While SIM is Simulcrypt specific, SEM and SLM also support generic notifications and logs and are applicable to any proprietary information modules. As such they can extend management functionality to the proprietary components if so desired by the component provider and the head-end facilitator.

7.2 The Common Information Model

The Common Information Model (CIM) definition is closely tied to SNMPv2 and SNMPv2 SMI, as this is the first option for a network management system realization, which can be transformed into the second one by means of the JIDM process. The Common Information Model specification maps directly into an SNMPv2 SMI MIB and therefore also includes the Internet Assigned Numbers Authority (IANA) Simulcrypt object number assignments.

The Common Information Model consists of four modules:

- MIB II as defined in RFC 1213 [6];
- the Simulcrypt Identification Module (SIM);
- the Simulcrypt Events Module (SEM);
- the Simulcrypt Logs Module (SLM).

The Simulcrypt Identification Module (SIM) reflects the Simulcrypt interfaces specified (e.g. ECMG/SCS). Therefore it is specified after these interfaces have been specified. The other three modules are used in all systems and are the pre-requisite for management of any Simulcrypt interface or the implementation of a Simulcrypt interface using the transaction based approach, i.e. C(P)SI/(P)SI interface.

The individual modules shall be implemented by Simulcrypt CA components if an integrated network management system is desired at the head-end and if a component is to be managed or monitored or if the interface is to be implemented using the transaction based transport. The technology underlying the module implementation can be different from component to component provided a JIDM based translator is provided to the common denominator platform of the head-end. For example, if the common denominator platform is CORBA and the module is implemented in an ECM generator as an SNMP MIB then a JIDM gateway needs to be provided between CORBA/IIOP and SNMP. Not all network elements will have the need for all information groups defined in the modules.

7.2.1 Object Containment Hierarchy

Objects are unambiguously identified (or named) by assigning them an object identifier (OID). OIDs are globally unique for the entire Internet and are defined as a sequence of non-negative integers organized hierarchically similar to UNIX or DOS file system names. Each sequence element is also associated a textual name for ease of use. The last sequence name is commonly used by itself as a shorthand way of naming an object. Although there is no uniqueness requirement for shorthand names within the Internet-standard framework, by convention, an attempt is made to make those names unique by using a different prefix for objects in each new MIB. Figure 4 illustrates the Internet OID tree.

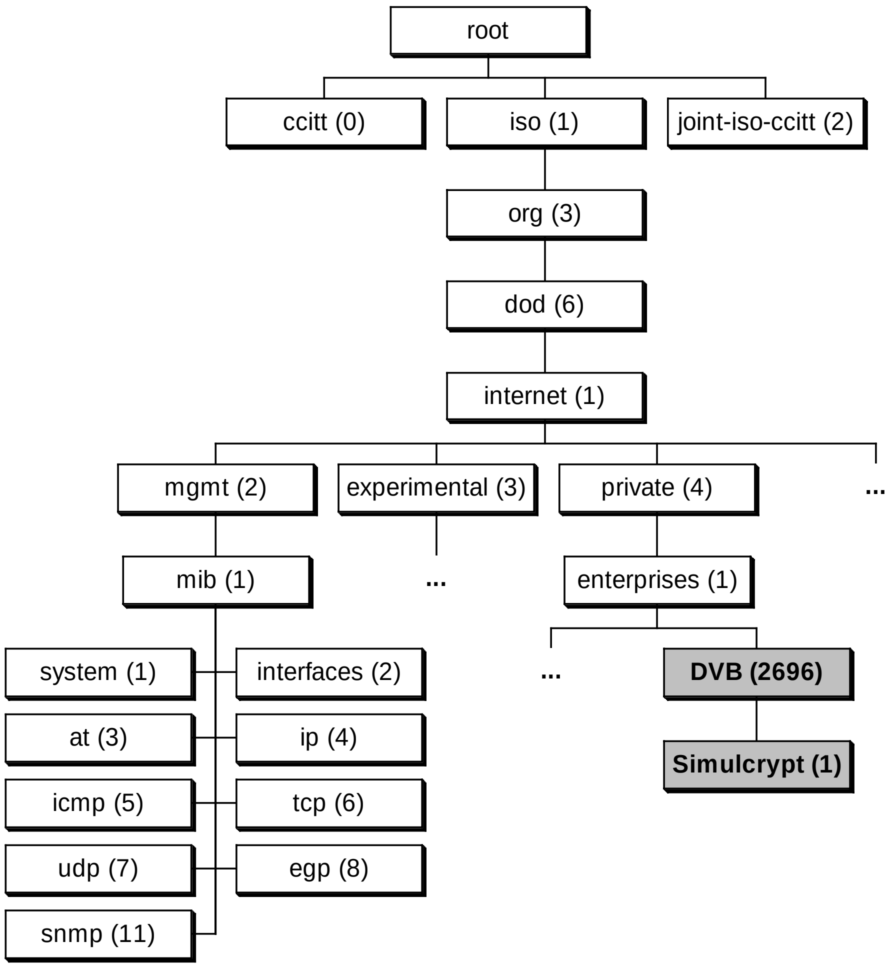


Figure 4: The Standard Internet OID Tree

Different OID formats can be used by interchanging the component names and numbers. For example:

- {iso org(3) dod(6) internet(1)} and 1.3.6.1 define the same object which is the Internet;
- {internet 4} and 1.3.6.1.4 define the same object which is the Private branch of the Internet subtree;
- {tcp 4} and 1.3.6.1.1.2.1.6.4 define the same object which is the TCP MIB module of the standard Internet management MIB.

For SNMP MIBs the following OID prefixes are important:

- internet defined as {iso(1) org(3) dod(6) 1};
- mgmt defined as {internet 2};
- experimental defined as {internet 3};

- private defined as {internet 4};
- mib, mib-1, and mib-2 which are defined as {mgmt 1};
- enterprises which is defined as {private 1}.

Objects for standard SNMP MIBs are defined under the "mib" branch of the hierarchy. "mib-1" has been superseded by "mib-2". Experimental MIBs being developed by IETF working groups define objects under the "experimental" branch. Proprietary MIBs define objects within an organization's subtree located under the "enterprises" branch (assigned by the Internet Assigned Numbers Authority - IANA).

The number assigned to DVB by IANA is **2696**. DVB Simulcrypt is assigned the first branch under DVB, which is **{enterprises 2696 1}** that corresponds to **1.3.6.1.4.1.2696.1**.

The DVB Simulcrypt MIB Modules are located as follows:

- {simMIB} which is 1.3.6.1.4.1.2696.1.1;
- {semMIB} which is 1.3.6.1.4.1.2696.1.2;
- {slmMIB} which is 1.3.6.1.4.1.2696.1.3.

Figure 5 illustrates the Simulcrypt MIB tree.

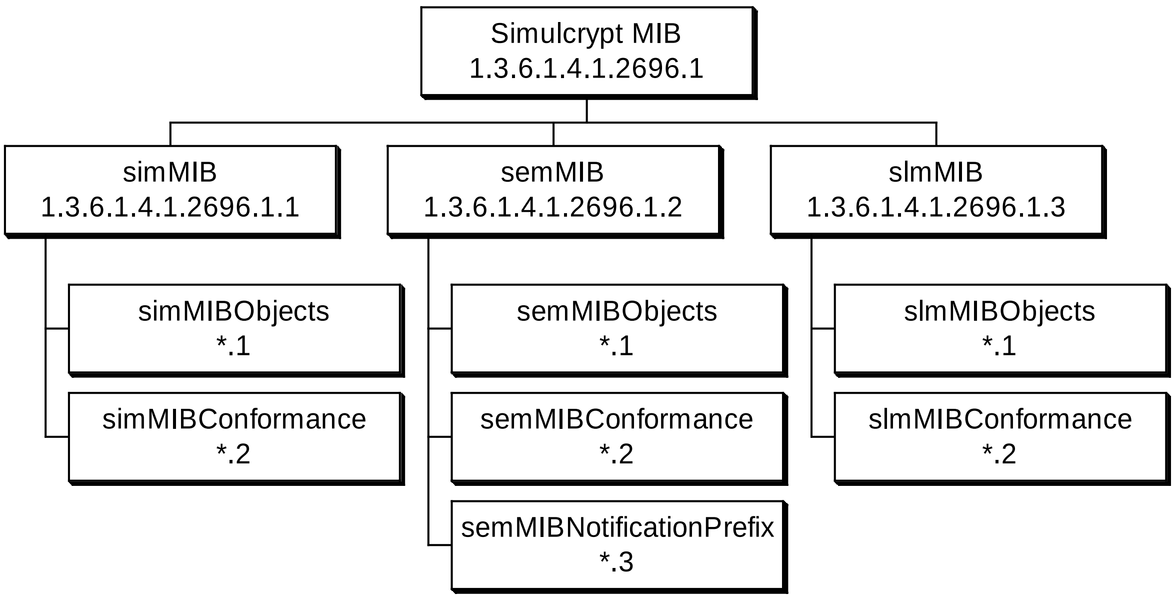


Figure 5: DVB Simulcrypt MIB Tree

7.2.2 MIB II

MIB II is a standard Internet MIB (see RFC 1213 [6]). Every SNMP agent is assumed to support MIB II. MIB-II, like its predecessor, the Internet-standard MIB, contains only essential elements. There is no need to allow individual objects to be optional. Rather, the objects are arranged into groups which are implemented by a system depending on whether the semantics of the group is applicable to an implementation. For example, an implementation has to implement the EGP group if and only if it implements the EGP. Following Groups are defined:

- System - Implementation of the System group is mandatory for all systems. The System group specifies things like system up time, system contact, system location, etc. If an agent is not configured to have a value for any of these variables, a string of length 0 is returned.
- Interfaces - Implementation of the Interfaces group is mandatory for all systems. This group defines the different subnetwork interfaces from this entity, i.e. Ethernet, FDDI, etc.
- Address Translation (obsolete).

- IP - Implementation of the IP Group is mandatory for all systems. This group specifies information related to IP like routing tables, address mappings, statistics, etc.

- ICMP - Implementation of the ICMP group is mandatory for all systems. This group specifies a number of different type of statistics related to ICMP.

- TCP - Implementation of the TCP group is mandatory for all systems that implement the TCP. This group contains TCP configuration information and information about the existing TCP connections. Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

- UDP - Implementation of the UDP group is mandatory for all systems, which implement the UDP. This group contains UDP related statistics as well as information of all current UDP listeners.

- EGP - Implementation of the EGP group is mandatory for all systems, which implement the EGP.

- Transmission - Based on the transmission media underlying each on a system, the corresponding portion of the Transmission group is mandatory for that system. When Internet-standard definitions for managing transmission media are defined, the transmission group used to provide a prefix for the names of those. Typically, such definitions reside in the portion of the MIB until they are "proven", then as a part of the Internet standardization process, definitions are accordingly elevated and a new identifier, under the transmission group is defined. By convention, the name assigned is: type OBJECT IDENTIFIER: = {transmission number}, where "type" is the symbolic value used for the media in the ifType column of the ifTable object, and "number" is the actual integer value corresponding to the symbol.

- SNMP - Implementation of the SNMP group is mandatory for all systems which support an SNMP protocol entity. Some of the objects defined will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station.

7.2.3 Concurrency Control

Concurrency control of simultaneous updates of an agent's MIB variables by multiple managers is accomplished by using the SMIB SIM module's administrative state variables (as defined in the ITU-T Recommendation X.731 [8] standard used in TMN) and the SNMP administrative framework. An administrative state of a group of objects is either locked, unlocked, or shutting down. These states and the corresponding state transitions are defined in ITU-T Recommendation X.731 [8].

A manager can only lock a variable group if it is part of the agent's, module's, etc. management community and if the agent, module, etc. is not already locked. Once an agent is locked it is protected from access by other managers as any manager wanting to access the concurrency-controlled agent, module, etc. will first attempt to lock it and will fail if that is not possible (i.e. managers are trusted). If a manager should crash leaving the agent, module, etc. locked, the designated back-up managers within the same community can unlock the agent.

7.2.4 The Simulcrypt Events Module (SEM)

SEM enables managers to configure the types of events that can be generated by an agent and when those events should be transformed into asynchronous notifications (SNMP Traps) to be sent to different managers. The mechanism and information model used are based on the ITU-T Recommendations X.733 [9] and X.734 [10] standards defining event management and alarm reporting. SEM defines the following object groups:

- Events Group - This group consists of event configuration information defining the types of events that the agent shall generate.

- Event Forwarding Discriminator (EFD) Group - This group consists of EFD configuration information defining what types of events an EFD will transform into notifications, at what times of day it will do so, and to which managers it will send the notifications to.

- Event Notifications - This group defines three types of notifications, which an agent can send to a manager. These are an alarm, a state change notification, and a value change notification. Each EFD specifies what type of notification is to be sent for an event that has occurred in the agent. The EFD also specifies the conditions under which such a notification is to be sent and the IP address of the manager to which the notification is to be sent. All standard SNMP traps are sent to the managers UDP port 162. Most management platforms support this mechanism.

The Events Group stores event descriptions in a table. Each row in the table corresponds to an event that the agent is to generate. The event description in that table specifies when the agent is to generate the event, i.e. because a MIB variable has crossed a specified threshold, because a state has been changed, etc.

Once the agent generates an event as specified in the Event table it checks the EFD Table to find an EFD that matches that event and specifies what kind of notification is to be generated and to which manager that notification is to be sent. The match is performed based on event characteristics such as event type etc.

The EFDs in the EFD Table are controlled by three state/status variables, the administrative state, the operational state, and the availability status. If the administrative state is not unlocked or the operational state is not enabled or the availability status is not available, the EFD is inactive (that means it is ignored by the agent). The manager sets the administrative and operational states. The availability status is set as a result of an automatic scheduling function that is also associated with the EFD and specified in the EFD table. This scheduling function includes specifications of a daily start and stop time and a weekly mask specifying when the EFD changes availability status from off-duty to available.

Figure 6 illustrates the MIB tree of the Simulcrypt Events Module (SEM):

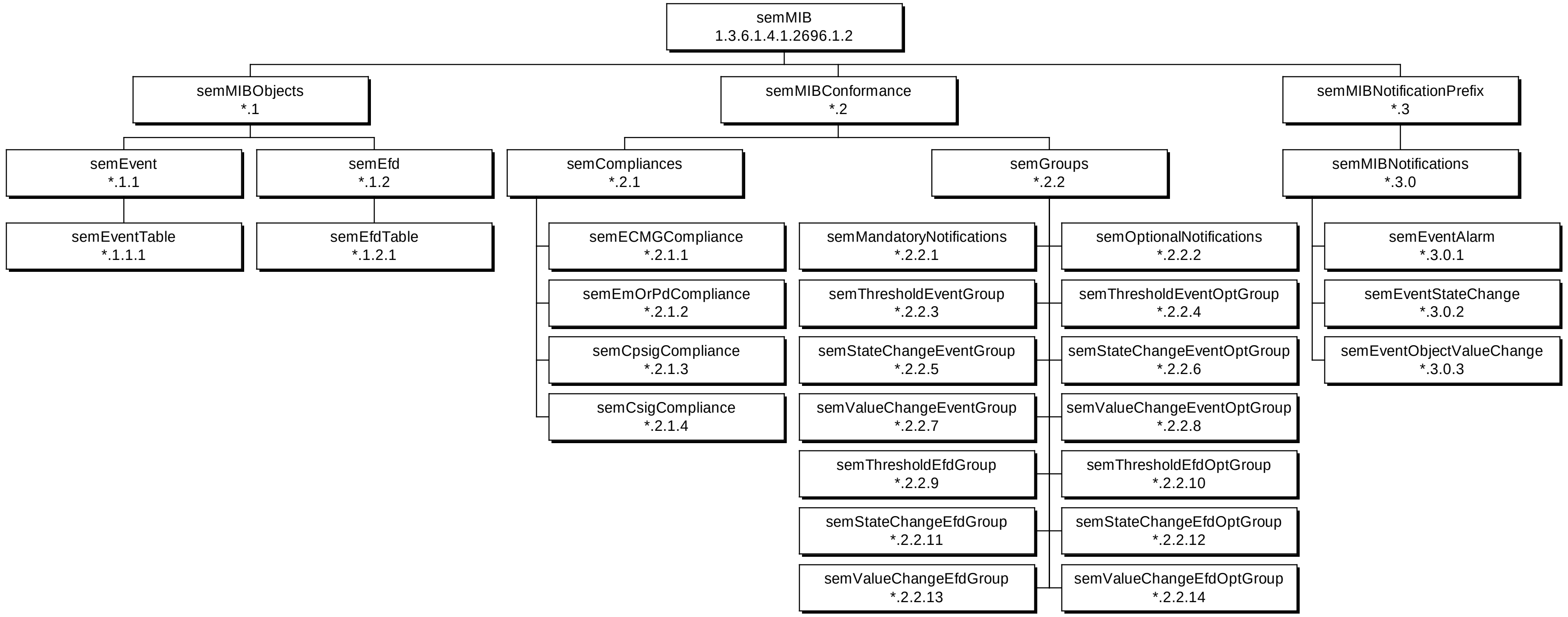


Figure 6: Simulcrypt Events Module SEM

7.2.4.1 Event Group

This group consists of event configuration information defining the types of events that the agent shall generate.

Table 7: CIM - SEM (P)SIG Group - Event Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
semEventName	bslbf/ the unique name of the target event, EntryName (SNMP)	provides a unique identification of an event	read-create
semEventAdminState	enumerated/administrative state of a table row, enumerated type (ITU-T Recommendation X.731 [8])	enables concurrency control between multiple management entities	read-create
semEventAlarmStatus	enumerated/alarm status of an event, enumerated type (ITU-T Recommendation X.731 [8])	enables event monitoring and clearing	read-create
semEventType	enumerated/indicates the type of event, enumerated type (ITU-T Recommendation X.734 [10])	enables differentiation between distinct event types	read-create
semEventText	bslbf/a description of an event's function and use, ASCII string of maximum 256 characters	enables textual description of an event for human readers	read-create
semEventChangedObjectId	bslbf/the object identifier of the MIB object to check and see if the event should fire, OBJECT IDENTIFIER (SNMP)	enables association of MIB objects with events	read-create
semEventToStateChange	4 uimsbf/if semEventChangedObjectId is a state/status variable this variable identifies the state that causes the event to be generated	enables association of events with state/status variables	read-create
semEventRisingThreshold	4 uimsbf/if semEventChangedObjectId is a threshold variable this variable indicates the threshold value to check against; if the value of semEventChangedObjectId is greater than or equal an event is generated; 32-bit unsigned integer	enables association of events with threshold variables	read-create
semEventFallingThreshold	4 uimsbf/if semEventChangedObjectId is a threshold variable this variable indicates the threshold value to check against; if the value of semEventChangedObjectId is less than or equal an event is generated	enables association of events with threshold variables	read-create
semEventProbableCause	enumerated/defines further probable cause for the last event of this type, enumerated type (ITU-T Recommendation X.734 [10] 10])	enables differentiation between event causes	read
semEventPerceivedSeverity	enumerated/defines the perceived severity of the last event of this type, enumerated type (ITU-T Recommendation X.734 [10])	enables differentiation of event severity level	read

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
SemEventTrendIndication	enumerated/indicates the trend of the last event of this type, enumerated type (ITU-T Recommendation X.734 [10])	enables the indication of the event trend, i.e. more/less severe	read
semEventBackedUpStatus	enumerated/indicates the backed up status of the object causing the event, enumerated type (ITU-T Recommendation X.731 [8])	enables identification of backed up objects	read-create
semEventBackedUpObject	bslbf/if the backed up status is backedUp this variable contains the object identifier of the back up object, OBJECT IDENTIFIER (SNMP)	enables specification of back up objects	read-create
semEventSpecificProblems	bslbf/identifies the object responsible for the problem, OBJECT IDENTIFIER (SNMP)	enables specification of specific problems	read-create
semEventFrequency	4 uimsbf/identifies the number of seconds to wait between event frequency checks, 32-bit unsigned integer	enables event throtelling	read-create
semEventSensitivity	enumerated/identifies whether the event is level or edge sensitive, Enumerated	enables two different types of event generation mechanisms: whenever a threshold is crossed and periodically as long as a threshold has been crossed	read-create
semEventStatus	enumerated/status variable for synchronizing row creation/deletion between management entities, RowStatus (SNMP)	enables synchronization of row creation/deletion	read-create

7.2.4.2 Event Forwarding Discriminator (EFD) Group

This group consists of EFD configuration information defining what types of events an EFD will transform into notifications, at what times of day it will do so, and to which managers it will send the notifications to. An EFD generates a notification if it is unlocked (administrative state), enabled (operational state) and available (availability status) and if all of the specified discriminators are true, i.e. if semEfdDiscriminatedTypes is specified then the type indicated has to match the type of the event for a notification to be generated. If multiple discriminators are specified by a single EFD then all have to be matched (i.e. logical AND) before a notification is generated. A not specified discriminator in an EFD is always TRUE. A single event can match multiple EFDs and generate multiple notifications if so specified by the semEfdOr variable.

The table is indexed by the event name (semEfdName) and the target address of the management entity, which is to receive the notification (semEfdTarget).

Table 8: CIM - SEM (P)SIG Group - EFD Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
semEfdName	bslbf/the unique name of the EFD, EntryName (SNMP)	provides a unique identification of an EFD	read-create
semEfdAdminState	enumerated/administrative state of a table row, enumerated type (ITU-T Recommendation X.731 [8])	enables concurrency control between multiple management entities	read-create
semEfdOperState	enumerated/operational state of an EFD, enumerated type (ITU-T Recommendation X.731 [8])	enables the indication of the current operation state	read-create
semEfdAvailStatus	enumerated/refelects the scheduling of the EFD, enumerated type (ITU-T Recommendation X.731 [8])	enables scheduling	read
semEfdStartTime	bslbf/defines the date and time at which an unlocked and enabled EFD starts functioning, i.e. changes the availability status from offDuty to available, DateAndTime (SNMP)	enables the scheduling of EFDs	read-create
semEfdStopTime	bslbf/defines the date and time at which an unlocked and enabled EFD stops functioning, i.e. changes its availability status from available to offDuty, DateAndTime (SNMP)	same	read-create
semEfdDailyStartTime	bslbf/defines the daily start time at which an unlocked and enabled EFD starts functioning, i.e. changes its availability status from offDuty to available, TimeTicks (SNMP)	enables daily scheduling of EFDs	read-create
semEfdDailyStopTime	bslbf/defines the daily stop time at which an unlocked and enabled EFD stops functioning, i.e. changes its availability status from available to offDuty, TimeTicks (SNMP)	same	read-create
semEfdWeeklyMask	1 uimsbf/defines the weekly schedule at which an unlocked and enabled EFD starts functioning, an octet string of 1 octet	enables weekly scheduling	read-create
semEfdTypes	enumerated/the event type that this EFD may generate notifications for, enumerated type (ITU-T Recommendation X.734 [10])	enables an EFD to be specialized for a particular event type	read-create
semEfdCause	enumerated/the probable cause that this EFD may generate notifications for, enumerated type (ITU-T Recommendation X.734 [10]	enables an EFD to be specialized by probable cause	read-create
semEfdSeverity	enumerated/the perceived severity that this EFD may generate notifications for, enumerated type (ITU-T Recommendation X.734 [10])	enables an EFD to be specialized by severity	read-create

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
semEfdSpecificProblems	bslbf/the identifier of the object that may cause the generation of a notification by this EFD, OBJECT IDENTIFIER (SNMP)	enables an EFD to be specialized by event causing Object	read-create
semEfdTrendIndication	enumerated/identifies the event trend that will cause a notification to be generated, enumerated type (ITU-T Recommendation X.734 [10])	enables an EFD to be specialized by event trend	read-create
semEfdChangedObjectId	bslbf/identifies the object whose value change shall cause the generation of a notification, OBJECT IDENTIFIER (SNMP)	enables an EFD to be specialized by value change of an object	read-create
semEfdToStateChange	4 uimsbf/the to state of the object that may cause the generation of a notification by this EFD	enables an EFD to be specialized by a state value	read-create
semEfdNotification	bslbf/identifies the notification object identifier to be generated if conditions are met, OBJECT IDENTIFIER (SNMP)	enables the association of a notification type with an EFD	read-create
semEfdOr	enumerated/identifies whether the EFD table shall be searched further for other possible matches and further possible notification generation, enumerated	enables multiple notifications to be generated by an event	read-create
semEfdTarget	bslbf/the IP address of the management entity to receive the notification if generated, IpAddress (SNMP)	enables the specification of the target management entity for the notifications generated by the EFD	read-create
semEfdText	bslbf/a description of an event's function and use, ASCII string of maximum 256 characters	enables textual description of an EFD for human readers	read-create
semEfdStatus	enumerated/status variable for synchronizing row creation/deletion between management entities, RowStatus (SNMP)	enables syncronization of row creation/deletion	read-create

7.2.4.3 Event Notification Group

This group defines three types of notifications which an agent can send to a manager. These are an alarm, a state change notification, and a value change notification. Each EFD specifies what type of notification is to be sent for an event that has occurred in the agent. The EFD also specifies the conditions under which such a notification is to be sent and the IP address of the manager to which the notification is to be sent. All standard SNMP traps are sent to the managers UDP port 162. Most management platforms support this mechanism.

The first notification type that can be generated is a semEventAlarm, which carries in addition to the standard SNMPv2 notification parameters the following objects from the events table:

- semEventName;
- semEventType;
- semEventProbableCause;
- semEventSpecificProblems;
- semEventPerceivedSeverity;
- semEventTrendIndication;
- semEventText.

The second notification type that can be generated is a semEventStateChange, which carries in addition to the standard SNMPv2 notification parameters the following objects from the events table:

- semEventName;
- semEventStateChange;
- semEventChangedObjectId.

The third notification type that can be generated is a semEventObjectValueChange, which carries in addition to the standard SNMPv2 notification parameters the following objects from the events table:

- semEventName;
- semEventChangedObjectId.

7.2.4.4 Conformance Requirements

The following table summarizes the conformance requirements for management entities implementing the Simulcrypt Events Module (SEM).

Table 9: CIM - SEM Conformance Requirements

Common Information Model- Simulcrypt Events Module Group	Management Entity Hosting or Representing an ECMG		Management Entity Hosting or Representing an EMMG		Management Entity Hosting or Representing a PDG		Management Entity Hosting or Representing a CPSIG		Management Entity Hosting or Representing a CSIG	
	mandatory	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**
Events Group - threshold events	name, type, text, object-id, thresholds, cause, severity, trend, frequency, status, admin state	backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, thresholds, cause, severity, trend, frequency, status, admin state	backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, thresholds, cause, severity, trend, frequency, status, admin state	backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, thresholds, cause, severity, trend, frequency, status, admin state	backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, thresholds, cause, severity, trend, frequency, status, admin state	backed up, back up-id, specific problems, sensitivity, alarm status
Events Group - state change events	name, type, text, object-id, to state, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, to state, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, to state, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, to state, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, to state, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status
Events Group - value change events	name, type, text, object-id, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status	name, type, text, object-id, frequency, status, admin state	cause, severity, trend, backed up, back up-id, specific problems, sensitivity, alarm status
EFD Group - threshold events	name, admin state, oper state, avail status, types, cause, severity, trend, object-id, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, cause, severity, trend, object-id, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, cause, severity, trend, object-id, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, cause, severity, trend, object-id, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, cause, severity, trend, object-id, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type
EFD Group - state change events	name, admin state, oper state, avail status, types, object-id, to state, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, to state, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, to state, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, to state, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, to state, notification, target, status	start, stop, dstart, dstop, week, specific problems, text, notification type

Common Information Model- Simulcrypt Events Module Group	**Management Entity Hosting or Representing an ECMG**		**Management Entity Hosting or Representing an EMMG**		**Management Entity Hosting or Representing a PDG**		**Management Entity Hosting or Representing a CPSIG**		**Management Entity Hosting or Representing a CSIG**	
	mandatory	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**
Events Group - value change events	name, admin state, oper state, avail status, types, object-id, notification, target, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, notification, target, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, notification, target, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, notification, target, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems, text, notification type	name, admin state, oper state, avail status, types, object-id, notification, target, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems, text, notification type
Notifications Group	alarm	state change, value change	alarm	state change, value change	alarm	state change, Value change	alarm	state change, Value change	alarm	state change, Value change

7.2.5 The Simulcrypt Logs Module (SLM)

SLM enables managers to configure the types of logs that can be generated by an agent. The mechanism and information model used are based on the ITU-T Recommendation X.735 [11] Log Model.

The log in addition to conceptually storing the logged information determines which information is to be logged. Each log contains a discriminator construct, which specifies the characteristics an event shall have in order to be selected for logging. SLM consists of the following object groups:

- Log Control Group: This group defines the types of logs tables the agent is maintaining, their discriminators, the log scheduling, etc.;
- The Logs Group: This group defines three logs: the alarm logs, the state change logs, and the object value change logs.

Logs are controlled by the Log Control Table as specified in the ITU-T Recommendation X.735 [11]. Each entry in that table associates events with logs and specifies when the event is to be logged in that log. The event is logged if the log discriminator holds. That is if the event is of a certain type, if it has been generated by a certain object, if it exceeds a certain threshold, etc. The log control entries themselves are controlled by state/status variables, the administrative state, the operational state, and the availability status. The manager can set the administrative and operational states. The availability status is set by the agent itself based on an automatic log control scheduling mechanism, which specifies the times during which the logs are to be made.

Log Control Table entries also specify log control information and log statistics.

The three logs defined are defined as tables in which each event is stored as a row. The logs in the alarm log table are logs of alarm events that have passed the log control discriminator in the Log Control Table. Similarly the logs in the state change log table are logs of state changes. And logs in the object value change table are logs of object value changes.

Figure 7 illustrates the Simulcrypt Logs Module MIB tree.

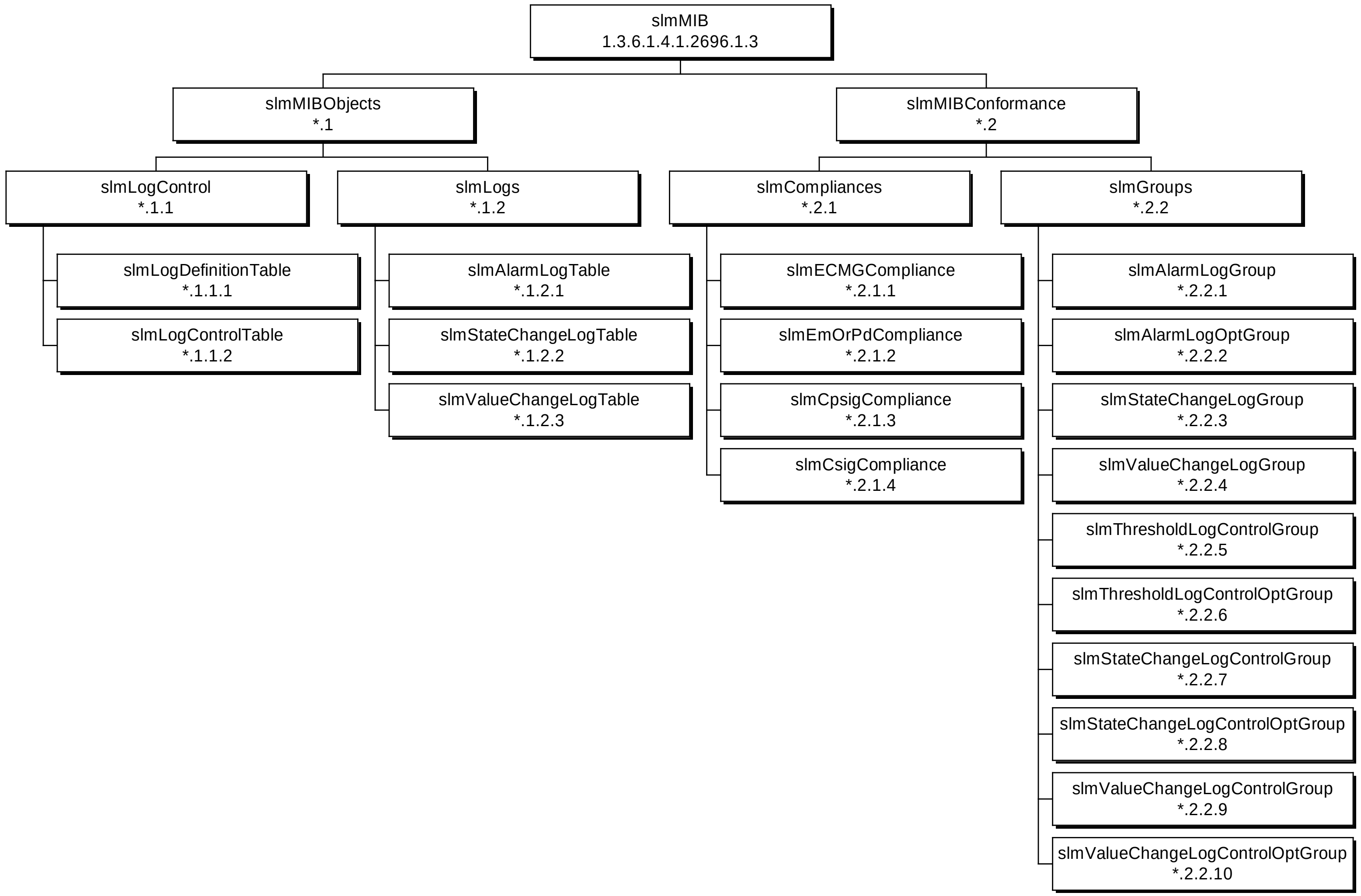


Figure 7: Simulcrypt Logs Module SLM

7.2.5.1 Log Control Group

This group defines the types of logs tables the agent is maintaining, their discriminators, the log scheduling, etc. The group consists of two tables, the Log Definition Table and the Log Control Table.

The Log Definition Table is used to define all logs in the system. Each entry consists of an slmLogDefinitionId, which is the Log identifier, variables defining the log state, and a specification of the log full action. The table is indexed by the logDefinitionName.

Table 10: CIM - SLM Log Definition Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
slmLogDefinitionName	bslbf/the unique name of the Log Definition Entry, EntryName (SNMP)	provides a unique identification of a Log Definition Entry	read-create
slmLogDefinitionId	bslbf/the unique log table identifier, OBJECT IDENTIFIER (SNMP)	provides a unique identification of Log Tables	read-create
slmLogDefinitionAdminState	enumerated/administrative state of a table row, enumerated (ITU-T Recommendation X.731 [8])	enables concurrency control between multiple management entities	read-create
slmLogDefinitionOperState	enumerated/operational state of an EFD, enumerated (ITU-T Recommendation X.731 [8])	enables the indication of the current operation state	read-create
slmLogDefinitionAvailStatus	enumerated/reflects the scheduling of the Log Control entry, enumerated (ITU-T Recommendation X.731 [8])	enables scheduling	read
slmLogDefinitionFullAction	enumerated/defines what action to take when the maximum log table size has been reached, enumerated (ITU-T Recommendation X.735 [8])	enables control of log full action	read-create
slmLogDefinitionMaxLogSize	4 uimsbf/defines the maximum size of a log table in number of octets	enables control of the maximum log table size	read-create
slmLogDefinitionCurrentLogSize	4 uimsbf/defines the current log table size	enables monitoring of logtable size	read
slmLogDefinitionNumberOfRecords	4 uimsbf/specifies the number of log records in the log table	enables monitoring of the log table size	read

The Log Control Table defines possibly multiple Log Controls for the different Logs. Each Log is identified by the Log Definition name. Each Log Control is identified by a Log Control name. The table is indexed by the LogDefinitionName and the LogControlName. Each log control consists of scheduling and log filtering information.

Table 11: CIM - SLM Log Control Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
slmLogDefinitionName	bslbf/the unique name of the Log Definition Entry, EntryName (SNMP)	provides a unique identification of a Log Definition Entry	read-create
slmLogControlName	bslbf/the unique name of the Log Control Entry, EntryName (SNMP)	provides a unique identification of a Log Control Entry	read-create
slmLogControlStartTime	bslbf/defines the date and time at which an unlocked and enabled Log Control entry starts functioning, i.e. changes the availability status from offDuty to available, DateAndTime (SNMP)	enables the scheduling of Log Controls	read-create
slmLogControlStopTime	bslbf/defines the date and time at which an unlocked and enabled Log Control entry stops functioning, i.e. changes its availability status from available to offDuty, DateAndTime (SNMP)	same	read-create
slmLogControlDailyStartTime	bslbf/defines the daily start time at which an unlocked and enabled Log Control entry starts functioning, i.e. changes its availability status from offDuty to available, TimeTicks (SNMP)	enables daily scheduling of log control entries	read-create
slmLogControlDailyStopTime	bslbf/defines the daily stop time at which an unlocked and enabled Log Control entry stops functioning, i.e. changes its availability status from available to offDuty, TimeTicks (SNMP)	same	read-create
slmLogControlWeeklyMask	bslbf/defines the weekly schedule at which an unlocked and enabled Log Control entry starts functioning, octet string of length 1	enables weekly scheduling	read-create
slmLogControlTypes	enumerated/the event type that this Log Control entry may generate logs for, enumerated (ITU-T Recommendation X.734 [10])	enables a Log Control entry to be specialized for a particular event type	read-create
slmLogControlCause	enumerated/the probable cause that this Log Control entry may generate logs for, enumerated (ITU-T Recommendation X.734 [10])	enables an Log Control entry to be specialized by probable cause	read-create
slmLogControlSeverity	enumerated/the perceived severity that this Log Control entry may generate logs for, enumerated (ITU-T Recommendation X.734 [10])	enables an Log Control entry to be specialized by severity	read-create
slmLogControlSpecificProblems	bslbf/the identifier of the object that may cause the generation of a log entry by this Log Control entry, OBJECT IDENTIFIER (SNMP)	enables an Log Control entry to be specialized by Object	read-create
slmLogControlToStateChange	bslbf/the to state of the object that may cause the generation of a log entry by this Log Control entry, 32–bit unsigned integer	enables an Log Control entry to be specialized by a state value	read-create
slmLogControlTrendIndication	enumerated/identifies the trend that will cause a log entry to be made, enumerated (ITU-T Recommendation X.734 [10])	enables specialization of log control based on event trends	read-create

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
slmLogControlChangedObjectId	bslbf/identifies the object that changed value and should be logged, OBJECT IDENTIFIER (SNMP)	enables specialization of log control based on objects causing the event	read-create
slmLogControlStatus	enumerated/status variable for synchronizing row creation/deletion between management entities, RowStatus (SNMP)	enables synchronization of row creation/deletion	read-create

7.2.5.2 Logs Group

The three logs defined are defined as tables in which each event is stored as a row. The logs in the alarm log table are logs of alarm events that have passed the log control discriminator in the Log Control Table. Similarly the logs in the state change log table are logs of state changes. And logs in the object value change table are logs of object value changes. The first table defines logs of alarm events as follows.

Table 12: CIM - SLM Alarm Log Table

Object	Size/Description	Object Justification	Head-end/ CA Maximum Access Right
slmAlarmLogName	bslbf/the unique name of the Log Control Entry, EntryName (SNMP)	provides a unique identification of the alarm log entry; it is identical to the event name	read
slmAlarmLogTime	4 uimsbf/the time at which the alarm has been logged, TimeTicks (SNMP)	provides a unique identification of Log Tables	read
slmAlarmLogText	bslbf/a textual description of the event being logged, ASCII string of maximum 256 characters	records the event description	read
slmAlarmLogType	enumerated/the event type of this log entry, enumerated (ITU-T Recommendation X.734 [10])	records alarm type	read
slmAlarmLogCause	enumerated/the event cause of this log entry, enumerated (ITU-T Recommendation X.734 [10])	records event cause	read
slmAlarmLogSeverity	enumerated/the alarm severity of the logged event, enumerated (ITU-T Recommendation X.734 [10])	receords event severity	read
slmAlarmLogSpecificProblems	bslbf/the identifier of the object that caused the logged event, OBJECT IDENTIFIER (SNMP)	records the id of objects causing the event	read
slmAlarmLogTrendIndication	enumerated/the trend of the event that has been logged, enumerated (ITU-T Recommendation X.734 [10])	records event trend	read
slmAlarmLogChangedObjectId	bslbf/identifies the object that changed value and caused the logged event, OBJECT IDENTIFIER (SNMP)	records the id of the object causing the event	read

The second table defines logs of state change events as follows.

Table 13: CIM - SLM State Change Log Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
slmStateChangeLogName	bslbf/the unique name of the Log Control Entry, EntryName (SNMP)	provides a unique identification of the log entry; it is identical to the event name	read
slmStateChangeLogTime	4 uimsbf/the time at which the alarm has been logged, TimeTicks (SNMP)	provides a unique identification of Log Tables	read
slmStateChangeLogText	bslbf/a textual description of the event being logged, ASCII string of maximum 256 characters	records the event description	read
slmStateChangeLogToStateChange	bslbf/the to state change of the event being logged, enumerated (ITU-T Recommendation X.734 [10])	records event to state change	read
slmStateChangeLogChangedObjectId	bslbf/identifies the object that changed value and caused the logged event, OBJECT IDENTIFIER (SNMP)	records the id of the object causing the event	read

The third table defines logs of value change events as follows.

Table 14: CIM - SLM Value Change Log Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
slmValueChangeLogName	bslbf/the unique name of the Log Control Entry, EntryName (SNMP)	provides a unique identification of the log entry; it is identical to the event name	read
slmValueChangeLogTime	4 uimsbf/the time at which the alarm has been logged, TimeTicks (SNMP)	provides a unique identification of Log Tables	read
slmValueChangeLogText	bslbf/a textual description of the event being logged, ASCII string of maximum 256 characters	records the event description	read
slmValueChangeLogChangedObjectId	bslbf/identifies the object that changed value and caused the logged event, OBJECT IDENTIFIER (SNMP)	records the id of the object causing the event	read

7.2.5.3 Conformance Requirements

The Simulcrypt Logs Module (SLM) is optional. However if log management is implemented the following conformance requirements hold.

Table 15: CIM - SLM Conformance Requirements

Common Information Model - CIM Simulcrypt Logs Module Group	Management Entity Hosting or Representing an ECMG		Management Entity Hosting or Representing an EMMG/		Management Entity Hosting or Representing a PDG		Management Entity Hosting or Representing a CPSIG		Management Entity Hosting or Representing a CSIG	
	mandatory	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**	**mandatory**	**optional**
Log Control Group - threshold events	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, cause, severity, trend, object-id, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, cause, severity, trend, object-id, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, cause, severity, trend, object-id, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, cause, severity, trend, object-id, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, cause, severity, trend, object-id, status	start, stop, dstart, dstop, week, specific problems
Log Control Group - state change events	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, to state, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, to state, status	start, stop, dstart, dstop, week, specific problems	admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, to state, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, to state, status	start, stop, dstart, dstop, week, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, to state, status	start, stop, dstart, dstop, week, specific problems
Log Control Group - value change events	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, status	name, log-id, start, stop, dstart, dstop, week, cause, severity, trend, specific problems	admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems	name, log-id, admin state, oper state, avail status, full action, max log, number of recs, current log size, types, object-id, status	start, stop, dstart, dstop, week, cause, severity, trend, specific problems
Logs Group - alarm logs	name, time, text, type, object-id	severity, problems, trend	name, time, text, type, object-id	severity, problems, trend	name, time, text, type, object-id	severity, problems, trend	name, time, text, type, object-id	severity, problems, trend	name, time, text, type, object-id	severity, problems, trend
Logs Group - state change logs	name, time, text, object-id, to state		name, time, text, object-id, to state		name, time, text, object-id, to state		name, time, text, object-id, to state		name, time, text, object-id, to state	
Logs Group - state change logs	name, time, text, object-id		name, time, text, object-id		name, time, text, object-id		name, time, text, object-id		name, time, text, object-id	

7.3 CAS component monitoring and configuration

Monitoring and configuration of CAS components is accomplished through the Simulcrypt Identification Module (SIM). SIM contains version information of the management software as well as Simulcrypt component identification, configuration, and status information. Its primary purpose is to provide uniform Simulcrypt component configuration and status information across all Simulcrypt elements of a Conditional Access System (Note that this is not the only purpose of SIM as it also facilitates the transaction based C(P)SI/(P)SI interface).

If a management system is implemented, a Simulcrypt CA component needs to implement only those SIM objects, which are required for that component type by the SIMF. If a particular object is not supported by the management entity of the Simulcrypt CA component (i.e. the SNMP agent) the standard SNMP error code noSuchName is to be returned.

Several totals listed below, mostly concerning number of errors, refer to the total since the agent has started; other totals, mostly concerning number of streams, channels, etc…, refer to current values. All bslbf and uimsbf units referred to below are in byte units.

The module can be extended by proprietary objects and groups as needed for specific DVB Simulcrypt Integrated Management Systems. SIM consists of the following object groups. All access rights can be further restricted by individual MIB views if so desired in particular implementations.

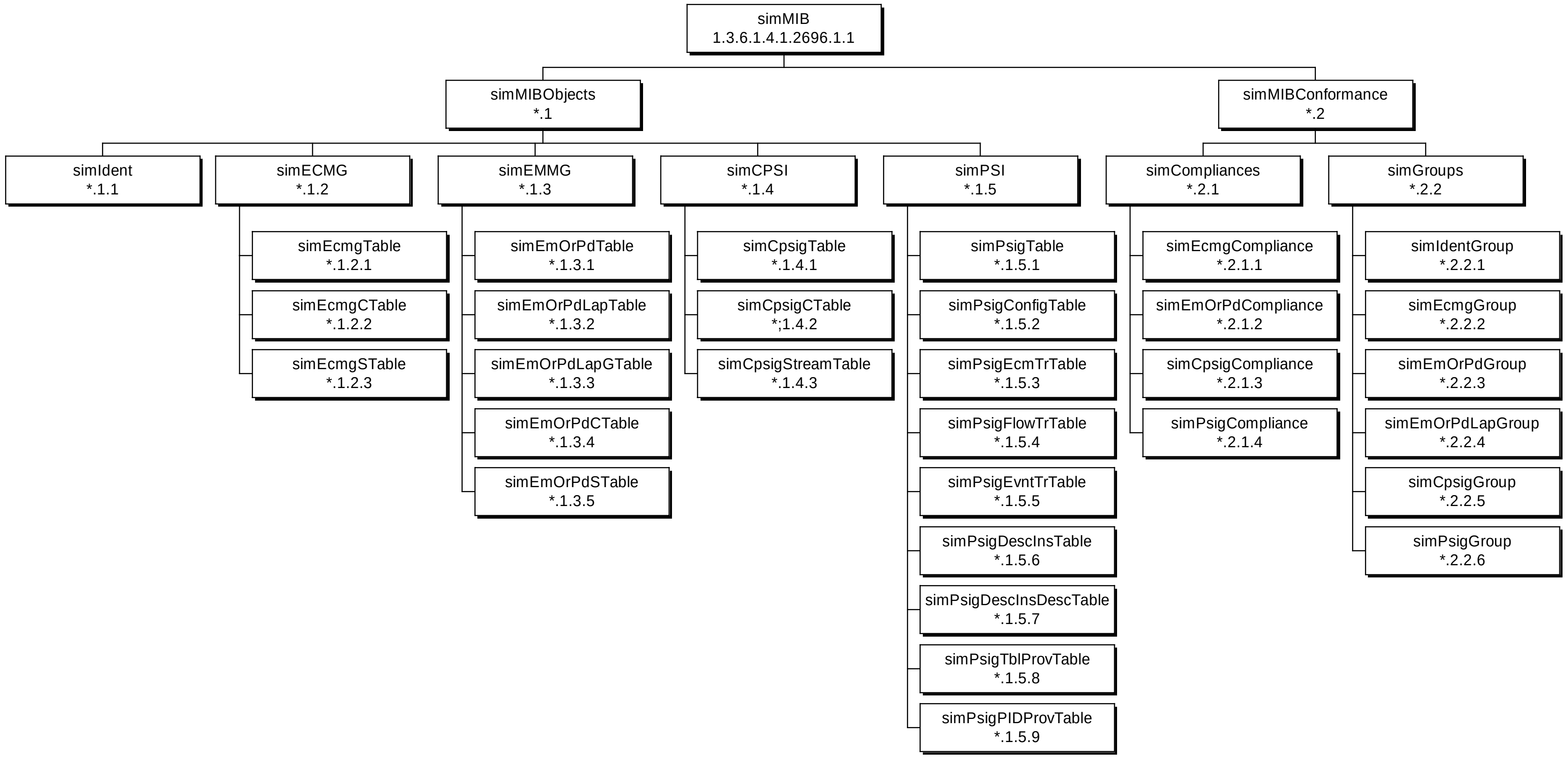


Figure 8: Simulcrypt Information Module SIM

7.3.1 Ident Group

This group is used for software configuration management of all Simulcrypt components and includes the following objects.

Table 16: CIM - Simulcrypt Identification Module

Object	Size/Description	Object Justification	Maximum Access Right
simSoftwareVersion	80 bslbf/ASCII string of software version	to facilitate software configuration management	read
simMibVersion	80 bslbf/ASCII string of MIB version	same	read
simMibPrivateVersion	80 bslbf/ASCII string of private MIB version	same	read
simAgentVersion	80 bslbf/ASCII string of agent version	same	read

7.3.2 ECM Generator Group

This group is used for configuration management and status monitoring of ECM Generators. It identifies each one of the ECM Generators by the IP Address and TCP Port Number. It associates Super_CAS_ids, ECM_channel_ids, and ECM_stream_ids with ECM Generators. It also associates status information and statistics with channels and streams. The ECM Generator Group consists of three conceptual tables.

The first table is the interconnection table and is used for the Head-end Network Manager to query the IP addresses and the port number to be used by an SCS to create a channel. It is indexed by a unique EcmgIndex which is an integer assigned by the ECMG agent.

Table 17: CIM - SIM ECMG Group - Interconnection Table

Object	Size/Description	Object Justification	Maximum Access Right
simEcmgIndex	2 uimsbf/ unique index of the table row	allows interconnection management	read
simEcmgIpAddress	octet string of 4 octets/ IP address of the ECMG	same	read
simEcmgTcpPort	2 uimsbf/TCP port number of the ECMG	same	read
simEcmgSuCasId	4 uimsbf/Super_Cas_id	same	read
simEcmgChannels	2 uimsbf/total number of channels this ECMG is currently maintaining	same	read
simEcmgCwPrs	4 uimsbf/total number of CW Provisioning requests received by this ECMG	statistic	read
simEcmgErrs	4 uimsbf/total number of communications errors for this ECMG	same	read
simEcmgTargetCpsig	4 uimsbf/index into the C(P)SIG table identifying the C(P)SIG associated with this ECMG	interconnection management	read
simEcmgCaMib	blsb/pointer to a provider proprietary Ecmg MIB (like ifSpecific in the interfaces group of MIB II)	enables extension of ECMG MIB by proprietary modules	read

The second ECMG table is used for monitoring channel information. It is indexed by the EcmgIndex and the ChannelId.

Table 18: CIM - SIM ECMG Group - Channel Table

Object	Size/Description	Object Justification	Maximum Access Right
simEcmgIndex	2 uimsbf/unique index of the table row	enables head-end or CAS network manager to monitor the status of individual channels and streams	read
simEcmgChannelId	2 uimsbf/identifier of a channel	same	read
simEcmgScsIpAddress	octet string of 4 octets/IP address of the SCS	same	read
simEcmgScsTcpPort	2 uimsbf/TCP port number of the SCS	same	read
simEcmgCStreams	2 uimsbf/total number of streams this ECMG is currently maintaining on this channel	same	read
simEcmgCCwPrs	4 uimsbf/total number of CW Provisioning requests this ECM has received on this channel	same	read
simEcmgCErrs	4 uimsbf/total number of error messages on this channel	same	read

The third table is used for monitoring stream information. It is indexed by the EcmgIndex, the ChannelId, and the StreamId.

Table 19: CIM - SIM ECMG Group - Stream Table

Object	Size/Description	Object Justification	Maximum Access Right
simEcmgIndex	2 uimsbf/unique index of the table row	enables head-end or CAS network manager to monitor the status of individual channels and streams	read
simEcmgChannelId	2 uimsbf/identifier of a channel	same	read
simEcmgStreamId	2 uimsbf/identifier of a stream	same	read
simEcmgEcmId	2 uimsbf/identifier of ECM stream	assigned by the head-end to uniquely identify each ECM stream	
simEcmgSLastCp	4 uimsbf/last crypto period processed for this stream	enables head-end or CAS network manager to monitor the status of individual channels and streams	read
simEcmgSCwPrs	4 uimsbf/total number of CW Provisioning requests this ECMG has received on this stream	same	read
simEcmgSErrs	4 uimsbf/total number of error messages for this ECMG on this Stream	same	read

7.3.3 EMMG/PDG Group

This group is used for management of EMM/PD Generators. It identifies each one of the EMM/PD Generators by the IP Address and TCP/UDP Port Number. It also associates client_ids, data_channel_ids, and data_stream_ids with EMM/PD Generators. It also associates status information and statistics with streams. The EMMG/PDG Generator Group consists of five conceptual tables.

The first table is used for information relevant to EMMG/PDG and is indexed by a unique EmOrPdIndex, which is assigned by the EMMG/PDG agent.

Table 20: CIM - SIM EMMG/PDG Group - EMMG/PDG Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simEmOrPdIndex	2 uimsbf/unique index of the table row	provides a unique identification of a row	read
simEmOrPdDataType	as defined in the DVB Simulcrypt Spec	indicates to network managers whether this is an EMMG or a PDG	read
simEmOrPdClientId	4 uimsbf/ClientID	identifies the client	read
simEmOrPdCommCapability	Enumerated/TCP or UDP or both	indicates the EMMG/PDG communications capability	read
simEmOrPdErrs	4 uimsbf/total number of communications errors for this EMMG/PDG	error statistic	read
simEmOrPdTargetCpsig	4 uimsbf/index into the C(P)SIG table identifying the C(P)SIG associated with this EMMG/PDG	interconnection management	read
simEmOrPdCaMib	bslbf/pointer to CA provider proprietary MIB	enables custom MIB extensions	read

The second table is used for configuration of EMMG/PDG. It is uniquely indexed by the EmOrPdLapIndex, which is a globally assigned quantity (with respect to the head-end) and associates globally assigned Logical Access Points (LAPs) with mux ports.

Table 21: CIM - SIM EMMG/PDG Group - Logical Access Point Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simEmOrPdLapIndex	2 uimsbf/global unique Logical Access Point (LAP) identifier	enables the association of a global logical identifier with a mux IP address and port	read-create
simEmOrPdAdminState	enumerated (ITU-T Recommendation X.734 [10])/the administrative state of a table row	used to control concurrent access to writeable objects as defined in ITU-T Recommendation X.734 [10]	read-create
simEmOrPdLapCommType	enumerated/TCP or UDP	defines communications type	read-create
simEmOrPdLapMuxIpAddress	octet string of 4 octets/IP address of the Mux	identifies the mux's IP address	read-create
simEmOrPdLapMuxPort	2 uimsbf/TCP/UDP port of the Mux	identifies the mux's port	read-create
simEmOrPdLapStatus	enumerated/row status of the entry, RowStatus (SNMP)	used to manage the creation/deletion of rows in a table	read-create

The third table is also used for configuration of EMMG/PDG. It associates LAP Groups and LAPs and is uniquely indexed by the EmOrPdLapGroup, and EmOrPdLapIndex. This association permits a manager to configure an EMMG/PDG to service multiple multiplexers and/or multiple multiplexer ports.

Table 22: CIM - SIM EMMG/PDG Group - Logical Access Point Group Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simEmOrPdLapGroup	2 uimsbf/not-unique identifier associating EMMGs to EMM populations	enables the establishment of multiple channels by the EMMG based on LapGroup	read-create
simEmOrPdLapIndex	2 uimsbf/global unique Logical Access Point (LAP) identifier	enables the association of a global logical identifier with a mux IP address and port	read-create
simEmOrPdLapGAdminState	enumerated (ITU-T Recommendation X.734 [10])/the administrative state of a table row	used to control concurrent access to writeable objects as defined in ITU-T Recommendation X.734 [10]	read-create
simEmOrPdLapGStatus	enumerated/row status of the entry, RowStatus (SNMP)	used to manage the creation/deletion of rows in a table	read-create

The fourth table is used for channel related monitoring information.The table is indexed by the EmOrPdIndex, EmOrPdLapIndex, and channel_id.

Table 23: CIM - SIM EMMG/PDG Group - Channel Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simEmOrPdIndex	2 uimsbf/unique index of the EMMG/PDG	identifies the EMMG/PDG	read
simEmOrPdLapIndex	2 uimsbf/logical global unique identifier	enables the association of a global logical identifier with a mux IP address and port	read
simEmOrPdChannelId	2 uimsbf/identifier of a channel	identifies channel and enables monitoring of it.	read
simEmOrPdCommType	enumerated/TCP or UDP	defines the communications type for the channel (TCP or UDP)	read-create
simEmOrPdCIpAddress	octet string of 4 octets/IP address	used with mux Ip address and port number to identify the TCP connection used for access of MIB II.	read
simEmOrPdCPort	2 uimsbf/TCP/UDP port number	same	read
simEmOrPdCErrs	2 uimsbf/total number of communications errors on this channel	statistic	read

The fifth table contains stream related information. It is indexed by the simEmOrPdIndex, simEmOrPDLapIndex, and simEmOrPdDataId.

Table 24: CIM - SIM EMMG/PDG Group - Stream Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simEmOrPdIndex	2 uimsbf/unique index of the EMMG/PDG	identifies the EMMG/PDG	read
simEmOrPdLapIndex	2 uimsbf/logical global unique identifier	enables the association of a global logical identifier with a mux IP address and port	read-create
simEmOrPdDataId	2 uimsbf/unique EMM or PD stream identifier	assigned by the head-end to uniquely identify the EMM or PD stream	read-create
simEmOrPdChannelId	2 uimsbf/identifier of a channel	enables head-end or CAS network manager to monitor the status	read
simEmOrPdBwidth	4 uimsbf/bandwidth	maximum bandwidth allocated for this EMM/Private Data stream	read-create
simEmOrPdStreamId	2 uimsbf/identifier of a stream	enables head-end or CAS network manager to monitor the status	read
simEmOrPdSErrs	4 uimsbf/total number of communications errors on this stream	statistic	read
simEmOrPdSBytes	4 uimsbf/total number of bytes sent by the EMMG/PDG on this stream	statistic	read

7.3.4 C(P)SIG Group

This group is used if the (P)SIG Group is not implemented and the stream/channel based protocol between the Custom (P)SIG and (P)SIG is implemented.

The C(P)SIG Group enables management of some aspects of the interaction between the Custom (P)SI Generators (C(P)SIG) and the (P)SI generator. It identifies each one of the C(P)SI Generators by the IP Address and TCP Port Number. It also associates Super_CAS_ids, Custom_channel_ids, Custom_stream_ids with C(P)SI Generators. It also associates status information and statistics with streams. The C(P)SI Generator Group consists of three conceptual tables.

The first table specifies C(P)SIG related information, which is posted by the entity hosting or representing the C(P)SIG. A CA/Head-end manager can read this table to communicate TCP/IP information to the (P)SIG, which will establish the TCP connection with the C(P)SIG. This table is indexed by a unique CpsigIndex, which is assigned by the CPSIG agent.

Table 25: CIM - SIM C(P)SIG Group - C(P)SIG Table

Object	Size/Description	Object Justification	Maximum Access Right
simCpsigIndex	2 uimsbf/unique index of the table row	provides a unique identification of a row	read
simCpsigSuperCasId	4 uimsbf/Super_CAS_id	client ID	read
simCpsigErrs	4 uimsbf/total number of communications errors for this C(P)SIG	statistic	read
simCpsigChannels	4 uimsbf/total number of channels currently maintained by this C(P)SIG	statistic	read
simCpsigCpsigIpAddress	octet string of 4 octets/IP address of the C(P)SIG	the IP address of the C(P)SIG	read
simCpsigCpsigPort	4 uimsbf/TCP port of the C(P)SIG	the port number of the C(P)SIG	read
simCpsigCaMib	bslbf/pointer to CA provider proprietary MIB, OBJECT IDENTIFIER (SNMP)	enables custom MIB extensions	read

The second table is used for channel related monitoring information.The table is indexed by the CpsigIndex and custom_channel_id.

Table 26: CIM - SIM C(P)SIG Group - Channel Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simCpsigIndex	2 uimsbf/unique index of the C(P)SIG	refers to the index in the first table and identifies the C(P)SIG	read
simCpsigChannelId	2 uimsbf/identifier of a channel	identifies channel and enables monitoring of it	read
simCpsigPsigIpAddress	octet string of 4 octets/IP address of the (P)SIG	used with the port number and C(P)SIG IP address and port number can identify the TCP connection used for access of MIB II.	read
simCpsigPsigPort	2 uimsbf/TCP port number of the (P)SIG	same	read
simCpsigCErrs	4 uimsbf/total number of communications errors on this channel	statistic	read
simCpsigCTstrms	4 uimsbf/total number of 'transport' streams on this channel	statistic	read
simCpsigCSstrms	4 uimsbf/total number of 'service' streams on this channel	statistic	read

The third table contains 'transport' stream related information. It is indexed by the CpsigIndex, custom_channel_id, and custom_stream_id.

Table 27: CIM - SIM C(P)SIG Group - Stream Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simCpsigIndex	2 uimsbf/unique index of the C(P)SIG	refers to the index in the first table and identifies the C(P)SIG	read
simCpsigChannelId	2 uimsbf/identifier of a channel	enables head-end or CAS network manager to monitor the status of streams on a channel	read
simCpsigStreamId	2 uimsbf/identifier of a stream	same	read
simCpsigStreamTStreamId	2 uimsbf/identifier of a stream	associates a transport stream with the stream	read
simCpsigStreamNid	2 uimsbf/network identifier	associates a network with this stream	read
simCpsigStreamOnid	2 uimsbf/original network identifier of this stream	associates an original network id with the stream	read
simCpsigStreamMaxCompTime	2 uimsbf/maximum time needed to process trigger by C(P)SIG	enables (P)SIG to estimate the time between its triggering and descriptor insertion by the C(P)SIG	read
simCpsigStreamTriggerEnable	2 uimsbf/trigger type definition	identifies which trigger types the C(P)SIG wants	read
simCpsigStreamLastTrigger	2 uimsbf/trigger type	identifies the type of the last trigger processed	read
simCpsigStreamLastEventId	2 uimsbf/event identifier	identifies the last event for which a trigger has been processed	read
simCpsigStreamLastServiceId	2 uimsbf/ the service identifier	identifies the service related to the last ECM related trigger	read
simCpsigStreamLastEsId	2 uimsbf/ the elementary stream identifier	identifies the elementary stream of the last trigger if the ECMs are to be applied component wide only and not service wide	read
simCsigStreamLastEcmPid	2 uimsbf/ the ECM PID	identifies the ECM PID of the last ECM related trigger	read
simCpsigStreamErrs	4 uimsbf/total number of communications errors on this stream	statistic	read
simCpsigStreamBytes	4 uimsbf/total number of bytes sent by the CPSIG on this stream	statistic	read

7.3.5 Conformance Requirements

Table 28 summarizes the conformance requirements for management entities implementing the Simulcrypt Identification Module (SIM), by group.

Table 28: CIM - SIM Conformance Requirements

Common Information Model - CIM Simulcrypt Identification Module Group	Management Entity Hosting or Representing an ECMG		Management Entity Hosting or Representing an EMMG		Management Entity Hosting or Representing a PDG		Management Entity Hosting or Representing a (P)SIG		Management Entity Hosting or Representing a C(P)SIG	
	mndt	**optnl**	**mndt**	**optnl**	**mndt**	**optnl**	**mndt**	**optnl**	**mndt**	**optnl**
Ident Group	X		X		X		X		X	
ECMG Group	X									
EMMG/PDG Group (without LAPG table)			X		X					
EMMG/PDG Group (LAPG Table)				X		X				
PSIG Group							X			
CPSI Group									X	

8 C(P)SIG ↔ (P)SIG interface

8.1 Overview and Scope

This section defines the interface that allows one or more CA Systems (CASs) to insert private data into standard MPEG-2 PSI and DVB SI tables that are generated by the head-end system.

The following subsections detail an application protocol model for the C(P)SIG ⇔ (P)SIG interface and two implementation-oriented protocols (connection-oriented protocol and SIMF-based protocol) based on this model.

Subclause 8.2 presents the model of application protocol defined to support the C(P)SIG ⇔ (P)SIG interface.

Subclause 8.3 specifies the C(P)SIG ⇔ (P)SIG protocol defined on the basis of command/response messages; it includes the state machines definition and the messages description.

Subclause 8.4 specifies the C(P)SIG ⇔ (P)SIG protocol defined in the network management environment; it includes the description of dedicated data structure (MIB) and the definition of rules of operation.

The head-end and each CAS shall comply with the requirements presented in annex E, clauses E.1 and E.2 respectively, as well as all specifications documented in the above-described sections.

As defined in the clause 4 and shown in figure 9, the logical components involved are as follows:

- **PSI Generator (PSIG):** the head-end process(es) responsible for generating MPEG-2 PSI (Program Specific Information) tables;

- **SI Generator (SIG):** the head-end process(es) responsible for generating DVB SI (Service Information) tables;

NOTE: The NIT (Network Information Table) is considered a DVB SI table.

- **Custom PSI Generator (CPSIG):** the CA System (CAS) process(es) responsible for generating CAS-specific private data for insertion in selected MPEG-2 PSI tables;

- **Custom SI Generator (CSIG):** the CAS process(es) responsible for generating CAS-specific private data for insertion in selected DVB SI tables;

- the generic term **(P)SIG** refers to a head-end process that serves as a PSIG, an SIG, or both (PSISIG). If the head-end supports the C(P)SIG ⇔ (P)SIG interface, it shall include one or more (P)SIG;

- the generic term **C(P)SIG** refers to a head-end process that serves as a CPSIG, a CSIG, or both (CPSISIG). Each CAS may (optionally) include one or more C(P)SIG.

This section defines the interface between the C(P)SIG and the (P)SIG.

Table 29: Values of (P)SIG_type parameter

(P)SIG_type values	C(P)SIG or (P)SIG processes
0x01 0x02 0x03	PSIG SIG PSISIG
0x04 0x08 0x0C	CPSIG CSIG CPSISIG
other values	DVB reserved

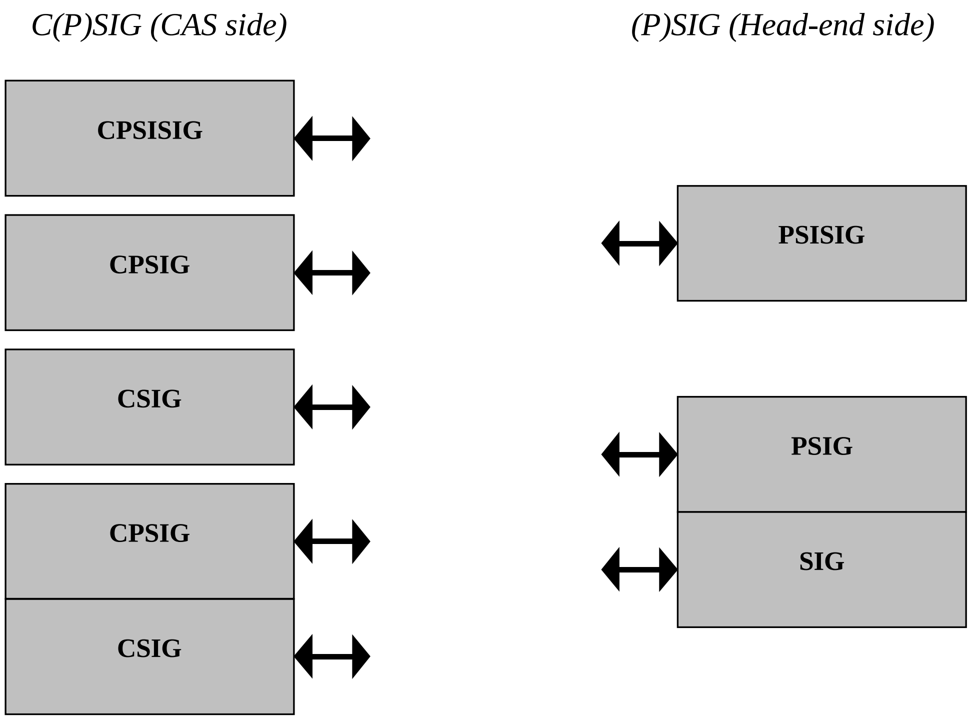


Figure 9: Possible logical architectures of C(P)SIG and (P)SIG

8.1.1 Note on commercial agreements

Certain aspects of the C(P)SIG ⇔ (P)SIG interface fall outside the scope of the present document, and shall be decided by commercial agreements. One such example is the set of non-mandatory PSI and SI tables generated by a (P)SIG. All such aspects of this interface are indicated in the text.

8.1.2 Note on the PDG ↔ MUX Interface

A CAS may also create private data in the form of private sections that comply with MPEG-2 and DVB section syntax. The PDG ⇔ MUX interface in clause 6 should be used to insert such private sections.

8.2 Application protocol model

8.2.1 Overview of the C(P)SIG ↔ (P)SIG Application Protocol

The C(P)SIG⇔(P)SIG Application Protocol includes and defines the following five transaction types:

- Trigger Transactions: Triggers are asynchronous events from the (P)SIG to the C(P)SIG. The (P)SIG first informs the C(P)SIG about the types of triggers it supports. The C(P)SIG then informs the (P)SIG which types of triggers it wishes to receive. The (P)SIG then communicates these triggers to the C(P)SIG as the corresponding events occur;

- (P)SI Table Provisioning Transactions: The C(P)SIG requests (P)SI tables from the (P)SIG and the (P)SIG communicates those to the C(P)SIG;
- C(P)SIG Descriptor Insert Transactions: The C(P)SIG requests that the (P)SIG inserts (P)SI descriptors into the transport stream. The (P)SIG informs the C(P)SIG about the success of the insertion;
- Service Change Transactions: The (P)SIG informs the C(P)SIG about a modification in service existence in a transport stream (addition of a service to a TS or deletion of a service from a TS);
- PID Provisioning Transactions: The C(P)SIG requests from the (P)SIG the PID value assigned by the head-end to a stream (ECM, EMM or private data) identified by a unique identifier which includes the type of the stream, the identifier of the CA system and the CA sub-system the stream belongs to, and an unique stream number.

In the following subsections configurations, topologies and the five different transaction types are defined. The transaction type specifications include only the definitions of the data exchanged by the components. The actual data exchange mechanisms are transport dependent and are fully defined in the following two clauses.

8.2.2 Configurations and Topologies

Figure 10 illustrates the relevant components in a head-end system with a single CA System (CAS) and in a head-end system with multiple CASs. In both diagrams, the *M-to-N* and *1-to-N* notation reflects logical associations, not process interconnections.

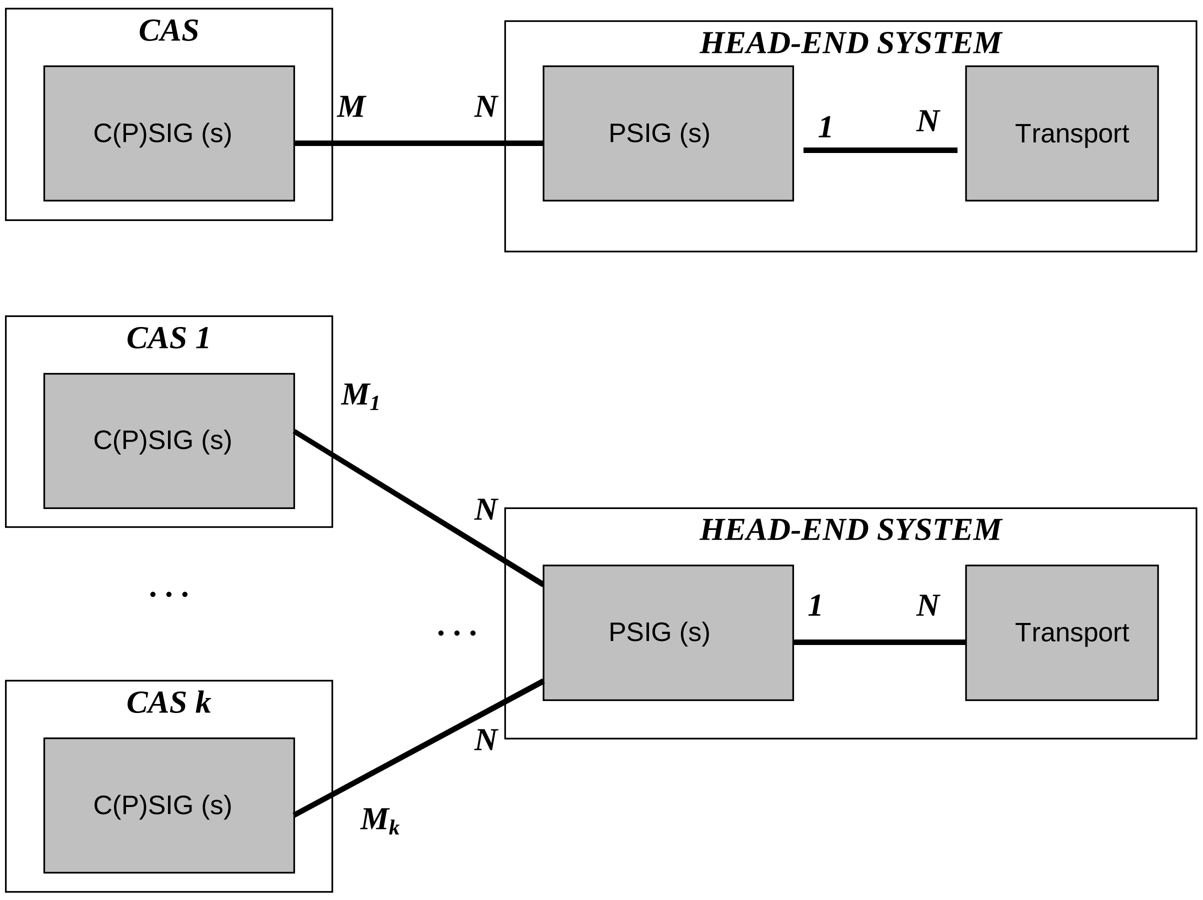


Figure 10: Head-end system with single CAS (top) and with multiple CASs (bottom)

The head-end includes the following components:

- one or more transport streams (TSs) that conform to MPEG-2 and DVB specifications. The C(P)SIG ⇔ (P)SIG Interface views each TS, and its data streams, as logical entities. As such, each TS is uniquely identified, per DVB SI, by the triple {original_network_id, network_id, transport_stream_id}. The C(P)SIG ⇔ (P)SIG interface does not deal with physical components (namely MUXes) that generate TSs;

- one or more (P)SIG components. Each (P)SIG is uniquely identified by its IP address and TCP port number;
- connections between the (P)SIGs and the TSs. The relationship between (P)SIG and TS is *1-to-N*: each (P)SIG generates standard PSI and/or SI tables for one or more TSs and no TS is served by more than one (P)SIG.

The head-end shall supply either a PSISIG or a {PSIG+SIG} to serve each TS.

The present document does not define the physical connection between (P)SIG and TS; it is a matter of head-end implementation.

Each CAS that supports the C(P)SIG ⇔ (P)SIG Interface includes the following components:

- one or more C(P)SIG components. Each C(P)SIG is uniquely identified by a custom_CAS_id, as well as its IP address and TCP port number.

The relationship between the C(P)SIG(s) of a CAS, and the (P)SIG(s) of the head-end, is M-to-N. Given also the 1-to-N relationship of (P)SIG to TS, this means:

- each C(P)SIG may generate CAS-specific private data for zero or more TSs;
- any TS may be served by zero or more C(P)SIGs, as long as said C(P)SIGs and TS are connected to the same (P)SIG.

The components of each CAS are logically independent of each other. Any TS may be served by zero or more C(P)SIGs from one or more CASs, as long as said C(P)SIGs and TS are connected to the same (P)SIG.

8.2.3 Trigger Transaction Type

The trigger is the means by which a (P)SIG component indicates to a C(P)SIG that a particular event is going to take place.

DVB Simulcrypt has identified the following trigger types:

- Following DVB-event;
- Future DVB-event;
- ECM stream setup;
- Access criteria change;
- ECM stream closure;
- Flow PID change.

The **Following DVB-event** trigger indicates that a new event is going to be broadcast on a given service.

The **Future DVB-event** trigger indicates that the head-end has at its disposal some information concerning a future event. The moment at which this message is sent is not specified by DVB.

The **ECM stream setup** trigger indicates that a new ECM stream is going to be opened.

The **Access criteria change** trigger indicates that the access criteria of an existing ECM stream is going to be changed.

The **ECM stream closure** trigger indicates that an ECM stream is due for closure.

The **Flow PID change** trigger indicates that the PID value of an existing ECM, EMM or private data stream is going to be changed.

With the exception of the **Future DVB-event** trigger, the time at which the trigger message is sent with respect to the corresponding event is computed by the (P)SIG, by subtracting the **max_comp_time** value from the time at which the corresponding event takes place. This lets time for the C(P)SIG to reply with descriptor_insert messages if needed, and control the precise timing of the insertion of descriptors.

For the head-end, the support of the **Future DVB-Event** is optional.

During the channel setup, the (P)SIG indicates what kind of triggers it is able to generate. The C(P)SIG has the possibility to subscribe to some trigger types per service. A trigger subscription is on a service basis, meaning, that a C(P)SIG can subscribe to different triggers depending on the services: such a trigger subscription can be modified at any time.

The (P)SIG communicates to the C(P)SIG the types of triggers its supports as an octet string of length 4 (**trigger_list**) encoded as bit mask with the meaning given in table 30 if set:

Table 30: Trigger types

trigger cause	trigger_list bit #	trigger type
DVB reserved	none	0x00000000
following event	0 (lsb)	0x00000001
future event	1	0x00000002
ECM stream setup	2	0x00000004
access criteria change	3	0x00000008
ECM stream closure	4	0x00000010
flow PID change	5	0x00000020
DVB reserved	6 to 15	0x00000040 to 0x00008000 (powers of 2 only)
user defined	16 to 31 (msb)	0x00010000 to 0x80000000 (powers of 2 only)
combination of any previous	any combination	any value from 0x00000000 to 0xFFFFFFFF

The C(P)SIG communicates to the (P)SIG which triggers it wishes to receive using the same mask.

The trigger information communicated by the (P)SIG to the C(P)SIG includes the following:

- original network identifier as defined by DVB SI (original_network_id);
- network identifier as defined by DVB SI (network_id);
- transport identifier as defined by DVB SI (transport_stream_id);
- service identifier as defined by DVB SI or the MPEG PAT program_number (service_id);
- trigger type coded the same way as the trigger mask defined above;
- either event related data, ECM related data or flow PID change related data depending on the trigger type.

Event related data is defined as follows:

- event identifier as defined in ETSI EN 300 468 [1] (event_id);
- event duration in hours, minutes, seconds coded as six 4-bit BCD digits;
- start time of the event in Modified Julian Date (MJD) and Universal Time, Coordinated (UTC) formats. It is contained in an octet string of length 5. The MJD is coded in the 16 LSBs followed by six 4-bit BCD digits representing UTC;
- private data which correspond to the event private data.

ECM related data is coded as follows:

- elementary stream identifier of the elementary stream to which the ECM stream is attached;
- ECM PID;
- ECM Identifier which corresponds to the ECM channel identifier, the ECM stream identifier, and the Super_CAS identifier;
- access criteria.

Flow-PID change related data is coded as follows:

- flow-type;
- flow-super-CAS-id;

- flow-id;
- flow-PID.

8.2.4 Table Provisioning Transaction Type

The C(P)SIG communicates to the (P)SIG the following information uniquely identifying in a given transport-stream the table the C(P)SIG is requesting, according to table 31:

Original network identifier and transport identifier (original_network_id and transport_stream_id as defined by DVB SI) identifying the transport stream carrying the requested table

Table identifier:

- original network identifier as defined by DVB SI (original_network_id);
- network identifier as defined by DVB SI (network_id);
- transport identifier as defined by DVB SI (transport_stream_id);
- service identifier as defined by DVB SI (service_id) or the MPEG PAT program_number;
- bouquet identifier as defined by DVB SI (bouquet_id);
- event identifier as defined by DVB SI (event_id);
- segment number in an EIT schedule as defined by DVB SI.

Table 31: Table Identifier coding and required parameters

Table	Table_id	Required parameters
all tables: identification of the transport stream carrying the requested table	-	original_network_id transport_stream_id
PAT	0x00	-
CAT	0x01	—
PMT	0x02	service_id (MPEG-2 program number)
NIT actual network	0x40	—
NIT other network	0x41	network_id$_{other}$
BAT	0x4A	bouquet_id
SDT actual TS	0x42	—
SDT other TS	0x46	original_network_id$_{other}$ transport_stream_id$_{other}$
EIT p/f actual TS	0x4E	service_id
EIT p/f other TS	0x4F	original_network_id$_{other}$ transport_stream_id$_{other}$ service_id
EIT schedule actual TS (whole sub-table)	0x50 — 0x5F	service_id
EIT schedule actual TS (single segment)	0x50 — 0x5F	service_id segment_number
EIT schedule actual TS (single event only)	0x50	service_id event_id
EIT schedule other TS (whole sub-table)	0x60 — 0x6F	original_network_id$_{other}$ transport_stream_id$_{other}$ service_id
EIT schedule other TS (single segment)	0x60 — 0x6F	original_network_id$_{other}$ transport_stream_id$_{other}$ service_id segment_number
EIT schedule other TS (single event only)	0x60	original_network_id$_{other}$ transport_stream_id$_{other}$ service_id event_id

The C(P)SIG requests either complete PSI tables or SI sub-tables (see ETSI EN 300 468 [1]). As EIT Schedule sub-tables may be very large, the protocol optionally allows the C(P)SIG to request only the part of the EIT Schedule sub-table that concerns a specific event or a specific segment. In case the request concerns a specific event_id, the value of the table_id field shall be set to 0x50 for the events pertaining to the current TS and 0x60 for the events pertaining to an EIT other.

The table requested is supplied by the (P)SIG to the C(P)SIG if available. The communications mechanism is transport dependent and specified in the next two Sections.

8.2.5 Descriptor Insertion Transaction Type

The C(P)SIG communicates to the (P)SIG the descriptors that the C(P)SIG wants to be inserted in the PSI/SI tables of a given transport stream. This includes all the information that the (P)SIG needs to know where and when to insert the descriptors. The information depends on the type of descriptor inserted and is specified below:

- original network identifier and transport stream identifier (original_network_id and transport_stream_id as defined by DVB SI) identifying the transport stream carrying the table where descriptors are to be inserted;
- trigger identifier if related to a previously transmitted trigger;
- insertion delay type which can be either *immediate* (i.e. no delay) or *synchronized* with the cause of a trigger if the trigger is identified;
- insertion delay: this parameter is significant only if insertion delay type is *synchronized* in seconds; the delay value can be positive in which case it indicates that the synchronization is to happen the delay time after the trigger cause, or negative in which case it indicates that the synchronization is to happen the delay time before the cause;
- location identifier, which identifies the table and descriptor, loop if applicable in which the (P)SIG is to insert the set of descriptors;
- original network identifier as defined by DVB SI (original_network_id);
- network identifier as defined by DVB SI (network_id);
- transport identifier as defined by DVB SI (transport_stream_id);
- service identifier as defined by DVB SI or the MPEG PAT program_number (service_id);
- bouquet identifier as defined by DVB SI (bouquet_id);
- event identifier as defined by DVB SI (event_id);
- elementary stream identifier of the elementary stream to which the ecm stream is attached;
- private data specifier as specified by ETSI EN 300 468 [1] and ETSI ETR 162 [3];
- descriptor to be inserted.

The private_data_specifier parameter enables the C(P)SIG to request the insertion of the set of descriptors within the scope of a private_data_specifier descriptor (as defined in ETSI EN 300 468 [1]), if supported by the head-end.

The insertion_delay_type and insertion_delay parameters provide synchronization information. They indicate to the (P)SIG when to insert the set of descriptors with respect to the ECM stream modification time or the event start time. The insertion_delay parameter indicates the amount of time between the insertion of the table and the insertion of the ECMs in the transport stream. If it is positive, it means that the insertion of the table in the transport stream shall occur after the start of the ECM broadcast. If negative, it means that the modified PSI/SI table shall be broadcast ahead of the ECM modification time in the transport stream.

Table 32 indicates, for each MPEG-2 PSI and DVB SI table, which parameters are needed.

Table 32: Location Identifiers and Required Parameters

Table	Location_id value	Required location parameters
all tables: identification of the transport stream carrying the targetted table	-	original_network_id transport_stream_id
CAT	0x01	—
PMT 1st loop	0x02	service_id (MPEG-2 program number)
PMT 2nd loop	0x03	service_id (MPEG-2 program number) ES_id
NIT 1st loop - actual network	0x04	—
NIT 2nd loop - actual network	0x05	original_network_id$_{\text{(2nd loop)}}$ transport_stream_id$_{\text{(2nd loop)}}$
NIT 1st loop - other network	0x06	network_id$_{\text{other}}$
NIT 2nd loop - other network	0x07	network_id$_{\text{other}}$ original_network_id$_{\text{(2nd loop)}}$ transport_stream_id$_{\text{(2nd loop)}}$
BAT 1st loop	0x08	bouquet_id
BAT 2nd loop	0x09	bouquet_id original_network_id$_{\text{(2nd loop)}}$ transport_stream_id$_{\text{(2nd loop)}}$
SDT (actual TS)	0x0A	service_id
SDT (other TS)	0x0B	original_network_id$_{\text{other}}$ transport_stream_id$_{\text{other}}$ service_id
EIT present/following (actual TS)	0x0C	service_id event_id
EIT present/following (other TS)	0x0D	original_network_id$_{\text{other}}$ transport_stream_id$_{\text{other}}$ service_id event_id
EIT schedule (actual TS)	0x0E	service_id event_id
EIT schedule (other TS)	0x0F	original_network_id$_{\text{other}}$ transport_stream_id$_{\text{other}}$ service_id event_id

The descriptor insertion "location" (PSI/SI table and descriptor loop, if applicable) is unambiguously qualified by the location_id and the location parameters given above. When a given C(P)SIG provides a set of descriptors for insertion, that set replaces the previous set at the same location (in the same private_data_specifier context).

The C(P)SIG can associate the descriptor insertion with a trigger by supplying a trigger_id parameter.

8.2.6 Service Change Transaction Type

The (P)SIG informs the C(P)SIG about a modification in service existence in a transport stream (addition of a service to a transport stream or deletion of a service from a transport stream). More sophisticated functions such as service_id change or the move of a service from one transport stream to another transport stream can be achieved by combining these two basic addition and deletion functions. The (P)SIG can give as well the scheduled day and time of the service change in the transport stream.

8.2.7 Flow PID Provisioning Transaction Type

The PID provision functionality is available only in the C(P)SIG ⇔ (P)SIG protocol. As used in this section, the word "flow" concerns an ECM stream, an EMM stream or a private data stream in a particular transport stream. A (P)SIG can provide a C(P)SIG with the PID of a flow by one of these three ways:

- by explicit request from the C(P)SIG to the (P)SIG: the flow PID provisioning functionality described in this section;

- by triggering the C(P)SIG when a change of PID occurs in a flow;
- for an ECM stream, by triggering the C(P)SIG when a new ECM stream is created.

A flow is unambiguously known by the head-end and by the CAS by using its unique identifier defined as the combination of:

- the type of the flow: flow_type = ECM, EMM or private data;
- the Super CAS_id this flow belongs to: flow_super_CAS_id;
- an individual number: flow_id; for a flow_super_CAS_id and a flow_type, the flow_id shall be unique.

Such a combination identifies uniquely a flow across all the Simulcrypt protocols used in a real configuration and across all the transport streams generated by the head-end.

Depending on the type of the flow, its unique identifier is assigned by the head-end (ECM) or by the CAS (EMM, private data) when the flow is created.

When a unique identifier exists in the system, the head-end is assumed to know which flow_PID value is associated with.

The consequences are that:

- the unique identifier of an ECM flow (absolute reference) is carried in the ECMG protocol in addition to the channel_id and stream_id parameters (references relative to the current ECMG-SCS connection); it shall be assigned by the head-end and provided to the ECMG at stream setup;
- the unique identifier of an EMM/private data flow (absolute reference) is carried in the EMMG/PDG protocol in addition to the channel_id and stream_id parameters (references relative to the current EMMG/PDG-MUX connection); it shall be assigned by the EMMG/PDG and provided to the head-end at stream setup;
- in the C(P)SIG protocol, the PID provisioning functionality allows the C(P)SIG to get the flow_PID value of a particular flow identified by its unique identifier;
- in the C(P)SIG protocol, the flow-PID-change trigger functionality allows the (P)SIG to trigger the C(P)SIG when a change occurs for the flow_PID value of a particular flow identified by its unique identifier;
- in the C(P)SIG protocol, the flow-PID of an ECM stream (i.e., ECM-PID) is given among other ECM related parameters by the (P)SIG when the ECM stream is created;
- it is assumed that a CAS can support internal links between ECMG and C(P)SIG or between EMMG/PDG and C(P)SIG to manage consistently the unique identifiers it is concerned by.

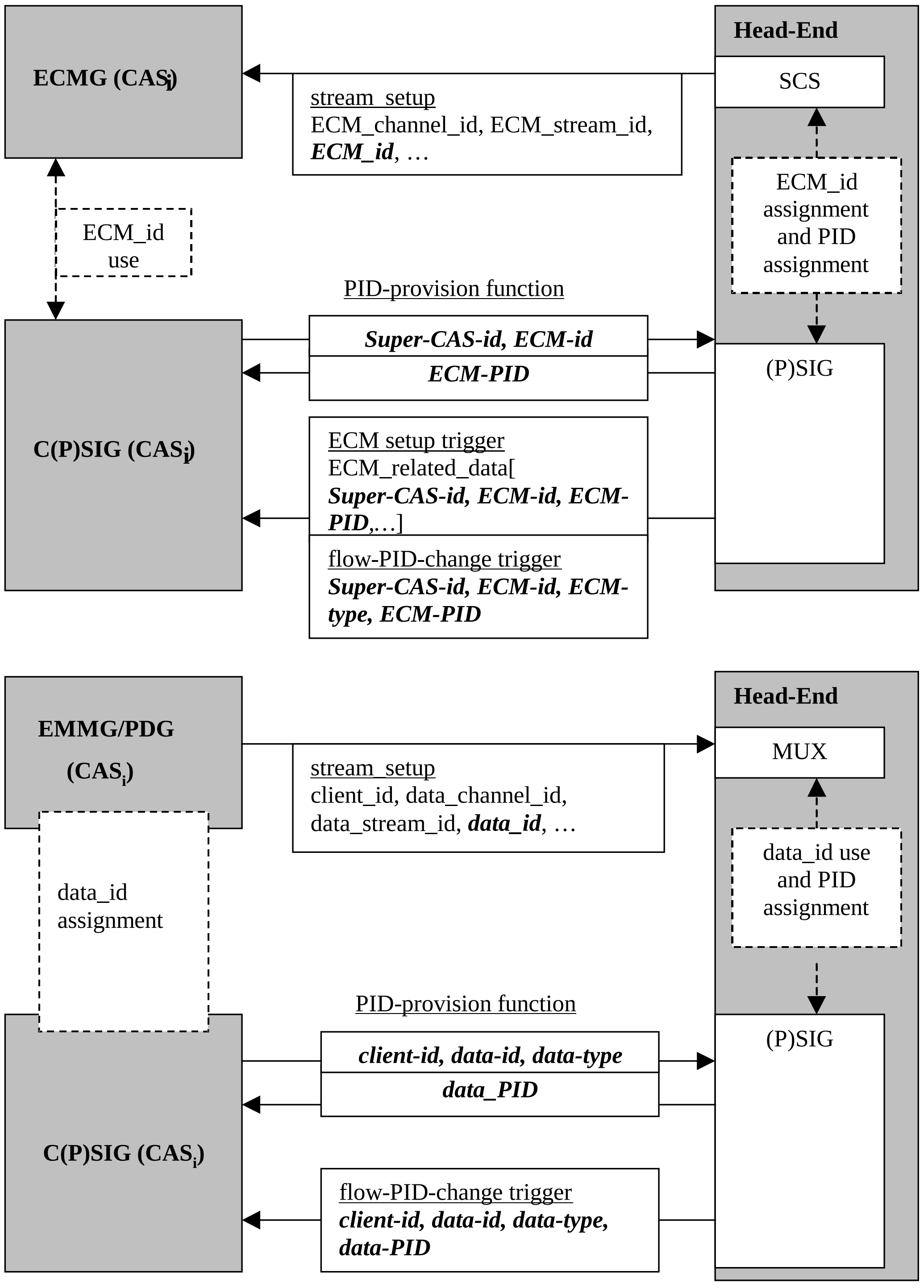
Figure 11 depicts this mechanism for ECM stream (in this case ECM_id and flow_id are homonym) and for EMM/private data stream (in this case data_id and flow_id are homonym, data_PID and flow_PID as well).

In the ECMG protocol, the type of the flow is implicitly ECM; the flow_super_CAS_id is the Super_CAS_id, which is implicitly in the channel_id. Hence in this protocol only the flow_id shall be given explicitly.

In the EMMG/PDG protocol, the type of the flow is given in the data_type parameter (EMM or PDG); the flow_super_CAS_id is the client_id, which is implicitly in the channel_id. Hence in this protocol only the flow_id shall be given explicitly.

In the C(P)SIG protocol:

- in the PID-provision function, the three parameters flow_type, flow_super_CAS_id, flow_id shall be given explicitly; in particular, the flow_super_CAS_id is not always the custom_CAS_id;
- in the ECM-setup trigger function, the flow_type is implicitly ECM; so both parameters flow_super_CAS_id and flow_id shall be given explicitly; in particular, the flow_super_CAS_id it not always the custom_CAS_id;
- in the flow-PID-change trigger function, the three parameters flow_type, flow_super_CAS_id and flow_id shall be given explicitly.



NOTE: The flow PID provisioning transaction allows a CAS to receive PID values for particular streams. The CA System is responsible for the (possibly undesirable) consequences, when using these PIDs. In particular, inserting these PIDs in private data can cause problems when re-multiplexing occurs.

Figure 11: PID provision mechanism for ECM and EMM/private data streams

8.2.8 Implementation of the C(P)SIG ↔ (P)SIG protocol

The generic C(P)SIG ⇔ (P)SIG interface defined in the sections above shall be implemented in one of the two following ways:

- in a connection-oriented protocol: in this case the implementation shall be compliant with the subclause 8.3;

- in a SIMF-based protocol: in this case, the implementation shall be compliant with the subclause 8.4.

8.3 Connection-oriented protocol

This section specifies the connection-oriented instance of the generic C(P)SIG ⇔ (P)SIG protocol defined in the subclause 8.2.

In such an implementation the C(P)SIG and (P)SIG processes communicate with each other via a connection-oriented protocol.

8.3.1 Overview of the C(P)SIG ↔ (P)SIG connection-oriented protocol

8.3.1.1 Principles

In this connection-oriented protocol, the (P)SIG is the client and the C(P)SIG is the server. The (P)SIG has a prior knowledge of the mapping between Custom_CAS_id and the IP addresses and port numbers of the C(P)SIG. Once the (P)SIG has established a TCP connection with the C(P)SIG, it opens a channel (see subclause 8.3.1.2) dedicated to the Custom_CAS_id. Then it opens as many streams (see subclause 8.3.1.3) as transport streams it manages. When a stream is open, the C(P)SIG may, for the relevant transport stream, request for PSI and SI tables, be triggered on particular conditions and insert private descriptors.

To achieve this, this connection-oriented protocol offers a set of messages at channel level and stream level, which can be used according to a specific state machine. Messages and state machine are described in the following sections.

The general format of the messages is defined in subclause 4.4.1.The following points also apply to all C(P)SIG ⇔ (P)SIG messages specified by this connection-oriented protocol:

- depending on their type or their use in the state machine, certain messages may be sent by a (P)SIG only (noted "C(P)SIG ⇐ (P)SIG"), others may be sent by a C(P)SIG only (noted "C(P)SIG ⇒ (P)SIG"), the remainder may be sent by either (noted "C(P)SIG ⇔ (P)SIG"). This is clearly indicated in the descriptions that follow;
- a message whose name ends with "_request" shall be answered by the receiver, using the corresponding message whose name ends with "_response";
- all messages include a 16-bit transaction_id, which is used to match messages with their responses. The transaction_id is assigned cyclically by the originator of the message (C(P)SIG or (P)SIG), and is unique per channel. Values of 0 through 32767 are reserved for C(P)SIG use; the (P)SIG shall use values 32 768 through 65 535;
- though a real implementation will include time_out management, the present document does not specify a time-out for a response, nor the actions (e.g. failure or retry) to be taken if a message is not received within a given time. The present document also does not define any specific mechanisms to ensure message delivery other than TCP itself.

8.3.1.2 Channels

8.3.1.2.1 Definition and types

The logical connection between a C(P)SIG and a (P)SIG is comprised of one and only one channel per CAS. There is only one channel per TCP connection.

Since a channel interconnects a single C(P)SIG with a single (P)SIG, any of seven types of channels is possible; refer to the leftmost diagram in figure 12. The channel types that can actually be established depend on the respective configurations of the CAS and head-end processes. For example, a CPSISIG may be interconnected with a {PSIG+SIG} with either or both of the connections depicted in the rightmost diagram in figure 12.

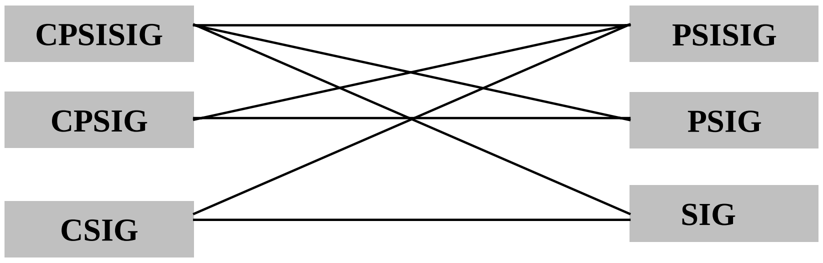


Figure 12: All types of C(P)SIG ⇔ (P)SIG channels

Each channel is identified by a 2-byte custom_channel_id. This value is unique per C(P)SIG (custom_CAS_id).

8.3.1.2.2 Channel establishment

The head-end is responsible for establishing all channels at system initialization time, or when a CAS is newly interfaced to the head-end. Accordingly, the head-end (and possibly the CAS) shall have prior information of the channels to be established, as well as all parameters required for connection.

This channel information is determined by commercial agreement, per business and technical requirements. The manner in which this information is maintained and used is beyond the scope of the present document.

8.3.1.3 Streams

8.3.1.3.1 Definition

As described in subclause 8.3.1.2, a channel connects a C(P)SIG with a (P)SIG. A channel logically connects a C(P)SIG with one or more TSs.

A **stream** is a logical connection between a C(P)SIG and a (P)SIG, which serves a single TS. As such, a stream "belongs" to a channel. *Within a channel*, the relationship between stream and TS is *1-to-1* (or *0-to-1:* a TS need not have any stream associated with it). Figure 13 illustrates this relationship.

The primary purpose of a stream is to serve as the logical conduit from a C(P)SIG to a (P)SIG, for the transmission of CAS-specific private data in a given TS. At least one stream shall be established, and operational, for a C(P)SIG to generate private data in any TS.

As a stream "belongs" to a channel, and a channel "belongs" to a CAS, a stream cannot convey information pertaining to more than one CAS.

Each stream is identified by a 2-byte custom_stream_id. This value is unique per channel.

The value of custom_stream_id uniquely identifies a TS. This TS is identified by the same value of custom_stream_id across all channels.

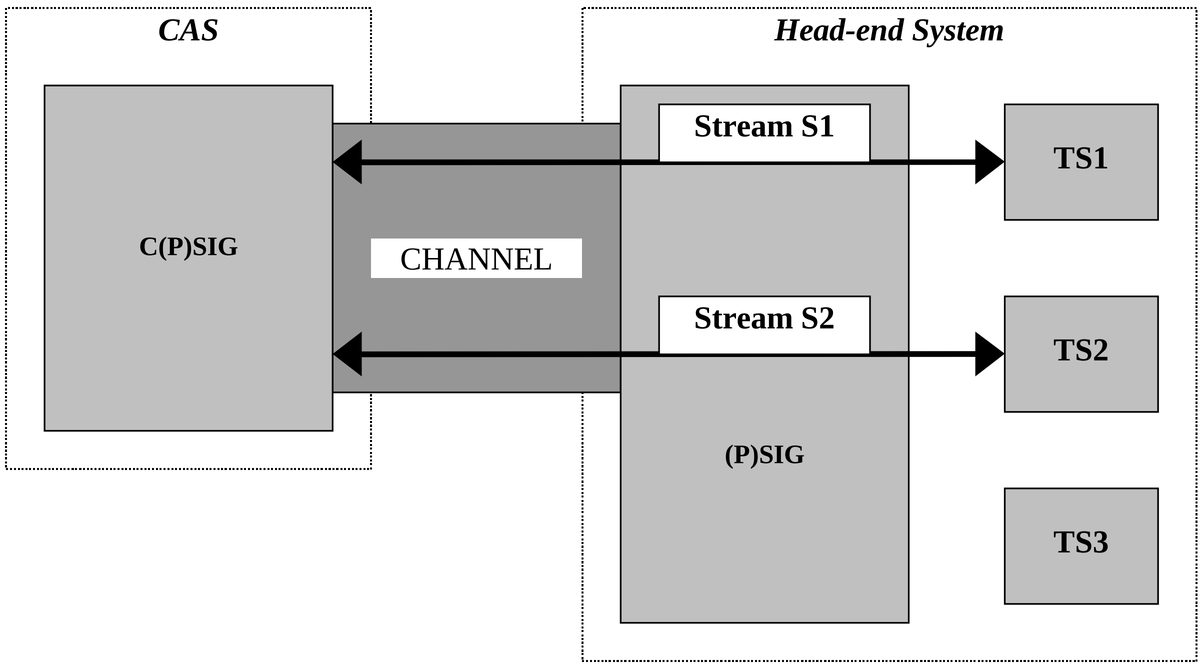


Figure 13: Stream interconnections (simple example)

Multiple streams shall be established per channel when the C(P)SIG needs to generate CAS-specific private data in more than one TS. Multiple streams corresponding to one TS cannot be configured for other reasons, e.g. performance, capacity or redundancy.

Refer also to annex F for a sample channel and stream configuration for a head-end with two CASs.

8.3.1.3.2 Stream establishment

The head-end is responsible for establishing all streams at system initialization time, or when a CAS is inserted to the head-end. Accordingly, the head-end (and possibly the CAS) shall have prior information of the streams to be established, as well as all parameters required for connection.

This stream information is determined by commercial agreement, per business and technical requirements. The manner in which this information is maintained and used is beyond the scope of the present document.

8.3.1.4 C(P)SIG ↔ (P)SIG message lists

There are two general classes of messages:

- **Channel-level messages** pertain to the configuration or status of a channel. Their semantic scope is a channel, and they do not reference streams or DVB services;

channel_setup	0x0301	C(P)SIG	⇐	(P)SIG
channel_status	0x0302	C(P)SIG	⇔	(P)SIG
channel_test	0x0303	C(P)SIG	⇔	(P)SIG
channel_close	0x0304	C(P)SIG	⇐	(P)SIG
channel_error	0x0305	C(P)SIG	⇔	(P)SIG

- **Stream-level messages** pertain to the configuration or status of a stream, the supply of currently transmitted PSI/SI, and to the transmission of CAS-specific private data for insertion in MPEG-2/DVB tables. These messages require a channel that is established and operational. Their semantic scope is a stream, a transport stream (TS) or a DVB (or MPEG-2) service, depending on the message.

stream_setup	0x0311	C(P)SIG	⇐	(P)SIG
stream_status	0x0312	C(P)SIG	⇔	(P)SIG
stream_test	0x0313	C(P)SIG	⇔	(P)SIG
stream_close	0x0314	C(P)SIG	⇐	(P)SIG
stream_close_request	0x0315	C(P)SIG	⇒	(P)SIG
stream_close_response	0x0316	C(P)SIG	⇐	(P)SIG
stream_error	0x0317	C(P)SIG	⇔	(P)SIG
stream_service_change	0x0318	C(P)SIG	⇐	(P)SIG
stream_trigger_enable_request	0x0319	C(P)SIG	⇒	(P)SIG
stream_trigger_enable_response	0x031A	C(P)SIG	⇐	(P)SIG
trigger	0x031B	C(P)SIG	⇐	(P)SIG
table_request	0x031C	C(P)SIG	⇒	(P)SIG
table_response	0x031D	C(P)SIG	⇐	(P)SIG
descriptor_insert_request	0x031E	C(P)SIG	⇒	(P)SIG
descriptor_insert_response	0x031F	C(P)SIG	⇐	(P)SIG
PID_provision_request	0x0320	C(P)SIG	⇒	(P)SIG
PID_provision_response	0x0321	C(P)SIG	⇐	(P)SIG

All channel-level and stream-level messages shall be supported by all C(P)SIG and (P)SIG processes, with any semantic exceptions noted in the present document.

8.3.1.5 Protocol state machines definition

This protocol is based on two state machines. The first state machine defines the transitions associated with each channel. The second state machine defines the transitions associated with each stream; this state machine is valid only for a channel in the Channel Open state.

The state machines defines:

- **States:** A total of eleven states is defined, four for the channel state machine, and seven for the stream state machine;
- **Transitions:** the set of messages that cause changes of state in the state machine.

The basic functionality of each message is presented in order to understand state machine operation. Precise and detailed syntax and semantics of each message are presented in subclause 8.3.2.

Independence and scalability: all channels in a complete system are mutually independent; the same is true of streams. Therefore, both state machines are scalable: concurrent state machines can be launched for each channel and stream in the system, whether the head-end is hosting one or multiple CASs.

8.3.1.6 Channel state machine

As described in subclause 8.3.1.2.2, the head-end establishes and maintains one or more channels between (P)SIG and C(P)SIG processes. This section presents the channel state machine, which defines the sequence of channel-level messages that shall be used to establish and maintain a **single channel**.

Channels may be established in any order. In addition, a (P)SIG needs not wait for the establishment of one channel to be complete, before commencing the establishment of another channel. Such considerations are out of the scope of the present document.

The channel state machine is found in figure 14. Each state found in this state machine is defined in subclauses 8.3.1.6.1 to 8.3.1.6.4.

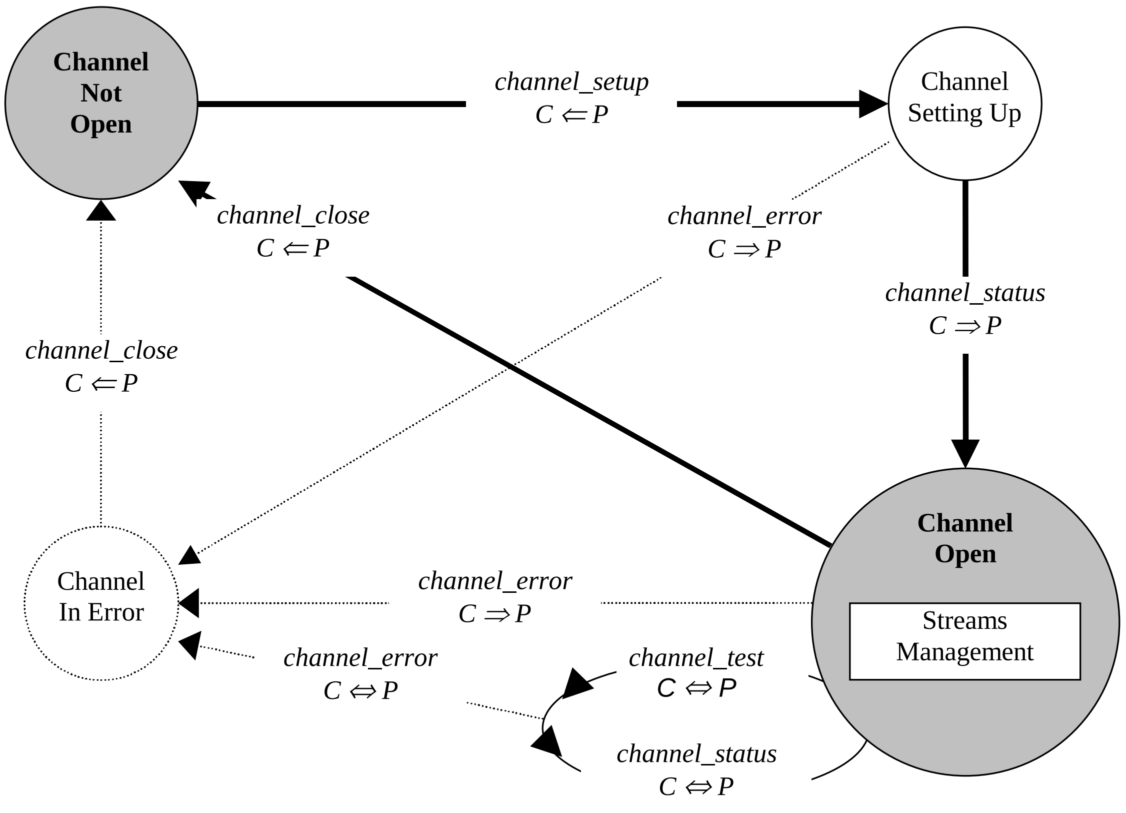


Figure 14: C(P)SIG ⇔ (P)SIG channel state machine

8.3.1.6.1 Channel Not Open

This state represents the initialization of the channel state machine. At this point, a TCP connection is assumed to be established and the channel has either not been initialized, or has been closed.

- The (P)SIG initializes a channel by sending a **channel_setup message** to the C(P)SIG on the other end of the channel. Channel_setup is the only permissible message in the Channel Not Open state. Parameters of channel_setup identify the (P)SIG type (PSISIG, PSIG or SIG) and the kinds of triggers supported by the (P)SIG for the new channel. Transmission and receipt of channel_setup move the state machine to the Channel Setting Up state.

8.3.1.6.2 Channel Setting Up

From this state, the C(P)SIG shall respond with either a channel_status or a channel_error message:

- The **channel_status** message acknowledges successful channel establishment, and that the channel is open. The C(P)SIG also indicates, via this message, the maximum number of streams that can be supported on the new channel. Transmission and receipt of channel_status move the state machine to the Channel Open state.

- The **channel_error** message acknowledges that the C(P)SIG could not open the channel, and that the channel shall be closed by the (P)SIG. One or more error codes explain the failure. Transmission and receipt of Channel_error move the state machine to the Channel In Error state.

8.3.1.6.3 Channel Open

This state represents the steady-state operation of the channel state machine. As long as the channel is open and error-free, streams may be opened, used and closed, per the stream state machine defined in subclause 8.3.1.7: the stream state machine defines the stream-level and data-level messages that can be sent on a stream within the channel, per the state of that stream.

Four kinds of channel-level messages can be sent while in Channel Open state:

- Either the C(P)SIG or the (P)SIG can send a **channel_test** message, in order to verify the error-free operation of the channel. This does not change the state of the channel state machine;

- If the channel is in an error-free situation, the receiver of the channel_test message shall reply with a **channel_status** message. This does not change the state of the channel state machine. Channel_status may be sent only in response to channel_test;

- If the C(P)SIG encounters an unrecoverable channel error at any other time, it shall send the (P)SIG a **channel_error** message. If the stream has unrecoverable errors, the receiver of the channel_test message shall reply with a **channel_error** message. One or more error codes explain the failure. Transmission and receipt of channel_error move the state machine to the Channel In Error state. Channel_error may be sent at any time from the Channel Open state;

- If the (P)SIG wants to close the channel for any reason, it shall send the C(P)SIG a **channel_close** message. Receipt of channel_close moves the state machine to the Channel Not Open state. Channel_close may be sent at any time from this state.

Channel_close also causes the immediate closure of all streams open in the channel.

8.3.1.6.4 Channel In Error

This temporary and short-lived state is used only to represent the fact that the C(P)SIG has encountered and reported an unrecoverable channel error. The (P)SIG shall close the channel:

- The (P)SIG sends a **channel_close** message to the C(P)SIG. Transmission and receipt of channel_close move the state machine to the Channel Not Open state.

Channel_close also causes the immediate closure of all streams open in the channel.

8.3.1.7 Stream state machine

As described in subclause 8.3.1.3.2, the head-end establishes one or more streams within a channel. This section presents the stream state machine, which defines the sequence of stream-level messages that shall be used to establish, maintain and use a **single stream** within a channel.

Streams may be established in any order (within a given channel, or globally). In addition, a (P)SIG needs not wait for the establishment of one stream to be complete, before commencing the establishment of another stream. Such considerations are out of the scope of the present document.

The channel shall be in the Channel Open state (see subclause 8.3.1.6.3) for a (P)SIG to initiate a stream state machine. The channel state machine, as defined in subclause 8.3.1.6, continues to operate in Channel Open state during the operation of the stream state machine. Accordingly, both C(P)SIG and (P)SIG processes shall properly handle any and all channel-level messages valid in Channel Open state (these messages are not shown in the stream state machine).

Closure of a channel (Channel Not Open state) causes the immediate closure of all streams open in the channel (reset of the stream state machine to Stream Not Open state).

The stream state machine is found in figure 15. Each state found in this state machine is defined in subclauses 8.3.1.7.1 to 8.3.1.7.7.

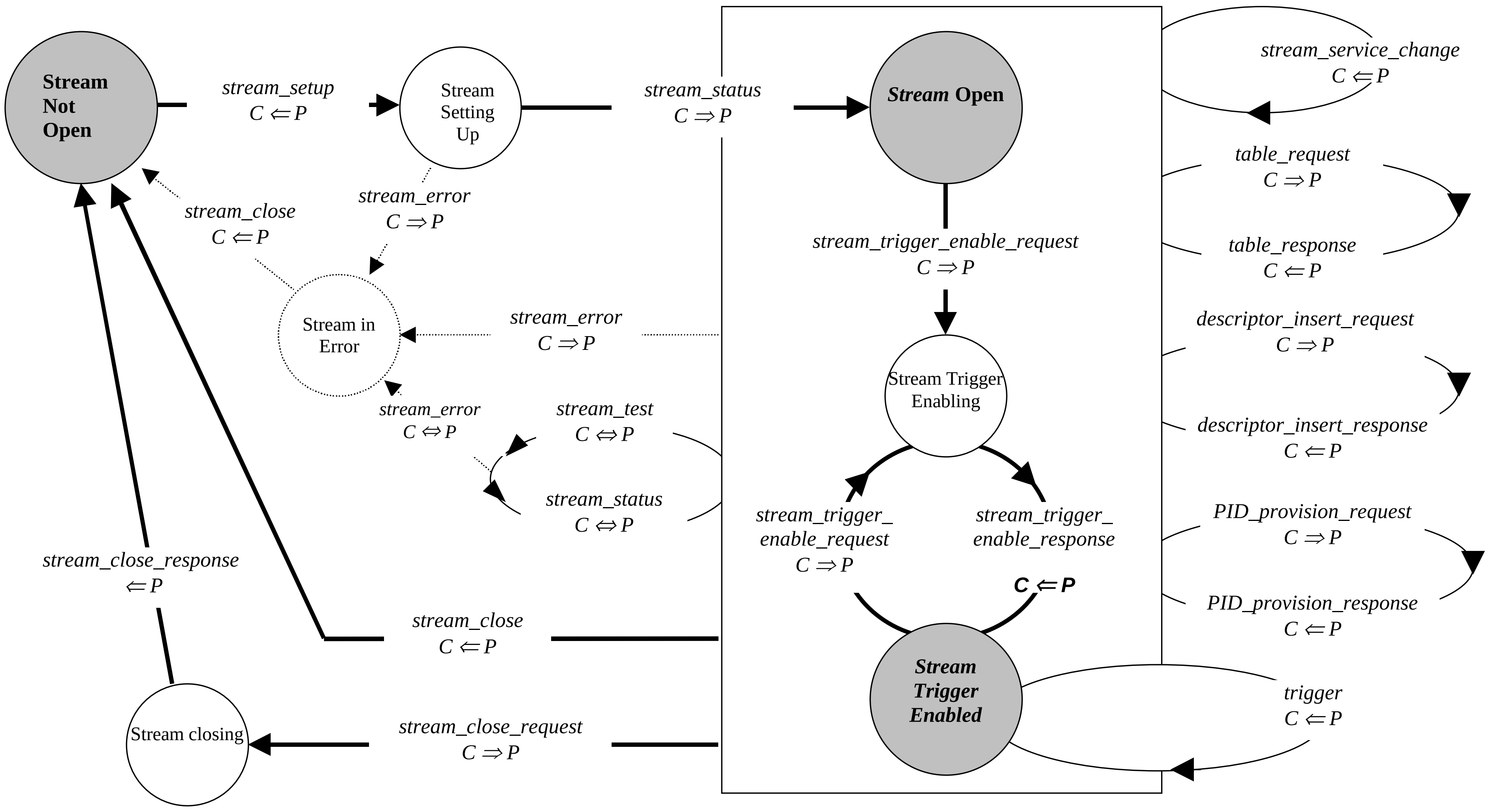


Figure 15: C(P)SIG ⇔ (P)SIG stream state machine

8.3.1.7.1 Stream Not Open

This state represents the initialization of the stream state machine. At this point, the stream has either not been initialized, or has been closed. The channel in which the stream is found shall be in the Channel Open state in order to proceed.

The (P)SIG initializes a stream by sending a **stream_setup message** to the C(P)SIG on the other end of the stream. stream_setup is the only permissible message in the Stream Not Open state. Parameters of stream_setup identify the transport stream (TS) associated with the stream, and list every service in the TS at the time the stream_setup is sent. Transmission and receipt of stream_setup move the state machine to the Stream Setting Up state.

8.3.1.7.2 Stream Setting Up

From this temporary and short-lived state, the C(P)SIG shall respond with either a stream_status or a stream_error message:

- The **stream_status** message acknowledges successful stream establishment, and that the stream is open. Transmission and receipt of stream_status move the state machine to the Stream Open state;
- The **stream_error** message acknowledges that the C(P)SIG could not open the stream, and that the stream shall be closed by the (P)SIG. One or more error codes explain the failure. Transmission and receipt of stream_error move the state machine to the Stream In Error state.

8.3.1.7.3 Stream Open

The stream is open and operational, and capable of performing all operations with exception of action triggering by the (P)SIG. Thirteen kinds of stream-level messages can be sent while in Stream Open state:

- The C(P)SIG sends a **stream_trigger_enable_request** message to enable the receipt of (P)SIG action triggers in the stream. Parameters of stream_trigger_enable_request indicate, for any or all services in the TS, which kinds of triggers the C(P)SIG wants to receive, and how long in advance of the action the C(P)SIG wants to be triggered. Transmission and receipt of stream_trigger_enable_request move the state machine to the Stream Trigger Enabling state;
- The C(P)SIG sends a **table_request** message to request data from any PSI, SI or private table currently transmitted. This does not change the state of the stream state machine. The (P)SIG shall respond to each table_request message with a table_response message;
- The (P)SIG sends a **table_response** message to supply data from any PSI, SI or private table currently transmitted, as requested by the C(P)SIG in a table_request message. This does not change the state of the stream state machine. However, *for a given stream,* the (P)SIG shall respond to table_request messages in the order received. Table_response may be sent only in response to table_request;
- The C(P)SIG sends a **descriptor_insert_request** message to request the insertion of CAS-specific private data in the TS. The private data is a list (possibly empty) of (a) private descriptors, and/or (b) private_data_bytes for a PAT or PMT CA_descriptor. Other parameters indicate the precise location within the TS in which to insert the data, and, optionally, with which triggered action the data insertion should be synchronized. The (P)SIG shall respond to each descriptor_insert_request message with a descriptor_insert_response message. The descriptor_insert_request message may be sent at any time from this state. This message does not change the state of the stream state machine;
- The (P)SIG sends a **descriptor_insert_response** message in response to the descriptor_insert_request message. Parameters indicate either successful data insertion in the TS or reasons for failure. The descriptor_insert_response message does not change the state of the stream state machine. However, *for a given stream,* the (P)SIG shall respond to descriptor_insert_request messages in the order received. Descriptor_insert_response may be sent only in response to descriptor_insert_request;

- The C(P)SIG sends a **PID_provision_request** message to request from the (P)SIG the PID value the head-end has assigned to a particular stream; this stream is identified by a unique identifier. The (P)SIG shall respond to each PID_provision_request message with a PID_provision_response message. The PID_provision_request message may be sent at any time from this state. This message does not change the state of the stream state machine;

- The (P)SIG sends a **PID_provision_response** message to provide the C(P)SIG with the PID value requested by the C(P)SIG in a PID_provision_request message. The PID_provision_response message does not change the state of the stream state machine. PID_provision_response may be sent only in response to PID_provision_request;

- The (P)SIG sends a **stream_service_change** message to signal the addition, deletion or move (TS and/or service reassignment) of a service. This message is not acknowledged by the C(P)SIG, and does not change the state of the stream state machine. Note that the head-end (any (P)SIG) shall send this message to all C(P)SIG processes operating in the Stream Open or Stream Trigger-Enabled state;

- If the (P)SIG wants to close the stream for any reason, it sends a **stream_close** message. Transmission and receipt of stream_close move the state machine directly to the Stream Not Open state. Stream_close may be sent at any time from this state. This message is not acknowledged by the C(P)SIG;

- If the C(P)SIG wants to close the stream for any reason, it sends a **stream_close_request** message. Transmission and receipt of stream_close_request move the state machine to the Stream Closing state. Stream_close_request may be sent at any time from this state;

- Either the C(P)SIG or the (P)SIG can send a **stream_test** message, in order to verify the error-free operation of the stream. This does not change the state of the stream state machine. Stream_test may be sent at any time from this state;

- If the stream is in an error-free situation, the receiver of the stream_test message shall reply with a **stream_status** message. This does not change the state of the stream state machine. Stream_status may be sent only in response to stream_test;

- If the stream has unrecoverable errors, the receiver of the stream_test message shall reply with a **stream_error** message. One or more error codes explain the failure. Transmission and receipt of stream_error move the state machine to the Stream In Error state.

8.3.1.7.4 Stream Trigger Enabling

This temporary and short-lived state is entered only after the C(P)SIG sends a stream_trigger_enable_request message while in the Stream Open state or in Stream Trigger Enabled state. One kind of stream-level messages can be sent while in Stream Trigger Enabling state:

- The (P)SIG sends a **stream_trigger_enable_response** message in reply to the C(P)SIG. Parameters of stream_trigger_enable_response indicate, for any or all services in the TS, which kinds of triggers the C(P)SIG *can actually* receive, and *approximately* how long in advance of the action the C(P)SIG *will actually* be triggered. Transmission and receipt of stream_trigger_enable_response move the state machine to the Stream Trigger-Enabled state (whether or not any triggers are really enabled). Stream_trigger_enable_response may be sent only in response to stream_trigger_enable_request.

8.3.1.7.5 Stream Trigger-Enabled

This state represents the normal steady-state operation of the stream state machine. The stream is open and operational, and capable of performing all operations, *including* (a) action triggering by the (P)SIG, and (b) requests for CAS-specific private data insertion by the C(P)SIG.

Fourteen kinds of stream-level messages can be sent while in Stream Trigger-Enabled state. The first thirteen are the same as for the Stream Open state; see subclause 8.3.1.7.3. The additional type of stream-level messages is as follows:

- the (P)SIG sends a **trigger** message to signal an action of one of the following types:
  - new DVB SI EIT Following event;
  - new head-end information about a future DVB SI EIT event;
  - creation, modification or closure of an ECM stream;
  - modification of an ECM, EMM or private data PID;
  - user-defined (per commercial agreement).

Parameters indicate the trigger type, and all information required to precisely qualify the action that caused the trigger.

The trigger message is not explicitly acknowledged by the C(P)SIG. However, the trigger shall be referenced by a subsequent descriptor_insert_request message (see below) if the C(P)SIG wants to synchronize private data transmission with the triggering action.

The trigger message may be sent at any time from this state. This message does not change the state of the stream state machine.

8.3.1.7.6 Stream In Error

This temporary and short-lived state is used only to represent the fact that the C(P)SIG has encountered and reported an unrecoverable stream error:

- The (P)SIG sends a **stream_close** message to the C(P)SIG. Transmission and receipt of stream_close move the state machine to the Stream Not Open state.

8.3.1.7.7 Stream Closing

This temporary and short-lived state is used only to represent the fact that the C(P)SIG has requested closure of the stream:

- The (P)SIG sends a **stream_close_response** message to the C(P)SIG, to confirm closure of the stream. Transmission and receipt of stream_close_response move the state machine to the Stream Not Open state.

8.3.1.8 Summary of messages permissible in each state

Table 33 provides a listing of the channel-level and stream-level messages that may be generated in each of the states of both state machines.

Table 33: Message/state cross-reference for the C(P)SIG ⇔ (P)SIG state machines

Message	State										
	Channel (assumes Stream Not Open)				Stream (assumes Channel Open)						
	Not Open	Setting Up	In Error	Open	Not Open	Setting Up	In Error	Closing	Open	Trigger Enabling	Trigger Enabled
channel_setup	X										
channel_status		X		X	X	X	X	X	X	X	X
channel_test				X	X	X	X	X	X	X	X
channel_close			X	X	X	X	X	X	X	X	X
channel_error		X		X	X	X	X	X	X	X	X
stream_setup					X						
stream_status						X			X		X
stream_test									X		X
stream_close							X		X		X
stream_close_request									X		X
stream_close_response								X			
stream_error						X			X		X
stream_trigger_enable_request									X		X
stream_trigger_enable_response										X	
stream_service_change									X		X
trigger											X
table_request									X		X
table_response									X		X
descriptor_insert_request									X		X
descriptor_insert_response									X		X
PID_provision_request									X		X
PID_provision_response									X		X

8.3.2 C(P)SIG ↔ (P)SIG message syntax and semantics

This section provides precise syntax and semantics for each of the messages in the C(P)SIG ⇔ (P)SIG connection-oriented protocol.

8.3.2.1 List of message parameters for the C(P)SIG ↔ (P)SIG protocol

Table 34: Parameter syntax in C(P)SIG ⇔ (P)SIG messaged-based protocol

Parameter_type	Parameter	Type (/ Units)	Length (bytes)
0x000D	access_criteria	user defined	variable
0x0100	bouquet_id	uimsbf	2
0x0101	CA_descriptor_insertion_mode	uimsbf	1
0x0102	custom_CAS_id	uimsbf	4
0x0103	custom_channel_id	uimsbf	2
0x0104	custom_stream_id	uimsbf	2
0x0105	descriptor	Per MPEG/DVB; see ISO/IEC 13818-1 [7] and ETSI EN 300 468 [1]	variable
0x0106	descriptor_insert_status	uimsbf	1
0x0107	duration	uimsbf	3
0x0108	ECM_related_data ES_id flow_super_CAS_id flow_id flow_PID access_criteria	-	variable
0x0109	DVB reserved	-	-
0x010A	DVB reserved	-	-
0x010B	ES_id	uimsbf	2
0x010C	event_id	uimsbf	2
0x010D	event_related_data event_id duration start_time private_data	-	variable
0x010E	flow_id	uimsbf	2
0x010F	flow_PID	uimsbf	2
0x0110	flow_PID_change_related_data flow_type flow_super_CAS_id flow_id flow_PID	-	9
0x0111	flow_super_CAS_id	uimsbf	4
0x0112	flow_type	uimsbf	1
0x0113	insertion_delay	tcimsbf (/ ms)	2
0x0114	insertion_delay_type	uimsbf	1
0x0115	last_section_indicator	boolean	1
0x0116	location_id	uimsbf	1
0x0117	max_comp_time	uimsbf (/ sec)	2
0x0118	max_streams	uimsbf	2
0x0119	MPEG_section	Per MPEG/DVB; see ISO/IEC 13818-1 [7] and ETSI EN 300 468 [1]	variable
0x011A	network_id	uimsbf	2
0x011B	original_network_id	uimsbf	2
0x011C	private_data	user-defined	variable
0x011D	private_data_specifier	uimsbf	4
0x011E	(P)SIG_type	uimsbf	1
0x011F	segment_number	uimsbf	1
0x0120	service_id	uimsbf	2
0x0121	service_parameters service_id trigger_list max_comp_time	-	8
0x0122	start_time	bslbf	5
0x0123	stream_change_timestamp	bslbf	5
0x0124	stream_change_type	uimsbf	1
0x0125	table_id	uimsbf	1
0x0126	transaction_id	uimsbf	2
0x0127	transport_stream_id	uimsbf	2
0x0128	trigger_id	uimsbf	2

Parameter_type	Parameter	Type (/ Units)	Length (bytes)
0x0129 0x012A	trigger_list trigger_type	bslbf uimsbf	4 4
0x012B to 0x6FFF	DVB reserved	-	-
0x7000 0x7001	error_status error_information	TBD user defined	2 variable
0x7002 to 0x7FFF 0x8000 to 0xFFFF	DVB reserved user defined	- -	- -

8.3.2.2 Parameter semantics

This section gives the semantic of the parameters and fields used in this protocol. When depending on the message context, some parameters can be completed in each message description in further sections.

- **access_criteria:** this parameter carries the access criteria concerning an ECM stream;
- **bouquet_id:** this parameter is the value of the bouquet_id field as defined by ETSI EN 300 468 [1];
- **CA_descriptor_insertion_mode:** this parameter is provided by a CPSIG/CPSISIG and indicates whether a PSIG/PSISIG has to insert the skeleton of CA_descriptors; such a skeleton corresponds to the CA_descriptor as defined in table 2.51 of ISO/IEC 13818-1 [7] with an empty private_data_byte part; it can have the following values:
  - **0x01:** the skeleton of the CA_descriptor is always inserted by the PSIG/PSISIG;
  - **0x02:** no CA_descriptor skeleton is inserted by the PSIG/PSISIG; the CPSIG/CPSISIG is always responsible for the generation of this descriptor;
  - **other values:** DVB reserved.
- **custom_CAS_id:** this parameter is the unique identifier of a C(P)SIG sharing a channel with a (P)SIG;
- **custom_channel_id:** this parameter is the identifier of a channel established by a (P)SIG with a C(P)SIG. It is unique per custom_CAS_id;
- **custom_stream_id:** this parameter is the identifier of a stream established by a (P)SIG with a C(P)SIG for a transport stream. It is unique per channel;
- **descriptor:** this field represents an instance of a descriptor provided by a C(P)SIG and to be inserted by the (P)SIG;
- **descriptor_insert_status:** this parameter indicates if a descriptor insertion requested by a C(P)SIG to a (P)SIG has been done correctly or has failed, according to the following values:
  - **0x00:** insertion has been done correctly; this value is significant after the actual insertion;
  - **0x01:** insertion failed because of the request was inconsistent (e.g. bad value for location_id, lack of some parameters for a given location_id); this value is significant immediately after the descriptor insertion request and prior the actual insertion;
  - **0x02:** insertion has failed because the target table is not generated by the (P)SIG; this value is significant immediately after the descriptor insertion request and prior the actual insertion;
  - **0x03:** insertion has failed because an error occurred at the moment of insertion of the descriptors in the table (e.g., missing space); this value is significant after the actual insertion attempt;
  - **other values**: DVB reserved.
- **duration**: this parameter contains the duration of the event in hours, minutes and seconds. The format is six 4-bit BCD digits. Example: 01:45:30 is coded as 0x014530;
- **ECM_related_data**: this field provides the C(P)SIG with the necessary information concerning an ECM stream that is going to be opened, closed or modified, whose PID is about to change, or that is attached to an event. It includes the following subfields:

  - ES_id;
  - flow_id: the identifier of the ECM stream;
  - flow_super_CAS_id: the Super_CAS_id the ECM belongs to;
  - flow_PID: the PID value of the concerned ECM stream;
  - access_criteria.

- **error_information**: this parameter describes an error reason; its values are given in subclause 8.3.4.16;
- **error_status**: this parameter describes an error reason; its values are given in subclause 8.3.4.16;
- **ES_id**: this parameter is the identifier of an Elementary Stream to which the ECM stream is attached. It is equal to the rank of the description of the Elementary Stream in the PMT. The value 0 identifies the whole service;
- **event_id**: this parameter is the value of the event_id field as defined by ETSI EN 300 468 [1];
- **event_related_data**: this field describes a DVB-event. The trigger message does not contain this parameter when the value of trigger type is different from 'following event' and 'future event'. It comprises the following subfields:
  - event_id;
  - duration;
  - start_time;
  - private_data.
- **flow_id**: this parameter uniquely identifies a stream for a given stream type and a given flow_super_CAS_id;
- **flow_PID**: this parameter is the PID value of a stream. It is left-padded with zeroes to fill 2 bytes;
- **flow_PID_change_related_data**: this field provides the C(P)SIG with the necessary information concerning an ECM, EMM or private data stream whose PID is about to change. It includes the following subfields:
  - flow_type;
  - flow_super_CAS_id;
  - flow_id;
  - flow_PID.
- **flow_super_CAS_id**: this parameter identifies the CA system and the CA subsystem a stream belongs to;
- **flow_type**: this parameter identifies the type of a stream:
  - **0x00:** EMM;
  - **0x01:** private data;
  - **0x02**: ECM;
  - **other values:** DVB reserved.
- **insertion_delay**: this parameter gives the amount of time between the time of a trigger cause and the time at which the descriptors associated to this trigger cause should be inserted. If insertion_delay is negative, the descriptor insertion shall be done before the trigger cause;
- **insertion_delay_type**: this parameter indicates to the (P)SIG timing requirements for transmission of the updated table, with respect to the event start time or ECM modification (as determined by the trigger_type parameter). This parameter can have either of these values:
  - **0x01** = "immediate": the (P)SIG shall immediately insert the set of descriptors in the table;

- **0x02** = "synchronized": the (P)SIG shall synchronize the insertion of the set of descriptors with the cause identified by trigger_id. An insertion_delay parameter indicates the amount of time between this cause and the time of the insertion of the descriptors in the table;
- **other values**: DVB reserved.

- **last_section_indicator**: this parameter is a boolean used in a set of messages carrying table sections to C(P)SIG; when true in a message, it indicates that this message is the last one in the sequence of responses;

- **location_id**: this parameter identifies the table and descriptor loop (if applicable) in which a (P)SIG is to insert a set of descriptors. See table 32 for values;

- **max_comp_time**: this parameter defines the delay for which a trigger precedes the corresponding scheduled action; depending on message context this delay is the one estimated by a (P)SIG or the one wanted by a C(P)SIG;

- **max_streams**: this parameter gives the maximum number of streams supported for a channel by a C(P)SIG or a (P)SIG, depending on message context;

- **MPEG_section**: this parameter includes one and only one MPEG-2 PSI or DVB-SI section;

- **network_id**: this parameter is the value of the network_id field as defined by ETSI EN 300 468 [1];

- **original_network_id**: this parameter is the value of the original_network_id field as defined by ETSI EN 300 468 [1];

- **private_data**: this parameter corresponds to the event related private data. Its contents are private;

- **private_data_specifier**: this parameter contains the value of a private_data_specifier as defined by ETSI EN 300 468 [1] and ETSI ETR 162 [3];

- **(P)SIG_type**: depending on the message context this parameter identifies a (P)SIG as a PSISIG, PSIG or SIG or a C(P)SIG as a CPSISIG, CPSIG or CSIG. See table 29 for values;

- **segment_number:** this field represents the number of the segment of an EIT schedule. Refer to ETSI ETR 211 [4] for the definition of a segment;

- **service_id**: this parameter is the value of the service_id field as defined by ETSI EN 300 468 [1] for a DVB service, or the PAT program_number of an MPEG-2 program; this parameter allows to refer to an MPEG2 program, which is not a DVB service;

- **service_parameters**: this field includes a set of the following subfields:
  - service_id;
  - trigger_list;
  - max_comp_time.

- **start_time:** this parameter contains the start time of the event in Modified Julian Date (MJD) and Universal Time, Coordinated (UTC) formats. This 5-byte field is coded as 40 bits: the 16 LSBs of MJD, followed by six 4-bit Binary Coded Decimal (BCD) digits representing UTC. Example: 93/10/13 12:45:00 is coded as 0xC079124500;

- **stream_change_timestamp**: this field signals the scheduled date and time of a stream change. The format is the same as the one of start-time field;

- **stream_change_type**: this parameter signals the type of stream change being made:
  - **0x01**: service creation;
  - **0x02**: service deletion;
  - **other values**: DVB reserved.

- **table_id**: this parameter is the value of the table_id field as defined by ISO/IEC 13818-1 [7];

- **transaction_id**: this parameter is the unique identifier of a command; depending on message context, this parameter may allow to identify a corresponding response;
- **transport_stream_id**: this parameter is the value of the transport_stream_id as defined by ISO/IEC 13818-1 [7];
- **trigger_id**: this parameter uniquely identifies a trigger occurrence. Such a trigger_id defined in a message may be referred to later on by another message;
- **trigger_list**: this parameter is a 32-bit vector, each bit of which corresponds to a trigger type, according to table 30. A bit set to 1 means that the corresponding trigger type is concerned. Depending on message context this vector gives:
  - either the intrinsic trigger possibilities of a (P)SIG (at channel setup);
  - or the trigger types a C(P)SIG wants to receive (at trigger enable request);
  - or the effective trigger types a (P)SIG can generate according to its intrinsic possibilities and to the C(P)SIG requests.
- **trigger_type**: this parameter identifies the type of trigger. See table 30 for values.

8.3.3 Channel-level messages

This section defines the syntax and semantics of each channel-level message. It refers to the syntax and semantic of parameters given in subclause 8.3.2; however for some parameters according to a particular message context, complementary descriptions can be given to complete or to replace descriptions given in subclause 8.3.2.

8.3.3.1 Channel_setup message: C(P)SIG ← (P)SIG

The channel_setup message is sent by a (P)SIG from the Channel Not Open state, to establish a channel with a C(P)SIG.

Parameter	Number of instances in message
transaction_id	1
custom_CAS_id	1
custom_channel_id	1
(P)SIG_type	1
trigger_list	1
max_streams	1

transaction_id is the unique identifier of an instance of this command, allowing to identify the corresponding response.

(P)SIG_type identifies the (P)SIG as a PSIG, SIG or PSISIG.

trigger_list gives the trigger types the (P)SIG can *actually* generate.

max_streams informs the C(P)SIG as to the maximum number of streams supported by the (P)SIG for this channel.

8.3.3.2 Channel_status message: C(P)SIG ↔ (P)SIG

The channel_status message is used either:

- by a C(P)SIG from the Channel Setting Up state, to indicate successful channel setup;
- by either a C(P)SIG or a (P)SIG from the Channel Open state, to indicate the status of a channel. This message follows a channel_test message issued by the process that shares the channel in question when no error has been detected.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
(P)SIG_type	1
trigger_list	1
max_streams	1
CA_descriptor_insertion_mode	0 to 1

transaction_id: is set to the transaction_id of the message to which channel_status refers. This is the transaction_id of the previous channel_setup or a channel_test message.

(P)SIG_type: identifies the type of the message sender as a PSISIG, PSIG or SIG if (P)SIG or as a CPSISIG, CPSIG or CSIG if C(P)SIG.

trigger_list: lists the kinds of action triggers supported by the (P)SIG for this channel. When channel_status is issued by the C(P)SIG (as a result of a channel_setup or a channel_test issued by the (P)SIG), this parameter shall be ignored by the (P)SIG.

max_streams: is used in one of two ways:

- when channel_status is a response to channel_setup, the C(P)SIG imposes its max_streams limitation on the (P)SIG. This shall be less than or equal to the value of max_streams supplied by the (P)SIG during channel_setup;
- when channel_status is a response to channel_test, max_streams defines the maximum number of streams supported on this channel. This is the value of max_streams returned by the C(P)SIG after channel_setup.

CA_descriptor_insertion_mode: this parameter is provided by the CPSIG/CPSISIG and indicates whether the PSIG/PSISIG has to insert the skeleton of the CA_descriptor; this skeleton corresponds to the CA_descriptor as defined in table 2.51 of ISO/IEC 13818-1 [7] with an empty private_data_byte part; this parameter is mandatory and significant only when channel_status is issued by the CPSIG as response to a channel_setup command, otherwise it shall be ignored or can be omitted; it can have the following values:

- 0x01: the skeleton of the CA_descriptor is always inserted by the PSIG;
- 0x02: no CA_descriptor skeleton is inserted by the PSIG; the CPSIG is always responsible for the generation of this descriptor;
- other values: DVB reserved.

8.3.3.3 Channel_test message: C(P)SIG ↔ (P)SIG

The channel_test message is sent by either a C(P)SIG or a (P)SIG from the channel Open state, to verify that:

- the TCP connection is still alive;
- the channel is in an error-free condition.

The receiver of the channel_test message shall reply with a channel_status message if the channel is free of errors, or a channel_error message if errors occurred.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

8.3.3.4 Channel_close message: C(P)SIG ← (P)SIG

The channel_close message is sent by the (P)SIG to indicate that the channel is to be closed. This message is not acknowledged.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1

transaction_id: is the unique identifier of an instance of this command.

8.3.3.5 Channel_error message: C(P)SIG ↔ (P)SIG

The channel_error message is sent by the recipient of a channel_test message or by the C(P)SIG at any time to indicate that an unrecoverable channel error occurred.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
error_status	1 to n
error_information	0 to n

If this message is generated by the C(P)SIG on unrecoverable error, **transaction_id** is the unique identifier of an instance of this command; if this message is a response to a previous channel_test message, **transaction_id** is set to the transaction_id of this channel_test message.

8.3.4 Stream-level messages

This section defines the syntax and semantics of each stream-level message. It refers to the syntax and semantic of parameters given in subclause 8.3.2; however for some parameters according to a particular message context, complementary descriptions can be given to complete descriptions given in that section.

8.3.4.1 stream_setup message: C(P)SIG ← (P)SIG

The stream_setup message is sent by a (P)SIG from the Stream Not Open state, to establish a new stream with a C(P)SIG. This message carries the values of service_id of all the services broadcast in that transport stream.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
network_id	1
original_network_id	1
transport_stream_id	1
service_id	0 to n

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

The transport stream associated with this **custom_stream_id** is identified by **network_id, original_network_id** and **transport_stream_id**.

Each service (DVB service or MPEG2 program, see service_id parameter description in subclause 8.3.2.2) present in the TS shall be described by a **service_id** parameter. The (P)SIG is required to signal NVOD reference services in this manner, even if it does not support private data insertion synchronized with NVOD reference events. The stream_service_change message is used to signal the addition or deletion of a service.

8.3.4.2 Stream_status message: C(P)SIG ↔ (P)SIG

The stream_status message is used either:

- by a C(P)SIG from the Stream Setting Up state, to indicate successful stream setup;

- by either a C(P)SIG or a (P)SIG from the Stream Open, Stream Trigger Enabling, or Stream Trigger-Enabled state, to indicate the status of a stream. This message follows a stream_test message issued by the process that shares the stream in question.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
service_id	0 to n

transaction_id: is set to the transaction_id of the message to which stream_status refers. This is the transaction_id of the previous stream_setup or a stream_test message.

All **service_id** parameters concerning the TS shall be supplied, in a same way as for the stream_setup message, when stream_status is issued by a (P)SIG. When stream_status is issued by the C(P)SIG (as a result of a stream_setup or a stream_test issued by the (P)SIG), this parameter is not used.

8.3.4.3 Stream_test message: C(P)SIG ↔ (P)SIG

The stream_test message is sent by either a C(P)SIG or a (P)SIG from the Stream Open state, to verify that the stream is in an error-free condition.

The receiver of the stream_test message shall reply with a stream_status message if the stream is free of errors, or a stream_error message if unrecoverable errors have occurred.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

8.3.4.4 Stream_close message: C(P)SIG ← (P)SIG

The stream_close message is sent by the (P)SIG to indicate that the stream is to be closed. This message is not acknowledged by the C(P)SIG.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1

transaction_id: is the unique identifier of an instance of this command.

8.3.4.5 Stream_close_request message: C(P)SIG → (P)SIG

The stream_close_request message is sent by the C(P)SIG to request the (P)SIG to close a stream.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

8.3.4.6 Stream_close_response message: C(P)SIG ← (P)SIG

The stream_close_response message is sent by the (P)SIG in response to a stream_close_request message.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1

transaction_id: is set to the transaction_id of the message to which stream_close_response refers. This is the transaction_id of the previous stream_close_request message.

8.3.4.7 Stream_error message: C(P)SIG ↔ (P)SIG

The stream_error message is sent by the recipient of a stream_test message or by the C(P)SIG at any time to indicate that an unrecoverable stream error had occurred.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
error_status	1 to n
error_information	0 to n

If this message is generated by the C(P)SIG on unrecoverable error, **transaction_id** is the unique identifier of an instance of this command; if this message is a response to a previous stream_test message, **transaction_id** is set to the transaction_id of this stream_test message.

8.3.4.8 Stream_service_change message: C(P)SIG ← (P)SIG

The stream_service_change message is sent by the (P)SIG to signal a modification in service existence in a TS (addition of a service to a TS or deletion of a service from a TS).

More sophisticated functions such as service_id change or the move of a service from one TS to another TS can be achieved by combining these two basic addition and deletion functions.

The head-end (any (P)SIG) shall send a stream_service_change message to all C(P)SIG processes operating in the Stream Open or Stream Trigger-Enabled state.

This message should be issued by the (P)SIG no later than the scheduled service change. The advance notification time for this message is not defined by the present document.

This message is not acknowledged by the C(P)SIG.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
service_id	1
stream_change_type	1
stream_change_timestamp	0 or 1

transaction_id: is the unique identifier of an instance of this message.

8.3.4.9 Stream_trigger_enable_request message: C(P)SIG → (P)SIG

The C(P)SIG sends this message to the (P)SIG for one of these reasons:

Notify the (P)SIG that it is ready to receive action triggers, which ones it wants to receive, and (if possible) how long before the scheduled action it would like to receive the trigger. This happens after stream setup, when the stream state machine is in the stream_open state. Transmission of this message, and its receipt by the (P)SIG, move the stream state machine to the stream_trigger_enabling state. The (P)SIG is required to response with a stream_trigger_enable_response message, whose transmission and receipt move the stream state machine to the stream_trigger-enabled state.

Change the information supplied to the (P)SIG in the most recent stream_trigger_enable_request message. This happens when the stream state machine is in the stream_trigger_enabled state. Transmission of this message and its receipt by the (P)SIG move the stream state machine to the stream_trigger_enabling state. The (P)SIG is required to response with a stream_trigger_enable_response message, whose transmission and receipt move the stream state machine to the stream_trigger_enabled state back.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
service_parameters	0 to n

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

The C(P)SIG may define for any service in the TS the trigger conditions it wants. For such each service, the C(P)SIG has to supply a **service_parameters** field including:

- a **service_id** parameter to identify the service;
- a **trigger_list** parameter to identify by which causes the C(P)SIG wants to be triggered;
- a **max_comp_time** parameter to define how far in advance of a scheduled action the C(P)SIG wants to receive the corresponding trigger.

8.3.4.10 Stream_trigger_enable_response message: C(P)SIG ← (P)SIG

The (P)SIG shall send this message in reply to a stream_trigger_enable_request message from the C(P)SIG. This message describes the types action triggers it can actually send to the C(P)SIG, on a per-service basis.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
service_parameters	0 to n

transaction_id: is set to the transaction_id of the stream_trigger_enable_request message to which this message refers.

For each service specified by the C(P)SIG in the preceding stream_trigger_enable_request message, the (P)SIG has to supply a **service_parameters** field including:

- a **service_id** parameter to identify the service;
- a **trigger_list** parameter to identify for which causes the (P)SIG can generate triggers. This trigger_list is the logical AND of the following:
    - the trigger_list from the preceding stream_trigger_enable_request message, which represents the triggers that the C(P)SIG would like to receive;
    - a similar vector of the (P)SIG's capabilities for the current channel. This latter vector is supplied as the trigger_list in the channel_setup message;
    - a logical vector where '0' bits represent the impossibility of generating certain trigger types for certain kinds of services.
- a **max_comp_time** parameter to define how far in advance of a scheduled action the (P)SIG might possibly, given sufficient advance notification, send the corresponding trigger. The actual advance notification is estimated by the (P)SIG; the present document does not provide any mechanisms to guarantee the accuracy of this value.

8.3.4.11 Trigger message: C(P)SIG ← (P)SIG

The trigger message is sent by the (P)SIG to the C(P)SIGs under any of the circumstances, identified by the value for the trigger_type parameter: it is generated according to the trigger types allowed in the relevant stream_trigger_enable_response message.

This message is not acknowledged.

The content of the trigger message depends on the trigger_type value, as shown in the following table:

Parameter	Number of instances in message Following event	Number of instances in message Future event	Number of instances in message ECM related event (all cases)	Number of instances in message PID change
transaction_id	1	1	1	1
custom_channel_id	1	1	1	1
custom_stream_id	1	1	1	1
service_id	1	1	1	0/1
trigger_id	1	1	1	1
trigger_type	1	1	1	1
event_related_data	1	1	0	0
ECM_related_data	0 to N	0	1	0
flow_PID_change_related_data	0	0	0	1

When the trigger message pertains to following event, it shall contain as many ECM_related_data fields as there are ECM stream references (i.e. CA-descriptors) pertaining to the relevant CAS in the whole service description (i.e. in PMT).

In a PID change trigger for an EMM stream, the service-id parameter shall be omitted; for other stream types, this parameter shall be included.

transaction_id: is the unique identifier of an instance of this message.

trigger_id: uniquely identifies this trigger occurrence: this trigger_id may be referred to later on by the descriptor_insert_request message.

8.3.4.12 Table_request message: C(P)SIG → (P)SIG

The table_request message is sent by the C(P)SIG to request MPEG-2 PSI or DVB SI table data from the (P)SIG. The value of the custom_stream_id parameter identifies the transport_stream on which the table is broadcast.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
table_id	1
network_id	0 or 1
transport_stream_id	0 or 1
original_network_id	0 or 1
service_id	0 or 1
bouquet_id	0 or 1
event_id	0 or 1
segment_number	0 or 1

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

table_id: is the value of the table_id field as defined by ISO/IEC 13818-1 [7]. It identifies the subtable requested by the CPSIG.

The C(P)SIG requests either complete PSI tables or SI sub-tables (see ETSI EN 300 468 [1]). As EIT Schedule sub-tables may be very large, the protocol optionally allows the C(P)SIG to request only the part of the EIT Schedule sub-table that concerns a specific event or a specific segment. In case the request concerns a specific event_id, the value of the table_id field shall be arbitrarily set to 0x50 for the events pertaining to the current TS, and 0x60 for the events pertaining to an EIT other.

Table 31 indicates which parameters (**network_id**, **original_network_id**, **transport_stream_id**, **service_id**, **bouquet_id**, **event_id**, **segment_number**) shall be present in the message, depending on the type of PSI/SI table data requested by the C(P)SIG. According to this table, the transport_stream_id and original_network_id parameters identifying the transport stream carrying the requested table are implicitly included in the custom_stream_id parameter.

8.3.4.13 Table_response message: C(P)SIG ← (P)SIG

The table_response message is sent by the (P)SIG as a response to the table_request message.

If the table does not fit into a single table_response message, the (P)SIG shall split the requested sub-table into several table_response messages with the same **transaction_id**. In every case the last table_response message for a table is indicated with the **last_section_indicator** set to "true".

When this message is a response to a request for EIT data pertaining to a single event, the sections that are returned with this message are limited to the sections containing data pertaining to that event. Likewise, when this message is a response to a request for EIT data pertaining to a single segment, the sections that are returned with this message are limited to the sections containing data pertaining to that segment, i.e. 8 sections are returned.

A single instance of a table_response may carry several **MPEG_sections**.

If there is no **MPEG_section**, this means that the table that is requested is not generated by the (P)SIG.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
last_section_indicator	1
MPEG_section	0 to n

transaction_id: is set to the transaction_id of the table_request to which this message refers.

8.3.4.14 Descriptor_insert_request message: C(P)SIG → (P)SIG

The descriptor_insert_request message is sent by the C(P)SIG. It indicates to the (P)SIG the descriptors that the C(P)SIG wants to be inserted in the PSI/SI tables.

The descriptors in this message cancel and replace all the existing descriptors in the same location, uniquely associated to the same custom_CAS_id and for the same private_data_specifier if given in the message. In particular, using an empty list removes all descriptors of the custom_CAS_id in that location.

This message carries all the information that the (P)SIG needs to know where and when to insert the descriptors. The table below describes the parameters that need to be present in the message, depending on the PSI/SI table which is addressed.

Depending on the needs of the C(P)SIG, this message can be sent asynchronously or as a consequence of a previous trigger message. In the latter case, it contains the trigger_id parameter.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
trigger_id	0 or 1
insertion_delay_type	1
insertion_delay	0 or 1
location_id	1
bouquet_id	0 or 1 (see table 32)
network_id	0 or 1 (see table 32)
original_network_id	0 or 1 (see table 32)
transport_stream_id	0 or 1 (see table 32)
service_id	0 or 1 (see table 32)
event_id	0 or 1 (see table 32)
ES_id	0 or 1 (see table 32)
private_data_specifier	0 or 1
descriptor	0 to n

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

custom_stream_id: identifies the transport stream on which the table that has to be updated, is broadcast.

trigger_id: refers to a previous trigger message. This field is optional. When not in the message the request for insertion is asynchronous.

The value of **insertion_delay_type** can be "synchronized" only when the descriptor_insert_request message is related to a trigger message, via the trigger_id value.

private_data_specifier: This parameter contains the value of a private_data_specifier (see ETSI EN 300 468 [1] and ETSI ETR 162 [3]); when this optional parameter is supplied, one of two cases applies:

- if the (P)SIG is able to generate a private_data_specifier_descriptor, the (P)SIG shall insert the following set of descriptors in the scope of a private_data_specifier_descriptor it generates with the value given by the private_data_specifier parameter;

- if the (P)SIG cannot generate a private_data_specifier_descriptor, the private_data_specifier parameter is ignored.

Table 32 indicates, for each MPEG-2 PSI and DVB SI table, which parameters (**location_id**, **bouquet_id, network_id, original_network_id, transport_stream_id, service_id, event_id, ES_id)** are needed to define where the descriptor given in the **descriptor** parameter shall be inserted. According to this table, the transport_stream_id and original_network_id parameters identifying the transport stream carrying the targetted table are implicitly included in the custom_stream_id parameter.

8.3.4.15 Descriptor_insert_response message: C(P)SIG ← (P)SIG

The (P)SIG shall send this message in reply to a descriptor_insert_request message from the C(P)SIG. This message describes the status of the descriptor insertion.

This message can be sent immediately after the reception of the corresponding request, or after the actual insertion has been attempted or completed.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
descriptor_insert_status	1

transaction_id: is set to the transaction_id of the descriptor_insert_request message to which this message refers.

8.3.4.16 PID_provision_request message: C(P)SIG → (P)SIG

The PID_provision_request message is sent by the C(P)SIG to receive from the (P)SIG the current PID value of a stream identified by the combination {flow_type, flow_super_CAS_id, flow_id}.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
flow_type	1
flow_super_CAS_id	1
flow_id	1

transaction_id: is the unique identifier of an instance of this command, allowing to identify the corresponding response.

8.3.4.17 PID_provision_response message: C(P)SIG ← (P)SIG

The (P)SIG shall send this message in reply to a PID_provision_request message from the C(P)SIG. This message gives the current PID value of a flow identified by the combination {flow_type, flow_super_CAS_id, flow_id}.

Parameter	Number of instances in message
transaction_id	1
custom_channel_id	1
custom_stream_id	1
flow_type	1
flow_super_CAS_id	1
flow_id	1
flow_PID	1

transaction_id: is set to the transaction_id of the PID_provision_request message to which this message refers.

8.3.5 Error status and error information

NOTE: TCP connection level errors are beyond the scope of the present document. Only channel, stream and application level errors are dealt with. These errors occur during the lifetime of a TCP connection.

Table 35: C(P)SIG connection-oriented protocol error values

error_status value	Error type
0x0000	DVB Reserved
0x0001	invalid message
0x0002	unsupported protocol version
0x0003	uUnknown message_type value
0x0004	message too long
0x0005	unknown custom_stream_id value
0x0006	unknown custom_channel_id value
0x0007	too many channels on this C(P)SIG
0x0008	too many data streams on this channel
0x0009	too many data streams on this C(P)SIG
0x000A	unknown parameter_type
0x000B	unknown transaction_id value in response message
0x000C	not compliant C(P)SIG- and (P)SIG-types
0x000D	invalid value for DVB parameter
0x000E	unknown custom_CAS_id value
0x000F	uUnknown bouquet_id value
0x0010	invalid CA_descriptor_insertion_mode value
0x0011	invalid descriptor_insert_status
0x0012	inconsistent ES_id value
0x0013	unknown event_id value
0x0014	uUnknown flow_id value
0x0015	uUnknown flow_super_CAS_id
0x0016	nvalid flow_type
0x0017	nvalid insertion_delay_type value
0x0018	nvalid location_id value
0x0019	unknown network_id value
0x001A	unknown original_network_id value
0x001B	invalid (P)SIG or C(P)SIG value
0x001C	invalid stream_change_type value
0x001D	invalid table_id value
0x001E	unknown transport_stream_id value
0x001F	invalid trigger_type value
0x0020	unknown service_id
0x0021	missing_service_id
0x0022	inconsistent length for DVB parameter
0x0023	missing mandatory DVB parameter
0x0024	invalid PID value
0x0025	unexpected trigger message
0x0026	unknown segment_number
0x0027	unknown trigger_id value
0x0028	unknown private_data_specifier value
0x0029	unknown descriptor_insert_status value
0x002A	custom_channel_id value already in use
0x002B	custom_stream_id value already in use
0x002C	flow_id value already in use
0x002D to 0x6FFF	DVB Reserved
0x7000	unknown error
0x7001	unrecoverable error
0x7002 to 0x7FFF	DVB Reserved
0x8000 to 0xFFFF	head-end specific / CA system specific / User defined

8.4 SIMF-based protocol

This section specifies the SIMF-based implementation of the C(P)SIG ⇔ (P)SIG interface. This implementation is based on the (P)SI Object Information Group of the Simulcrypt Identification Module (SIM) and the Operations Reference Point (ORP) operations paradigm that can be used between any components within Simulcrypt (see clause 7):

- this operations paradigm is introduced in subclause 8.4.1;
- its application to the C(P)SIG ⇔ (P)SIG interface and the conceptualization of the (P)SI object information group are introduced in subclause 8.4.2;

- the (P)SI object information group is defined in subclause 8.4.2.4.

8.4.1 Operations Reference Points (ORPs)

The information exchange and synchronization between any number components is accomplished through the SIMF as follows:

- the component advertises information such as SI/PSI information through a Management Information Base (examples SNMPv2 SMI MIB); that is done through a software agent (e.g. SNMPv2 agent) or through an object server (e.g. CORBA object server or RMI object server) or a web server;
- the component implements the SIMF events and logs modules which enables the asynchronous notification capability of the management system;
- the CAS Manager reads the Head-end/Uplink MIBs on a periodic basis, updates its custom information and registers itself as a recipient of events caused by updates in the Head-end/Uplink MIB;
- the CAS Manager updates the Head-end/Uplink with CA information on a periodic basis or upon receiving an event from the Head-end/Uplink or another CAS component through one of the following means:
    - sets the Uplink MIBs through the network management protocol;
    - updates its own MIBs, which causes events that result in notifications to be sent to the Head-end/Uplink manager.

Figure 16 illustrates this synchronization and information exchange mechanism within the Uplink/Head-end.

An Operation Reference Point (ORP) defines the interface between two Head-end/Uplink components. It replaces a custom communications protocol, which would need to be designed for that particular interface by a generic information exchange and event notification mechanism.

The four basic ingredients, which enable ORP, based information exchange and synchronization are:

- information access through a standardized management protocol such as SNMP or CMIP or through a standardized OO protocol such as CORBA IIOP or Java RMI;
- information advertisement through a management information base or an object broker;
- event trigger configuration, event forwarding configuration, and event notifications;
- event logging.

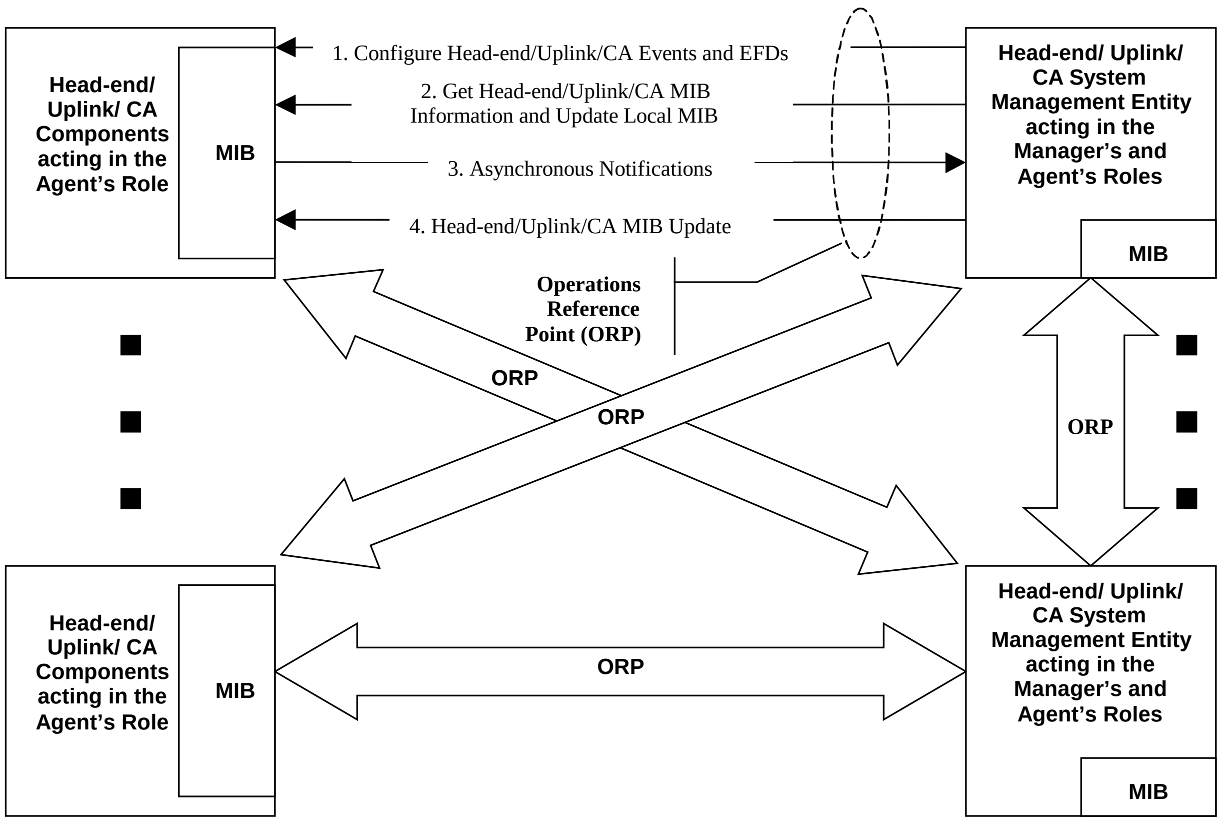

Head-end/ Uplink/ CA Components acting in the Agent's Role
MIB
1. Configure Head-end/Uplink/CA Events and EFDs
2. Get Head-end/Uplink/CA MIB Information and Update Local MIB
3. Asynchronous Notifications
4. Head-end/Uplink/CA MIB Update
Head-end/ Uplink/ CA System Management Entity acting in the Manager's and Agent's Roles
MIB
Operations Reference Point (ORP)
ORP
ORP
ORP
Head-end/ Uplink/ CA Components acting in the Agent's Role
MIB
ORP
Head-end/ Uplink/ CA System Management Entity acting in the Manager's and Agent's Roles
MIB

Figure 16: Operations Reference Points

8.4.2 Application of ORPs to the C(P)SIG ↔ (P)SIG Interface

Information exchange and synchronization between C(P)SI and (P)SI generators is accomplished as follows using ORPs:

- the (P)SIG implements three SNMPv2 SMI modules:
  - events module as specified in Common Information Modules Section;
  - logs module as specified in Common Information Modules Section;
  - (P)SI/C(P)SI information within the (P)SIG which is defined in a group consisting of 9 tables as follows:
    - (P)SIG Information - this table is used to identify the different (P)SIG communications profiles; there is one entry in this table for each transport stream supported by the (P)SIG; if different (P)SIG configurations are supported for the same transport stream, there is one entry in this table for each of these configurations;
    - (P)SIG Configuration - this table is used by the C(P)SIG to configure the (P)SIG with information defining triggering parameters for ECM, EMM, and Event triggers;
    - ECM Trigger Table - this table contains all currently active ECM triggers;
    - Flow PID Change Trigger Table - this table contains all currently active Flow PID change triggers;
    - Event Trigger Table - this table contains all currently active event triggers;
    - Descriptor Insert Table - this is the table used by the C(P)SIG to communicate descriptor insertion information to the (P)SIG;
    - Descriptor Insert Descriptor Table - this is a sub-table of the Descriptor Insert Table containing the actual descriptors to be inserted;
    - Table Provisioning Request Table - this is the table used by the C(P)SIG to request table provisioning by the (P)SIG;

- PID Provisioning Table - this is the table used by the C(P)SIG to request Flow PID provisioning by the (P)SIG.

The five transaction types across the C(P)SIG ⇔ (P)SIG ORP are:

1) ECM/Event/Flow Change Triggering - An event occurs within the (P)SIG which causes asynchronous information (e.g. (P)SI tables) to be communicated to the C(P)SIG and which may be followed up by the C(P)SIG inserting descriptors into the (P)SIG's (P)SI information;

2) (P)SI Table Provisioning - The C(P)SIG requests and obtains (P)SI information from the (P)SIG;

3) (P)SI Descriptor Insertion - The C(P)SIG requests that a descriptor be inserted into the (P)SIG's (P)SI tables;

4) Transport Stream Service Changes - A transport stream service change is signalled by the (P)SIG by changing a variable in its static configuration table. All interested C(P)SIGs can receive value change notifications by configuring the (P)SIGs EFD table;

5) PID Provisioning - The C(P)SIG requests and gets the current PID value of a particular stream identified by its unique identifier.

8.4.2.1 ECM/Event/Flow Change Triggering

The (P)SI/C(P)SI ORP facilitates ECM/Event/Flow Change Triggering as follows:

1) the (P)SIG's Event Table is configured from start-up to generate events whenever there is a new entry in any of the three Trigger tables (i.e. the ECM, and Event Trigger tables); for each type of Trigger there is a different type of event;

2) the (P)SIG advertises which types of triggers it supports through a SIM variable;

3) the C(P)SIG configures the (P)SIG's Configuration Table with triggering parameters such as the types of triggers it wishes to receive, delay, etc;

4) the C(P)SIG configures the (P)SIG's Event Forwarding Discriminator (EFD) table with event forwarding information such as IP address, notification type (confirmed/unconfirmed), filtering (i.e. CAS_id);

5) the (P)SIG generates a trigger and populates the Trigger table as configured;

6) the generation of a trigger causes an event to be generated, which is forwarded as defined in the EFD table;

7) the C(P)SIG receives the event and reads the trigger from the (P)SIG.

If the trigger causes the C(P)SIG to insert descriptors the C(P)SIG writes descriptors into the (P)SIG's descriptor insert table.

8.4.2.2 (P)SI Table Provisioning

The C(P)SIG requests (P)SI through the (P)SIG's Provisioning table. This table is just an interface to the (P)SI information within the (P)SIG or a proxy function to such information. The reply to the provisioning request contains the desired table or a part of the desired table if the table is too large to be sent in one reply. If the part number returned is 0 the table part returned is the last or only table part. Otherwise, it is the sequence number of the next table part that should be retrieved by the C(P)SIG and the C(P)SIG has to continue requesting table parts until it receives one with the sequence number 0.

8.4.2.3 (P)SI Descriptor Insertion

The C(P)SIG requests (P)SI descriptor insertion through the (P)SIG's Descriptor Insert table. That table may also be just an interface to (P)SIG's descriptor insert application. The reply to the descriptor insert request informs the C(P)SIG whether the descriptor insertion was successful.

8.4.2.4 Transport Stream Service Changes

The (P)SIG announces changes in services on the transport stream by changing a variable in the (P)SIG Configuration Table that contains the service identifiers of all services on the transport stream.

Each C(P)SIG configures the (P)SIG's event and EFD tables so that it will receive a notification if the (P)SIG transport stream service list variable changes.

8.4.2.5 PID Provisioning

The C(P)SIG requests from the (P)SIG the PID value assigned by the head-end to a stream (ECM, EMM or private data) through the PID Provisioning table. That table may be also just an interface to (P)SIG's database. The stream is identified by its type, the identifier of the CA system and the CA sub-system the stream belongs to, and a unique stream number. The reply to this request contains the PID value.

8.4.3 SIM (P)SIG Group Specification

This group consists of nine tables:

- (P)SIG Information Table;
- (P)SIG Configuration Table;
- ECM Trigger Table;
- Event Trigger Table;
- Flow PID Change Trigger Table;
- Descriptor Insert Table,
- Descriptor Insert Descriptor Table;
- Table Request Table;
- PID Provisioning Table.

8.4.3.1 Information Table

The first table is the (P)SIG information table. It is used by the (P)SIG to advertise its configuration, the transport streams it handles, and the services contained in all the transport streams. The table is indexed by simPsigIndex. The information consists of the following.

Table 36: CIM - SIM (P)SIG Group - (P)SIG Information Table

Object	Size/Description	Object Justification	Head-end/ CAS Manager Maximum Access Right
simPsigIndex	2 uimsbf/unique index of the (P)SIG Table	identifies the (P)SIG communications profile	read
simPsigType	1 uimsbf/(P)SIG type definition	identifies whether the (P)SIG is a PSIG, a SIG or a PSISIG	read
simPsigTriggerSupport	4 uimsbf/trigger type definition	identifies which trigger types the (P)SIG supports	read
simPsigNetworkId	2 uimsbf/the network identifier (see ETSI EN 300 468 [1])	identifies the network	read
simPsigONetworkId	2 uimsbf/the original network identifier (see ETSI EN 300 468 [1])	identifies the original network	read
simPsigTransStreamId	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream	read
simPsigTSServices	bslbf/list of services identifiers	identifies all services on the transport stream	read

8.4.3.2 Configuration Table

The second table is the configuration table. It is used to configure (P)SIG/C(P)SIG interaction. Each table row access is controlled by the administrative state variable and the row status variables as defined in the corresponding ITU-T and IETF standards. The table is indexed by the simPsigConfigCustCasId, the simPsigConfigIndex and the simPsigIndex. Each C(P)SIG enters one or more entries (rows) into this table. Each row defines one (P)SIG/C(P)SIG communications profile.

Each communications profile is identified by the unique index. There may be multiple entries for the same Custom Conditional Access System Identifier (simPsigConfigCustCasId). Each entry defines the types of triggers that the C(P)SIG enables and the amount of time the C(P)SIG needs after receiving a trigger to insert descriptors into the (P)SIG's table (simPsigConfigMaxCompTime). The enabling is also characterized by any combination of the following parameters: service identifier and CA_descriptor insertion mode.

Table 37 summarizes this information.

Table 37: CIM - SIM (P)SIG Group - Configuration Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigConfigIndex	2 uimsbf/unique index of the (P)SIG Configuration Table	identifies the C(P)SIG communications profile.	read-create
simPsigIndex	2 uimsbf/unique index of the (P)SIG Table	identifies the (P)SIG communications profile	read-create
simPsigConfigAdminState	enumerated/administrative state of the table row (as in ITU-T Recommendation X.731 [8])	enables locking for synchronized access of multiple head-end or CAS network managers	read-create
simPsigConfigCpsigType	1 uimsbf/C(P)SIG type definition	identifies whether the C(P)SIG is a CPSIG, a CSIG or CPSISIG	read-create
simPsigConfigCustCasId	4 uimsbf/Custom CAS Identifier	enables identification of C(P)SIGs	read-create
simPsigConfigMaxCompTime	2 uimsbf/maximum time needed to process trigger by C(P)SIG	enables (P)SIG to estimate the time between its triggering and descriptor insertion by the C(P)SIG	read-create
simPsigConfigServiceId	2 uimsbf/service identifier (see ETSI EN 300 468 [1])	identifies the service	read-create
simPsigConfigTriggerEnable	2 uimsbf/trigger type definition	identifies which trigger types the C(P)SIG wants	read-create
simPsigConfigCADInsMode	enumerated/defines CA_descriptor insertion mode	identifies whether the (P)SIG has to insert the skeleton of the CA_descriptor	read-create
simPsigConfigEntryStatus	enumerated/row creation status as defined in the RFC 1901 [15] to RFC 1908 [22]	enables row creation control	read-create

8.4.3.3 ECM Trigger Table

The third table is the ECM Trigger table. It is used for the (P)SIG to enter ECM Triggers and for the C(P)SIG to read them. Upon entering a new entry (ECM Trigger) into this table an event is automatically generated by the (P)SIG as specified in the Events Configuration Table of the Simulcrypt Events Module (SEM). This event is an object value change event and carries the instance identifier of ECM Trigger Table Index as changed object identifier and the Custom CAS identifier as specific problems. This allows EFD filtering in the SEM on the Custom CAS identifier and communicates the event type (through the event name) and the ECM Trigger index to the C(P)SIG through the EFD and through a value change notification. The value change notification can be either confirmed or unconfirmed.

Each trigger is identified by the unique index (simPsigEcmTrIndex) and announces a single ECM stream creation or ECM stream deletion. The ECM stream created/deleted can be DVB Service wide or just component wide. If it is component wide a head-end wide Elementary Stream identifier is included in the table.

There may be multiple table rows per trigger depending on whether there are multiple components attached or not. Each table row is uniquely indexed by the simPsigEcmTrIndex.

Each table row consists of a network identifier (simPsigEcmTrNetworkId), the original network identifier (simPsigEcmTrONetworkId), the transport stream identifier (simPsigEcmTrTransStreamId), the service identifier (simPsigEcmTrServiceId), possibly the elementary stream identifier if the ECM stream is to be component level only (simPsigEcmTrEsId), the ECM trigger type which could be ECM stream open, close or access_criteria change (simPsigEcmTrType), the Super CAS Identifier (simPsigEcmTrSuCasId), the ECM unique identifier (simPsigEcmTrEcmId), the ECM PID (simPsigEcmTrEcmPid), the access criteria (simPsigEcmTrAccessCriteria).

Table 38 summarizes this information.

Table 38: CIM - SIM (P)SIG Group - ECM Trigger Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigEcmTrIndex	2 uimsbf/unique identifier of the ECM Trigger	identifies the (P)SIG ECM Trigger	read-create
simPsigEcmTrNetworkId	2 uimsbf/the network identifier (see ETSI EN 300 468 [1])	identifies the network	read
simPsigEcmTrONetworkId	2 uimsbf/the original network identifier (see ETSI EN 300 468 [1])	identifies the original network	read
simPsigEcmTrTransStreamId	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream	read
simPsigEcmTrServiceId	2 uimsbf/ the service identifier (see ETSI EN 300 468 [1])	identifies the service	read
simPsigEcmTrEsId	2 uimsbf/ the elementary stream identifier	identifies the elementary stream if the ECMs are to be applied component wide only and not service wide	read
simPsigEcmTrType	enumerated/identifies the trigger type	distinguishes between ECM stream open, close, and access criteria change triggers	read
simPsigEcmTrSuCasId	4 uimsbf/the Super CAS identifier	identifies the CAS	read
simPsigEcmTrEcmId	2 uimsbf/the ECM stream identifier	identifies uniquely the ECM stream	read
simPsigEcmTrEcmPid	2 uimsbf/the ECM PID	identifies the ECM PID	read
simPsigEcmTrAccessCriteria	bslbf/the access criteria	specifies the access criteria	read

8.4.3.4 Flow PID Change Trigger Table

The fourth table is the Flow PID Change Trigger table. It is used for the (P)SIG to enter Flow PID Change Triggers and for the C(P)SIG to read them. Upon entering a new entry (Flow PID ChangeTrigger) into this table an event is automatically generated by the (P)SIG as specified in the Events Configuration Table of the Simulcrypt Events Module (SEM). This event is an object value change event and carries the instance identifier of Flow PID Change Trigger Table Index as changed object identifier and the Custom CAS identifier as specific problems. This allows EFD filtering in the SEM on the Custom CAS identifier and communicates the event type (through the event name) and the Flow PID Change Trigger index to the C(P)SIG through the EFD and through a value change notification. The value change notification can be either confirmed or unconfirmed.

Each trigger is identified by the unique index (simPsigFlowTrIndex) and announces a single Flow PID change. Each table row consists of the trigger index (simPsigFlowTrIndex), the flow type (simPsigFlowTrType), the super CAS identifier (simPsigFlowTrSuCasId), the flow identifier (simPsigFlowTrFlowId), and the flow PID (simPsigFlowTrFlowPID). The table is indexed by the trigger index.

Table 39 summarizes this information.

Table 39: CIM - SIM (P)SIG Group - Flow PID Change Trigger Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigFlowTrIndex	2 uimsbf/unique identifier of the Flow Trigger	identifies the (P)SIG Flow Trigger	read-create
simPsigFlowTrType	enumerated/specifies the flow type	the flow type can be ECM, EMM or private data	read
simPsigFlowTrSuCasId	4 uimsbf/the Super CAS identifier	identifies the CAS	read
simPsigFlowTrFlowId	2 uimsbf/flow identifier	uniquely identifies the flow for a given flow type and CAS identifier	read
simPsigFlowTrFlowPID	2 uimsbf/flow PID	identifies the flow PID	read

8.4.3.5 Event Trigger Table

The fifth table is the Event Trigger table. It is used for the (P)SIG to enter Event Triggers and for the C(P)SIG to read them. Upon entering a new entry (Event Trigger) into this table an event is automatically generated by the (P)SIG as specified in the Events Configuration Table of the Simulcrypt Events Module (SEM). This event is an object value change event and carries the instance identifier of the Event Trigger Table Index as changed object identifier and the Custom CAS identifier as specific problems. This allows EFD filtering in the SEM on the Custom CAS identifier and communicates the event type (through the event name) and the Event Trigger index to the C(P)SIG through the EFD and through a value change notification. The value change notification can be either confirmed or unconfirmed.

The Event Trigger table is indexed by the unique Event Trigger Index (simPsigEvntTrIndex).

Each trigger is identified by the unique index (simPsigEvntTrIndex). Each table row consists of a network identifier (simPsigEvntTrNetworkId), the original network identifier (simPsigEvntTrONetworkId), the transport stream identifier (simPsigEvntTrTransStreamId), the identifier of the service (simPsigEvntTrServiceId), the event identifier (simPsigEvntTrEventId), the start time (simPsigEvntTrStartTime), the duration of the event (simPsigEvntTrDuration), and the event related private data (simPsigEvntTrPrivateData). Table 40 summarizes this information.

Table 40: CIM - SIM (P)SIG Group - Event Trigger Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigEvntTrIndex	2 uimsbf/unique index of the (P)SIG EVENT TriggerTable	identifies the (P)SIG Event Trigger	read
simPsigEvntTrNetworkId	2 uimsbf/the network identifier (see ETSI EN 300 468 [1])	identifies the network	read
simPsigEvntTrONetworkId	2 uimsbf/the original network identifier (see ETSI EN 300 468 [1])	identifies the original network	read
simPsigEvntTrTransStreamId	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream	read
simPsigEvntTrServiceId	2 uimsbf/service identifier (see ETSI EN 300 468 [1])	identifies the service	read
simPsigEvntTrEventId	2 uimsbf/event identifier (see ETSI EN 300 468 [1])	identifies the event	read
simPsigEvntTrStartTime	5 bslbf/the start time in Modified Julian Date (MJD) and Universal Time, Coordinated (UTC) formats. The Field is coded as 16 bits of LSBs of MJD, followed by six 4-bit (BCD) digits representing UTC.	Indicates the start time	read
simPsigEvntTrDuration	3 uimsbf/the event duration in hours, minutes, and seconds as six 4-bit BCD digits	indicates the duration of the event	read
simPsigEvntTrPrivateData	bslbf/variable length event related user private data	event related user private data	read

8.4.3.6 Descriptor Insert Table

The sixth table is the Descriptor Insert table. It is used for the C(P)SIG to enter (P)SI descriptors and for the (P)SIG to read them. Entering of a descriptor completes the cycle which was started by EMM/ECM/Event triggering. The actual descriptors to be inserted are written into the descriptor table. The descriptor insert status is communicated back in the descriptor insert reply message (i.e. each set is confirmed by a get-response in SNMP) or as a value change notification (i.e. the C(P)SIG configures events and EFDs to monitor the status object identifier instance of the descriptor table entry). Each table row access is controlled by the administrative state variable and the row status variables as defined in the corresponding ITU-T and IETF standards.

Each descriptor is identified by the unique index (simPsigDescInsIndex). Each table row consists of the index and type of the trigger that caused the descriptor insertion (simPsigDescInsTrIndex, simPsigDescInsTrType), of a descriptor insert location designator (simPsigDescInsLocationId), of a number of parameters specifying the descriptor, and also the descriptor insertion delay (simPsigDescInsDelay) and the descriptor insertion delay type (simPsigDescInsDelayType).

Table 41 summarizes this information.

Table 41: CIM - SIM (P)SIG Group - Descriptor Insert Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigDescInsIndex	2 uimsbf/unique index of the (P)SIG Descriptor Insert Table	identifies the (P)SIG Descriptor Insert	read-create
simPsigDescInsAdminState	enumerated/administrative state of the table row (as in ITU-T Recommendation X.731 [8])	enables locking for synchronized access of multiple head-end or CAS network managers	read-create
simPsigDescInsTrIndex	2 uimsbf/unique index of the (P)SIG Trigger Table	identifies the (P)SIG Trigger associated with this descriptor insertion	read-create
simPsigDescInsTrType	enumerated/identifies the type of the trigger	indicates whether the trigger index is an ECM, or Event index	read-create
simPsigDescInsLocationId	enumerated/location identifier of the descriptor destination	identifies the table and position where the descriptor is to be inserted	read-create
simPsigDescInsNetworkId	2 uimsbf/the network identifier (see ETSI EN 300 468 [1])	identifies the network	read-create
simPsigDescInsONetworkId	2 uimsbf/the original network identifier (see ETSI EN 300 468 [1])	identifies the original network	read-create
simPsigDescInsTransStreamId	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream	read-create
simPsigDescInsServiceId	2 uimsbf/service identifier (see ETSI EN 300 468 [1])	identifies the service	read-create
simPsigDescInsElmStreamId	2 uimsbf/elementary stream identifier (see ETSI EN 300 468 [1])	identifies the elementary stream	read-create
simPsigDescInsBouquetId	2 uimsbf/bouquet identifier (see ETSI EN 300 468 [1])	identifies the bouquet	read-create
simPsigDescInsEventId	2 uimsbf/event identifier (see ETSI EN 300 468 [1])	identifies the event	read-create
simPsigDescInsONetworkId2loop	2 uimsbf/original network identifier (see ETSI EN 300 468 [1])	identifies the original network in the 2nd loop	read-create
simPsigDescInsNetworkIdOther	2 uimsbf/network identifier (see ETSI EN 300 468 [1])	identifies the other network	read-create
simPsigDescInsTransStreamId2OrO	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream in 2nd loop or other	read-create
simPsigDescInsDelayType	enumerated/immediate or synchronized	indicates whether the (P)SIG shall immediately insert the set of descriptors in the table or shall synchronize the insertion with a triggered event start or ECM stream modification	read-create

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigDescInsDelay	tcimsbf/ms	if the delay insertion type is synchronized this indicates the amount of time between the time the event is supposed to occur and the time at which the descriptors are to be inserted; a negative time indicates that the tables should be broadcast before the ECM stream modification or event start	read-create
simPsigDescPrivDataSpfier	2uimsbf (see ETSI EN 300 468 [1] and ETSI ETR 162 [3])	Private data specifier which is inserted into the TS by (P)SIG if possible.	read-create
simPsigDescInsEntryStatus	enumerated/row creation status as defined in the RFC 1901 [15] to RFC 1908 [22]	enables row creation control	read-create

8.4.3.7 Descriptor Insert Descriptor Table

The seventh table is an extension to the descriptor insert table. It consists of a list of descriptors (simPsigDescInsDescriptor) to be inserted and their insertion return status (simPsigDescInsDescriptorStatus). The table is indexed by the index of the descriptor insert table and by its own unique index. This allows multiple descriptors to be associated with the same insertion and thus also with the same trigger. Each table row access is controlled by the administrative state variable and the row status variables as defined in the corresponding ITU-T and IETF standards.

Table 42 summarizes this information:

Table 42: CIM - SIM (P)SIG Group - Descriptor Insert Descriptor Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigDescInsDescIndex	2 uimsbf/unique index of the (P)SIG Descriptor Insert Descriptor Table	identifies the (P)SIG Descriptor Insert Descriptor	read-create
simPsigDescInsDescAdminState	enumerated/administrative state of the table row (as in ITU-T Recommendation X.731 [8])	enables locking for synchronized access of multiple head-end or CAS network managers	read-create
simPsigDescInsDescriptor	bslbf/the descriptor	the descriptor to be inserted	read-create
simPsigDescInsDescriptorStatus	enumerated/the descriptor insertion status	indicates whether the descriptor has been inserted or not	read-create
simPsigDescInsDescEntryStatus	enumerated/row creation status as defined in the RFC 1901 [15] to RFC 1908 [22]	enables row creation control	read-create

8.4.3.8 Table Request Table

The eighth table is the Table Request table. It is used for the C(P)SIG to request and retrieve (P)SI tables from the (P)SIG. The table is used as an interface to the (P)SI database. Since tables retrieved may exceed the maximum message size supported by the underlying transport (i.e. SNMP) a mechanism is provided to support arbitrary length tables. The reply to the provisioning request contains the desired table or a part of the desired table if the table is too large to be sent in one reply. If the part number returned is 0 this table part returned is the last or only table part. Otherwise, it is the sequence number of the next table part that should be retrieved by the C(P)SIG and the C(P)SIG has to continue requesting table parts until it receives one with the sequence number 0.

Each provisioning request is identified by the unique index (simPsigTblProvIndex). Each table row consists of a table identifier (simPsigTblProvTableId), of a network identifier (simPsigTblNetworkId), of a original network identifier (simPsigTblONetworkId), the transport stream identifier (simPsigTblTransStreamId), the service identifier (simPsigTblServiceId), the bouquet identifier (simPsigTblBouquetId), the event identifier (simPsigTblEventId), the segment number (simPsigTblSegmentNr), the table part requested (simPsigTblProvPart), and the table part number (simPsigTblProvPartNumber). The initial request always has a part number 0. Each subsequent request of the same table has the part number requested.

Table 43 summarizes this information.

Table 43: CIM - SIM (P)SIG Group - Table Request Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigTblProvIndex	2 uimsbf/unique index of the (P)SIG Table Request Table	identifies the (P)SIG Table Request	read-create
simPsigTblProvTableId	enumerated/table identifier	identifies the table requested	read-create
simPsigTblNetworkId	2 uimsbf/the network identifier (see ETSI EN 300 468 [1])	identifies the network	read-create
simPsigTblONetworkId	2 uimsbf/the original network identifier (see ETSI EN 300 468 [1])	identifies the original network	read-create
simPsigTblTransStreamId	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream	read-create
simPsigTblServiceId	2 uimsbf/service identifier (see ETSI EN 300 468 [1])	identifies the service	read-create
simPsigTblBouquetId	2 uimsbf/bouquet identifier (see ETSI EN 300 468 [1])	identifies the bouquet	read-create
simPsigTblEventId	2 uimsbf/event identifier (see ETSI EN 300 468 [1])	identifies the event	read-create
simPsigTblONetworkId2loop	2 uimsbf/original network identifier (see ETSI EN 300 468 [1])	identifies the original network in the 2nd loop	read-create
simPsigTblNetworkIdOther	2 uimsbf/network identifier (see ETSI EN 300 468 [1])	identifies the other network	read-create
simPsigTblTransStreamId2OrO	2 uimsbf/transport stream identifier (see ETSI EN 300 468 [1])	identifies the transport stream in 2nd loop or other	read-create
simPsigTblSegmentNr	2 uimsbf/segment number(see ETSI ETR 211 [4])	number of the segment	read-create
simPsigTblProvPart	bslbf/the table part	the table part	read
simPsigTblProvPartNumber	2 uimsbf/ the table part number	the table part number	read-create

8.4.3.9 PID Provisioning Table

The ninth table is the PID Provisioning table. It is used for the C(P)SIG to request from the (P)SIG the PID value assigned by the head-end to a stream (ECM, EMM, or private data). The stream is identified by its type, the identifier of the CA system and the CA sub-system the stream belongs to, and a unique stream number. The reply to this request contains the PID value.

The PID Provisioning table is indexed by the Super CAS identifier (simPsigProvSuCasId), and the flow identifier (simPsigProvFlowId). A PID provisioning request is made by the C(P)SIG by simply reading the corresponding PID variable in the table, i.e. an SNMP get on simPsigProvFlowPID indexed by the right flow type, Super CAS identifier, and flow identifier. The actual implementation of the table may be just an interface to an appropriate database or application.

Table 44 summarizes this information.

Table 44: CIM - SIM (P)SIG Group - PID Provisioning Table

Object	Size/Description	Object Justification	Head-end/ CA Manager Maximum Access Right
simPsigPIDProvFlowType	enumerated/flow type specification	identifies whether the PID provisioning request is for an ECM, EMM, or private data flow	read-create
simPsigPIDProvSuCasId	4 uimsbf/Super CAS identifier	identifies the CAS	read
simPsigPIDProvFlowId	2 uimsbf/flow identifier	uniquely identifies the flow for a given flow type and CAS identifier	read
simPsigPIDProvFlowPID	2 uimsbf/flow PID	identifies the flow PID	read

8.4.4 Conformance Requirements

Table 45 summarizes the conformance requirements for management entities implementing the Simulcrypt Identification Module (SIM), by group.

Table 45: CIM - SIM (P)SIG Group - Conformance Requirements

Common Information Model - CIM Simulcrypt Identification Module Group	Management Entity Hosting or Representing an ECMG		Management Entity Hosting or Representing an EMMG		Management Entity Hosting or Representing a PDG		Management Entity Hosting or Representing a (P)SIG		Management Entity Hosting or Representing a C(P)SIG	
	mndt	optnl	mndt	optnl	mndt	optnl	mndt	optnl	mndt	optnl
Ident Group	X		X		X		X		X	
ECMG Group	X									
EMMG/PDG Group (without LAPG table)			X		X					
EMMG/PDG Group (LAPG Table)				X		X				
PSIG Group							X			
CPSI Group									X	

9 Timing and Playout Issues

9.1 Timing issues

In all systems there is a Crypto Period when data is scrambled with a particular Control Word. For STBs to regenerate the CW in time, the ECM playout has to be correctly synchronized with this CP.

To accommodate different synchronization approaches, the SCS will be responsible for requesting enough CWs and ECM packets in advance of their playout time. The timing diagram in figure 17 illustrates this relationship between CW generation, ECM generation, ECM playout and Crypto Period.

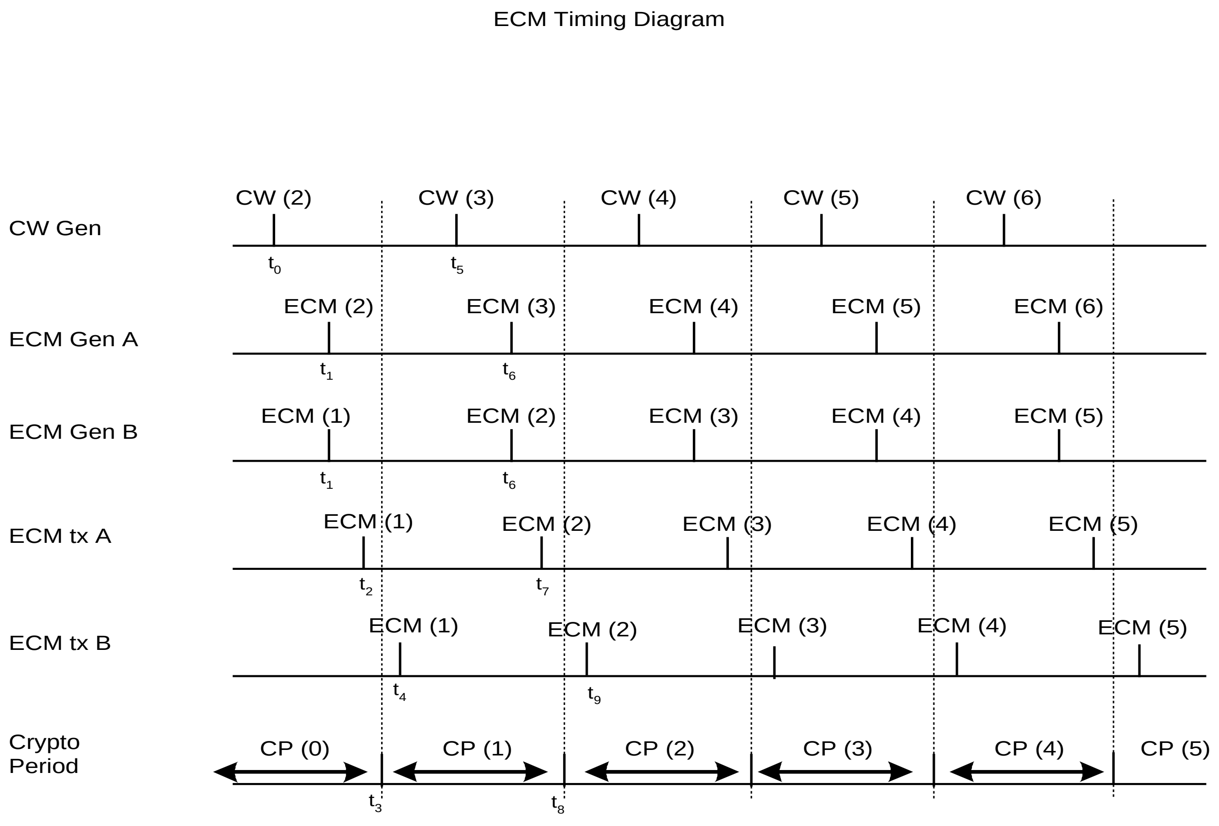


Figure 17: ECM timing diagram

At the beginning of CP(2), at t8, the scrambler begins using CW(2) to encrypt the signal. Each CA system shall ensure that its STBs obtain this CW in advance of this point.

CA system A achieves this by producing its ECM for a single CW only. As soon as ECM Gen. A receives CW(2), at t0, it is able to produce ECM(2), at t1. This ECM(2) is transmitted sufficiently prior to the beginning of CP(2), to ensure that the STB can obtain CW(2) before CP(2).

CA system B achieves the same result by producing its ECM for two CWs. As soon as ECM Gen. B receives CW(2), at t0, it is able to produce ECM(1), at t1. ECM(1), which encompasses CW(1) and CW(2), is transmitted from the middle of CP(1), at t4. This ensures that the STB can obtain CW(2) before CP(2) begins. After CP(2) begins, at t9, CW(2) and CW(3) are available in ECM(2).

9.2 Delay Start

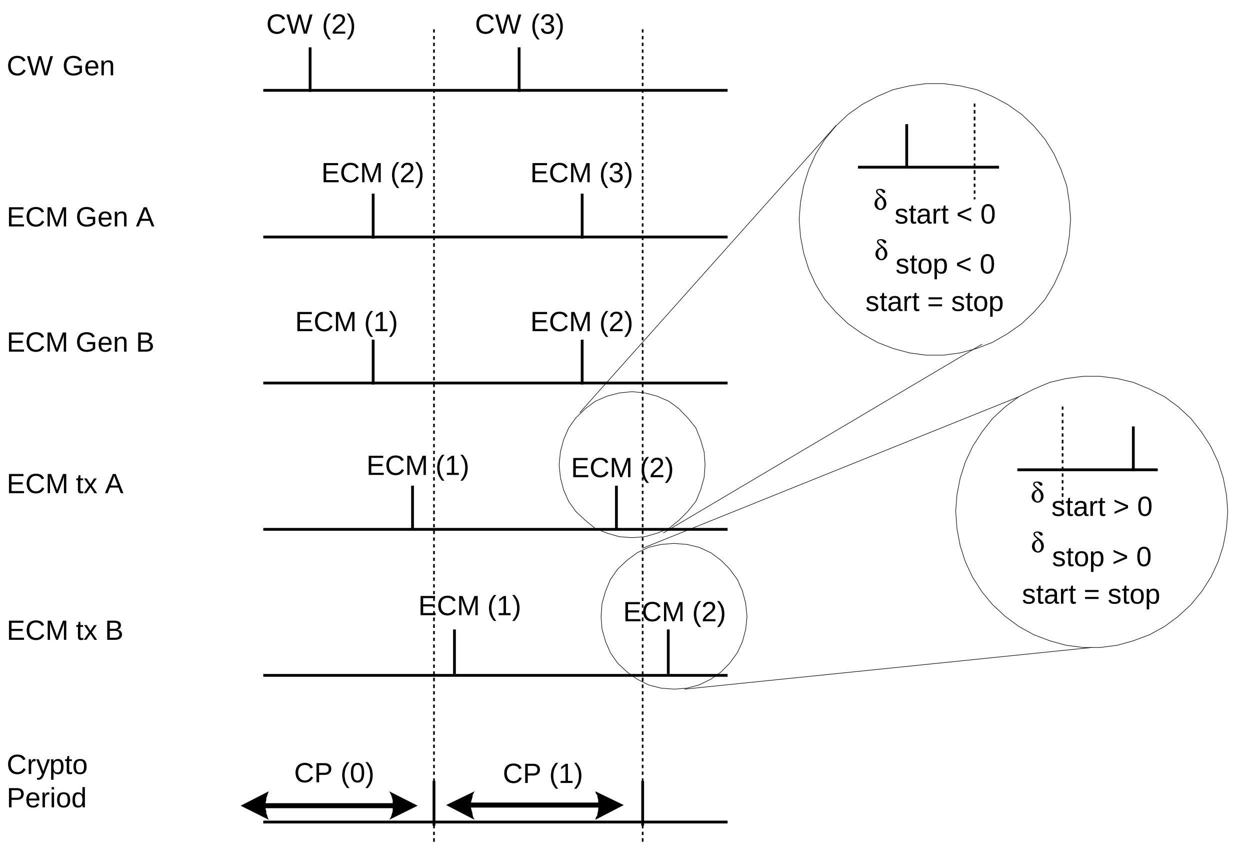


Figure 18: Delay_start and Delay_stop

delay_start: This signed integer represents the amount of time between the start of a Crypto Period, and the start of the broadcasting of the ECM attached to this period. If it is positive, it means that the ECM shall be delayed with respect to the start of the Crypto Period. If negative, it means that the ECM shall be broadcast ahead of this time. This parameter is communicated by the ECMG to the SCS during the channel setup.

delay_stop: This signed integer represents the amount of time between the end of a Crypto Period, and the end of the broadcasting of the ECM attached to this period. If it is positive, it means that the end of the ECM broadcast shall be delayed with respect to the end of the Crypto Period. If negative, it means that the ECM broadcast shall be ended ahead of time. This parameter is communicated by the ECMG to the SCS during the channel setup.

It is usual for delay_start and delay_stop to be equal, but this is not mandatory; this is illustrated in figure 19. The figure shows the case where ECM transmission is stopped *around* a CP boundary.

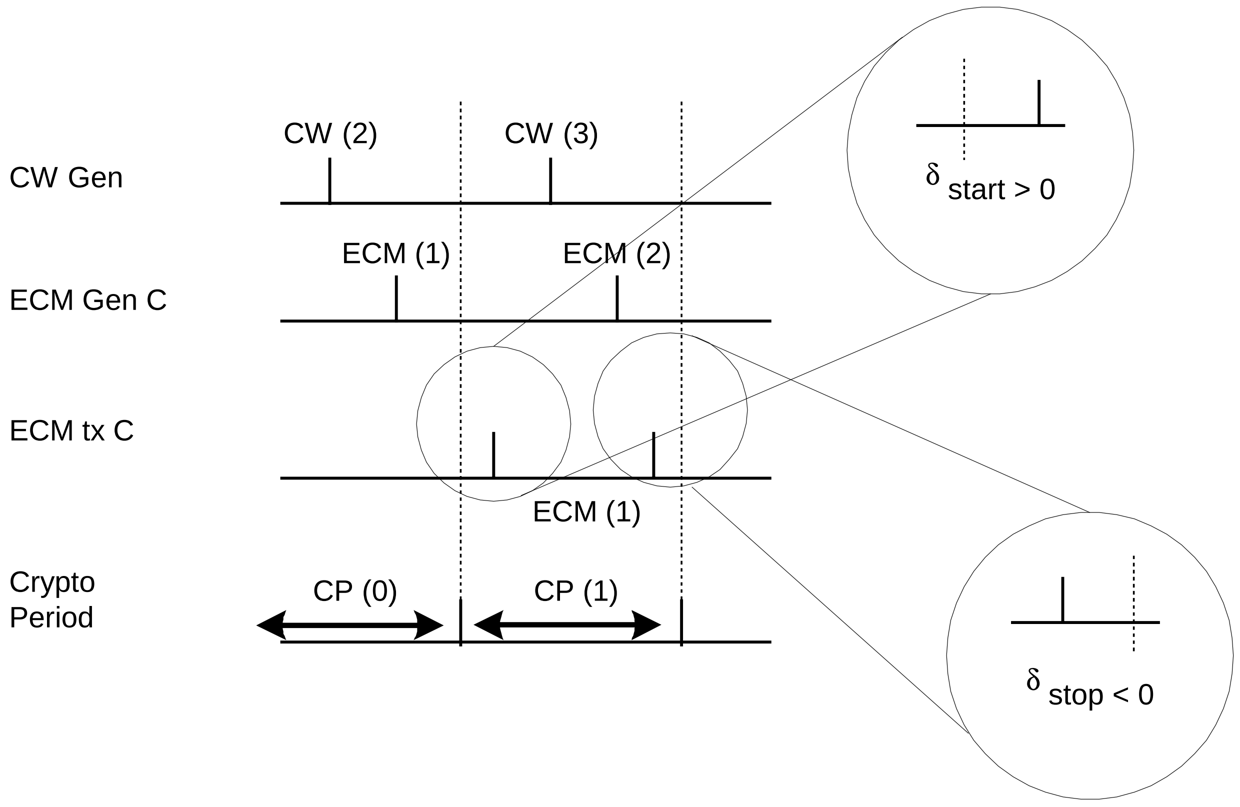


Figure 19: End of ECM transmission around cryptoperiod boundary

9.3 Playout Issues

9.3.1 ECMs

When an ECMG sends an ECM_datagram (in the ECM_response message) for a particular ECM stream it implicitly means that:

- the SCS shall trigger the playout of this ECM at the time calculated with the delay start parameter;
- the SCS shall stop the playout of the previous ECM of the same stream at the same time;
- in other words the playout of two ECMs of the same stream can never overlap;
- the playout of an ECM is also stopped by the SCS when the time calculated with the delay stop parameter is reached.

If an ECMG fails (i.e. the SCS times-out while waiting for an ECM_response message), the SCS has the option to extend the duration of the current Crypto period (e.g. to attempt to reconnect or switch to a backup device). In such as case the playout of ECMs is extended accordingly.

9.3.2 EMMs and Private Data

The MUX should playout EMMs/Private Datagrams in the order in which they arrive.

9.4 Crypto Period Realignment

If a new event starts in the middle of a Crypto Period (either from a program or a change in Access Criteria), Crypto Periods may need to be re-aligned. It is the SCS's responsibility to make sure that, during this re-alignment, no Crypto Period duration drops below the nominal_CP_duration. In other words, the Crypto Period can only be extended. For example, if 21:00:00 starts a new event and the nominal crypto period duration is 20 seconds, and a crypto period starts at 20:59:30, then the SCS is not allowed to make a 10 second crypto period between 20:59:50 and 21:00:00. Instead, if it needs to align Crypto periods with the start of the events, it shall lengthen the previous crypto period to 30 seconds, so that it ends exactly at 21:00:00.

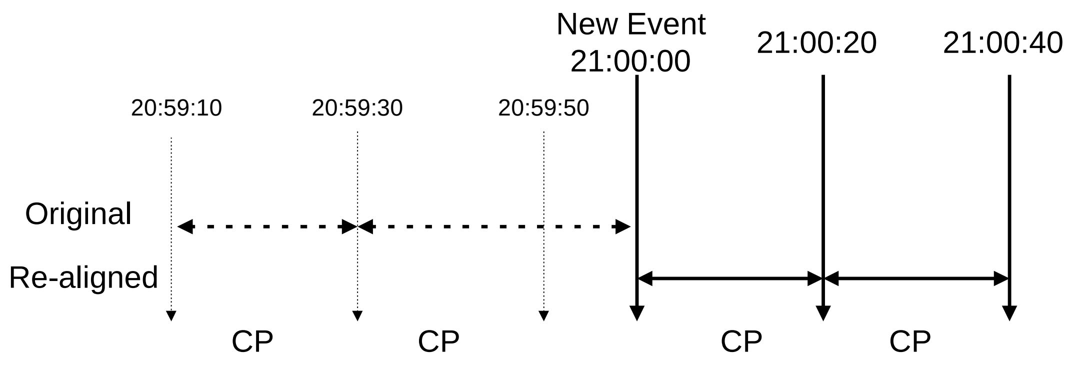


Figure 20: Event Realignment

Annex A (normative): System Layering

A.1 Introduction

Each paragraph in this annex describes a single system layer as defined within the OSI model. The presentation layer is not described in the context of the present document.

A.2 Physical Layer

The physical layer provides the physical facilities required to enable the linking together of hosts that need to exchange data.

The physical layer interface shall be ethernet. 10 Base-T (or another fully compatible layer) shall be used on all the interfaces defined by the present document.

A.3 Data Link Layer

The data link layer provides the facilities that allow two hosts that are physically and directly connected (without a third host separating the two) to exchange data. The functionality of the data link layer is covered by the ethernet protocols.

A.4 Network Layer

The network layer provides the facilities to allow two hosts to exchange data directly or indirectly in a network of intervening hosts and gateways.

The network layer, providing point-to-point communication, shall be IP (Internet Protocol - RFC 791 [13]). Hosts within IP are uniquely identified by their IP address.

A.5 Transport Layer

The transport layer provides the facilities to allow two hosts, either directly or indirectly connected, to exchange data over one or several interconnected networks. Additionally, the transport layer allows communication to take place based on connections between an individually addressable end-point on one host and another individually addressable end-point on the same host or on another host. The transport layer allows as well communication in broadcast mode.

The transport layer shall be TCP (RFC 793 [14]) or UDP (RFC 768 [12]), according to the application protocol.

A.6 Session Layer

The data exchange facility as provided by the session layer to the application layer has the following features:

- Connection Based or Broadcast (*): all communication takes place between two uniquely defined communication end-points, or from one to several communication end-points;
- Sequenced (*): All data transmitted arrives at its destination in the order it was sent;
- Reliable (*): Data integrity is maintained, no data is lost;

- Two-way (*): Both communication end-points of a particular connection can send and receive data;
- Unformatted: The session layer does not impose any structuring on the data it transports; the data presents itself to the receiver as an unformatted data byte stream (data structuring into messages is a responsibility of entities in the application layer).

The features (*) depend on the transport layer protocol (TCP or UDP) or on the session and presentation layers (specific or SNMP).

The number of connections that can be concurrently open at any one time is determined by the operating system under which the applications making use of the stream layer facilities execute. Additionally, in the event of an unexpected connection closure or connection loss and in the event of data read or write errors, the entity in the application layer that opened the connection or performs the read or write operation is notified.

The session layer, providing these facilities, shall be a socket stream interface.

A.7 System Layering Overview / Communications Protocol stack

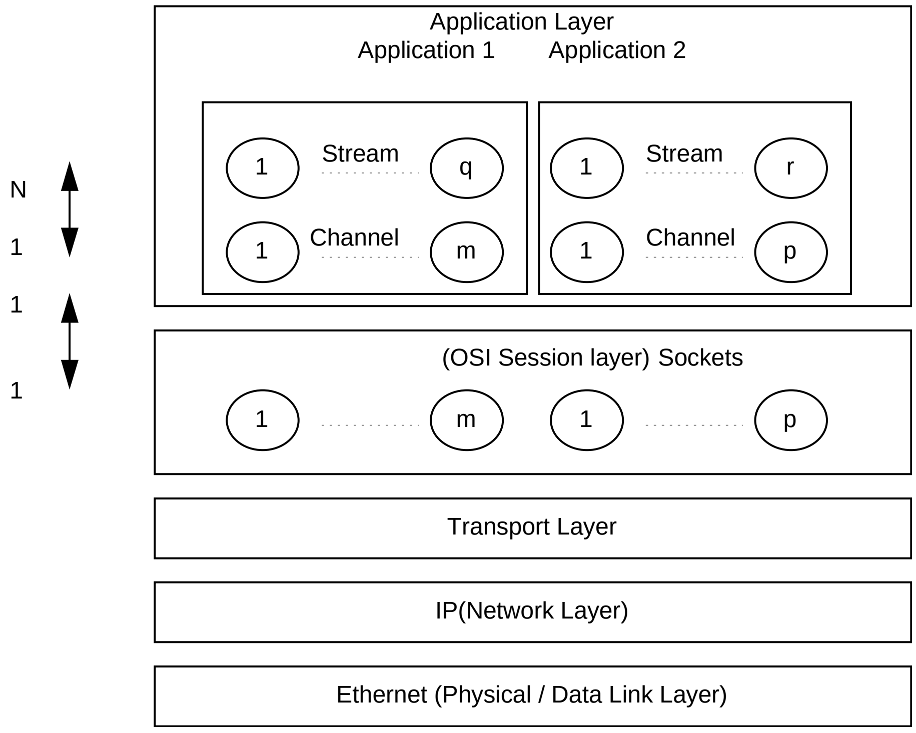


Figure A.1: Systems Layering overview diagram

On the left of this diagram, the mappings between the entities ports, sockets, connections and sessions is depicted:

- '1 <-> 1' indicates a direct association between an instance of an entity of a given type and an instance of a lower layer entity;
- '1 <-> N' indicates that potentially multiple instances of an entity of a given type map onto a single instance of a lower layer entity.

A.8 TCP or UDP Connection Establishment

Connections between client and server are initiated by the client. After establishment of a connection, both client and server have an open socket, identifying the connection, allowing them to exchange data. IP address information required by the client to open a connection is made available by the server in one of two ways:

- Statically: IP address information is defined by static methods;
- Dynamically. This method may use a DNS (Domain Name Server). The DNS can be consulted by the client to retrieve the required information;
- TCP port or UDP port information required by the client to open a connection is made available by the server. Port number information is defined by static methods.

Annex B (informative): SCS Coexistence

B.1 Introduction

This annex describes how the ECMG ⇔ SCS message interfaces could be used to communicate with another SCS (e.g. in a hot-standby configuration). In a Simulcrypt environment all the information that an SCS needs to communicate with a Mux, is encompassed by the Channel Status, Stream Status, CW Provision and ECM Response messages. This information can be passed to another SCS, which can then use this information to maintain its internal status.

B.2 Example scenario

The EIS will trigger the beginning of a CA event by sending access conditions and start and stop times to SCS_1. SCS_1 will then determine which ECMGs are involved with this CA event. It will establish connections, channels and streams with the appropriate ECMGs. During this establishment, each ECMG will pass, via the Channel and Stream Status messages, all ECMG specific data. SCS_1 will then begin passing CWs to the ECMGs, receiving ECM datagrams in response and synchronize the ECM playout.

In the example given in figure B.1, SCS_1 will additionally pass all messages received by each ECMG and the information contained in the CW Provision message on to SCS_2. This information can be transmitted to SCS_2 using the same interface as the SCS ⇔ ECMG. The main difference is that the CW Provision message, which is normally *sent* by SCS_1 to the ECMG, will now be *received* by SCS_2 from SCS_1.

This information will enable SCS_2 to reproduce the environment (connections, channels and streams) running on SCS_1.

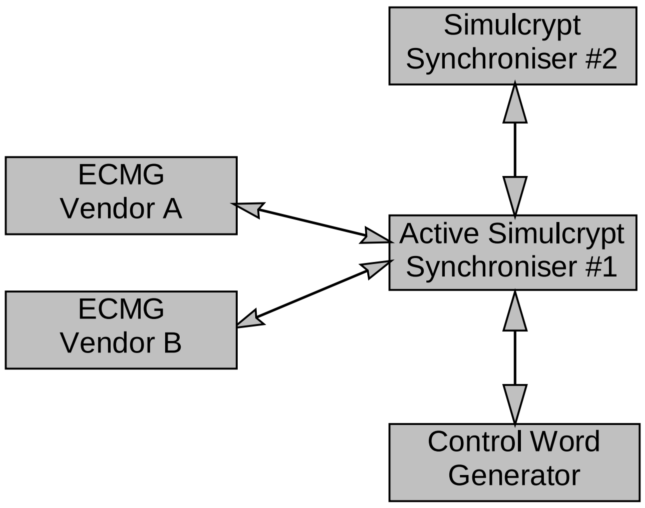


Figure B.1: Example of SCS redundancy

Annex C (informative): Control word generation and testing

C.1 Introduction

The control word generator is an integral part of the DVB Simulcrypt system, which generates low-level cryptographic keys that are used directly to scramble content. It should supply control words that meet certain statistical properties for randomness. Using appropriate (preferably physical) generating techniques and applying statistical tests will reduce to an acceptable level the probability of inadvertently generating control words with deterministic properties.

Since generating highly random control words and applying the appropriate tests for randomness imposes little incremental technical complexity or cost, there is great motivation for implementing the methods described below or their equivalent. Moreover, commercially available hardware and software solutions for this problem make it an easy task to generate control words with high confidence in their randomness qualities.

C.2 Background

The generated control words should approach as nearly as practicable true random sequences. In general, the criteria for producing cryptographically secure random sequences include:

- they have to *appear* random, that is they have to seem to an observer to possess the qualities of true random sequences;
- an observer should not be able to predict the next bit in a sequence even if armed with complete knowledge of the generating algorithm/hardware;
- a given sequence should not be reproducible by running a generator more than once using the same input;

Certifying sequences by applying the statistical tests described below can satisfy criteria 1 and 2, and seeded pseudo random sequences can meet these requirements. However any pseudo random algorithm is just as subject to attack as is an encryption algorithm.

Satisfying Criteria 1-3 produces sequences that approach true randomness (by most definitions) and are probably random enough for use as cryptographic keys in most applications, but (3) cannot be achieved using pseudo random techniques. This is not to imply that no pseudo random technique is acceptable for use in generating Simulcrypt control words, only that great care has to be taken to avoid generating sequences that that *appear* random but that can be successfully analysed by an attacker. This is not an easy task, but there are simple tests that can be applied to the algorithm during development that will help insure it satisfies criteria 1 and 2. For instance, an acceptable sequence should not be appreciably compressible (by more than about 1 or 2%) using commercial compression programs.

C.3 Generation

The best method to generate random sequences involves using physical phenomena to produce a Gaussian distributed white noise source with a flat magnitude spectrum (± 1 db, 100 Hz - 120 kHz). Physical methods typically use a thermal or radioactive noise source and are fed to a high-speed comparator to produce a digital output. Such sources are readily available and are simple to construct, and their output cannot be replicated even though an attacker may possess an exact copy of the generator hardware (i.e. They satisfy goal 3 above). This cannot be said of pseudo-random sequences generated by LFSRs even when operated in combinatorial arrangements. Various attacks can be successfully mounted against such methods.

Recommendation:

Simulcrypt control word generators should preferably use a physical source such as thermal noise, diode noise, MISC or the equivalent to generate random sequences. Pseudorandom techniques should be used advisedly and only after exhaustive testing to insure at least criteria 1 and 2 are met.

C.4 Control word randomness verification testing

There are numerous tests for randomness that can be applied to sequences, however in practical implementations there are two fundamental tests that can be relied upon to detect any significant defect in both pseudo- and true random sequences.

C.4.1 1/0 bias

The 1/0 bias test is usually performed on a sequence of convenient length and the comparator is trimmed to produce a logical 0 probability, p(0), of 0,5.

$p(0) = 0{,}5 + e \pm 0{,}001$

where e = bias factor

XORing bits together will exponentially converge to:

$p(0) = 0{,}5$

A two-bit example:

$p(0) = (0{,}5 + e)^2 + (0{,}5 - e)^2 = 0{,}5 + 2e^2$

A four-bit example:

$p(0) = 0{,}5 + 8e^4$

Recommendation:

1/0 bias detection tests should be run on the generated sequences, and corrections should be made when needed.

C.4.2 Autocorrelation

Autocorrelation is defined as a discernible relationship in the variation of a variable over time. In order for a random sequence to be non-deterministic, its autocorrelation property has to be minimized. There are many algorithms available to measure autocorrelation properties, and most specify the test to be run on small (100 kbit/s) blocks where actual values should not vary more than three standard deviations from expected values for two consecutive blocks. Depending on the required speed, the tests may be run continuously or at frequent intervals.

Recommendation:

Autocorrelation tests should be run on sequences at intervals sufficient to ensure with reasonable certainty that no deterministic properties exist.

C.5 Testing locations

Although it is recommended that the above tests be conducted at the output of the control word generator, CA operators should consider conducting similar tests at the input of the of their ECMGs to confirm the randomness of the control words they receive.

Annex D (informative): Security Method for the SCS ↔ ECMG Interface

The following is a recommended method for encrypting the clear control word data traversing non-secure networks between the SCS and ECMG devices. For simplicity, it is performed at the application level within the Simulcrypt protocol. Since the CW data consists of an 8-byte block transferred over the interface once per CP, it is quite straightforward to encrypt using a standard block encryption algorithm. Key management as specified here is both simple and effective, requiring minimal resources. This method is facilitated by using the CW_encryption parameter and its sub-parameters as specified in the CW_provision message format.

D.1 Algorithm Selection

Although the Algorithm_type parameter in the CW_encryption parameter allows the selection of multiple encryption algorithms, it is recommended that the head-end/uplink operator and the external CA provider(s) agree beforehand on the algorithm to be used. The Algorithm_type parameter is also useful in specifying the key entropy (i.e. 40 bit vs. 56 bit) used with a particular algorithm where external CA systems are capable of both versions. Weakened key strengths are sometimes necessary to satisfy the requirements of governmental authorities. In the 40 bit case, two padding bytes of value 0x00 shall replace bytes 5 and 6 of the selected key as shown below.

If the selected key value is:	9B F2 74 A0 B1 9A E6
Then the 40 bit adjusted key is:	00 00 74 A0 B1 9A E6

It is the responsibility of the Simulcrypting participants to select an algorithm that is strong enough, appropriate to this application, and compliant with national restrictions. Examples of algorithms that may be suitable can be found in FIPS 46-1 [23].

The encryption mode specified is Electronic Code Book and can be used with any 56 bit block cipher. Encryption and decryption are shown in Figure D.1 and D.2 respectively.

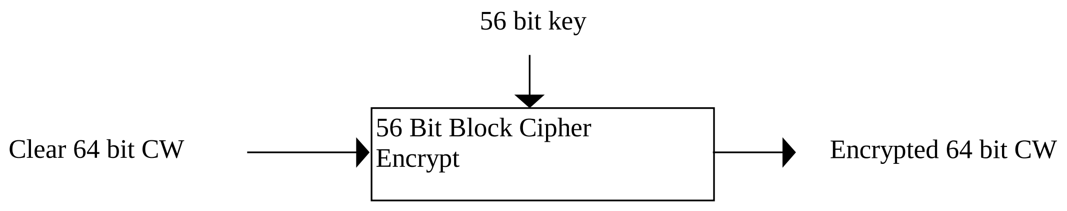


Figure D.1: Control Word encryption (SCS head-end/uplink function)

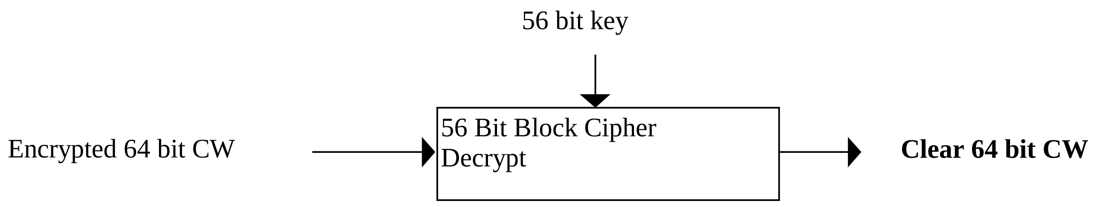


Figure D.2: Control Word decryption (External ECMG function)

D.2 Control Word processing

Only the 8-byte control word data field in the CP_CW_combination parameter is subject to encryption. The control word data is parsed from the 2-byte CP data field prior to encryption and is re-concatenated with it following encryption. This maintains an 8-byte plaintext field and requires only a single iteration of the encryption algorithm. Moreover, additional processing steps such as padding are avoided. Figure D.3 illustrates the parsing operation.

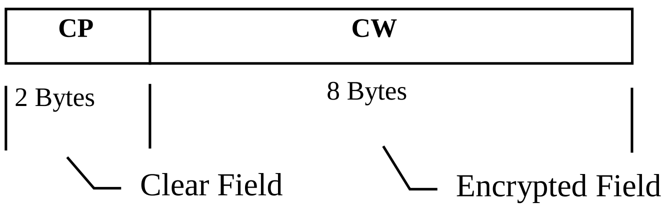


Figure D.3: CP_CW_combination field parsing

SCS devices need only implement the encryption mode of the algorithm, while ECMG devices need only implement the decryption mode.

D.3 Key Management

Under the present document, key management involves the generation, selection, and distribution of key data to be used in the algorithm for CW encryption. This has to be done in a standardized way in order to insure interoperability between SCS and ECMG devices.

D.3.1 Key Generation/Distribution

The encryption algorithm implementation in the present document uses key data generated by a good random source (see annex C) and stored on media for use in both SCS and ECMG devices. Each SCS (head-end/uplink) operator is responsible for securely generating the key data for his SCS ⇔ ECMG interface(s) and for securely distributing it upon request to all external CA operators for use in their ECMGs. In the event a key list is suspected or known to be compromised, the SCS operator is responsible for generating and distributing a replacement list as soon as possible. Moreover, SCS operators may agree to generate and distribute new key lists on a periodic basis to insure key security. The suggested size of the keyspace is 2 048 bytes. This provides 292 possible 7-byte keys per key list for use in the algorithm.

Highly-cautious SCS operators may wish to detect and exclude weak and semi-weak keys when generating keylists, however due to the nature of the application, the occasional use of these keys does not pose a security threat. Moreover, if the random source is robust, neither weak nor semi-weak keys are likely to be generated with any significant probability.

To facilitate seamless transition from one key list to another, two independent 2 048-bit lists are used. Both the active list and the future list shall reside on both the SCS and all Simulcrypting ECMGs before the head-end/uplink facility performs the transition function. The list in current use shall be identified using the most significant bit in the CW_encryption parameter; this bit is designated as **Key_list_sel** (key list select bit). When reset (0), the *A* key list is in use; when set (1), the *B* list is in use. Transition from one key list to the other is accomplished by setting or resetting the Key_list_sel bit.

In order to avoid placing headers in the key lists, the SCS and ECMG software shall examine the Key_list_sel bit at the time lists are loaded to determine whether the list being loaded is to be designated the *A* or *B* list. If the active list is *A*, the list is loaded as *B*; if the active list is *B*, it is loaded as *A*. If no lists are active (during initialization) the software shall allow the operator to manually enter the designator.

D.3.2 Selection

Selection of the 56-bit key data for use in the algorithm is accomplished by using a randomly-generated 11-bit key select vector. This vector is used as an index into the 2 048 key space as shown in Figure D.4.

Only every seventh address is legal as a key select vector (i.e. 0, 7, 14, 21, etc.) through 2 037. This provides 292 possible completely independent keys.

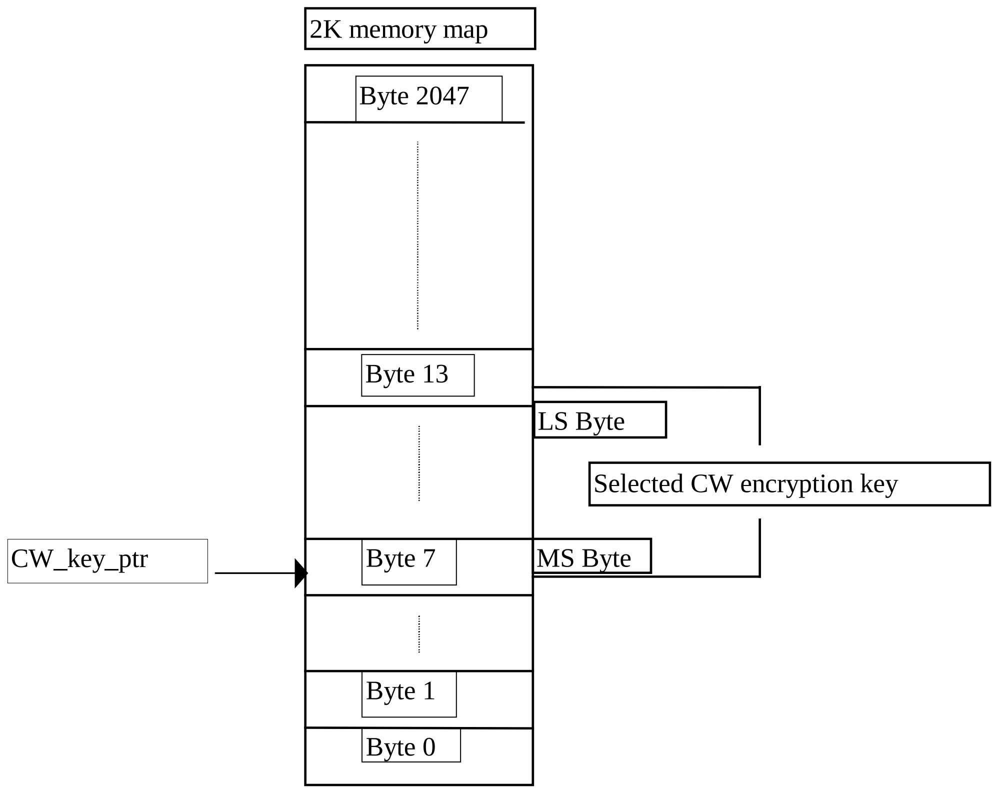


Figure D.4: Example of Control Word encryption key selection

The CW_encryption parameter is represented as a two-byte field as shown below. Bits 0-10 comprise the CW_key_pointer, and bits 11-13 are used to designate the cryptographic algorithm or key size in use. Bit 14 is used to invoke fixed key mode, and bit 15 is the A/B key list designator (Key_list_sel).

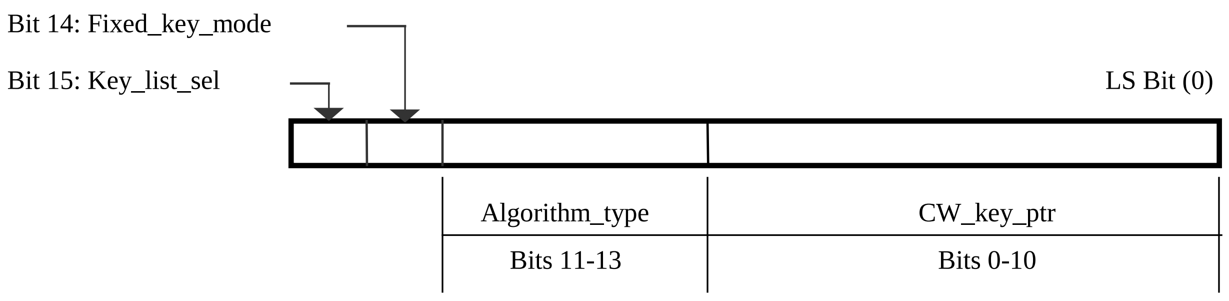


Figure D.5: CW_encryption Parameter (two bytes)

New keys are selected for each instance of the CW_provision message, and where more than one CW is conveyed in a single CW_provision message, all CWs are encrypted using the same key. One key is required per each CW encryption or decryption operation.

In the event 40 bit keys are in use, the padding method previously described is applied to the selected key data.

D.3.3 Key Pointer Distribution

The present document uses a symmetric algorithm, which is the same key is used for both encryption and decryption. Therefore both the SCS and ECMG shall use the same key as selected by the CW_key_ptr parameter within the CW_encryption parameter. This parameter is generated in the SCS and shall be conveyed in the CW_provision message immediately following the CP_CW_combination parameter. Subclause 5.5.7 specifies the required CW_encryption parameter. The entire table is reproduced below for convenience; it includes the sub-parameters specific to this security method.

Parameter	Number of instances in message
ECM_channel_id	1
ECM_stream_id	1
CP_number	1
CW_encryption CW_key_ptr Algorithm_type Fixed_key_mode Key_list_sel	**0 to 1**
CP_CW_combination	CW_per_msg.
CP_duration	0 to 1
access_criteria	0 to 1

CW_encryption: This parameter contains the four sub-parameters listed below that enable encrypting of control words over the SCS ⇔ ECMG interface. If the parameter is included in the CW_provision message, control word scrambling is invoked; if omitted, CWs are being issued in the clear.

CW_key_ptr: This 11-bit field contains an index that points to the active CW encryption key contained on a 2 048 byte (or smaller) key list. It is a randomized value (1 of 292) generated within the SCS which points to the MS byte of a seven-byte key used in block cipher Electronic Code Book mode. Legal values include every 7th address in the 2 048 space (0, 7, 14, 21, etc).

algorithm_type: This field may be used either to signal the type of encryption algorithm in use for CW encryption or the key length of a given algorithm. It is useful where it may be desirable to change either the fundamental algorithm or its key length providing both the head-end and all external ECMGs have the appropriate capabilities. In most cases, the Simulcrypting participants will agree on these parameters in advance.

fixed_key_mode: This bit is used to bypass the key list and use a key contained in ROM for encryption of the control word. This key will need to be agreed upon by all Simulcrypting participants. The security method described in Annex *n* uses a defined fixed key value.

key_list_sel: In order to facilitate smooth changeover from one keylist to another, two independent lists should be maintained on each ECMG and the SCS. This bit allows selection of one of the two lists as the active list.

D.3.4 Fixed Key Mode

There are instances in practical cryptosystems when it is desirable to temporarily encrypt messages under a common fixed key. This mode is useful when troubleshooting system failures, during initialization, or when the SCS has not installed a key list. It provides a fallback mode that is more secure than sending control words in the clear. When invoked, the SCS encrypts all CWs under the same fixed key, which is located in ROM and is not part of any key list. Fixed key mode is invoked using the Fixed_key_mode bit (14) of the CW_encryption parameter (see figure D.5). When set (1), the encryption key is selected from the appropriate (A or B) key list as designated by the pointer value and the key list select bit. When reset (0), the fixed key is used as the encryption key. The ECMG cannot invoke fixed key mode. The value of the fixed key is:

- 56-bit version: 4D A1 9F F0 AF 6B 8F;
- 40-bit version: 00 00 9F F0 AF 6B 8F.

It was generated in accordance with annex C. Since this mode can be invoked at any time by the SCS, the ECMG software should include an alarm to alert operators when fixed key mode is in effect.

D.4 Encryption Function Toggling

Although it is unlikely that either head-end operators or external CA providers will wish to discontinue CW encryption once it is invoked, there may be an occasional need to temporarily revert to clear CW transmission for system troubleshooting or for other reasons. This is accomplished by the SCS deleting the CW_encryption parameter from the CW_provision message. If the ECMG does not receive the CW_encryption parameter in the CW_provision message, it does not apply decryption to the received control words. Clear control should only be sent by prior arrangement between head-end operators and external CA providers or in emergencies; the ECMG does not have the capability to invoke clear CW transmission. ECMG designers may wish to include an alarm in their firmware that would activate upon detection of clear control word mode.

Annex E (informative): Summary of Requirements for C(P)SIG ↔ (P)SIG interface

This section provides a high-level summary of the requirements imposed on the head-end system and each CAS in support of the C(P)SIG ⇔ (P)SIG interface.

The head-end and each CAS comply with the requirements presented in annex E, clauses E.1 and E.2, respectively, as well as all specifications in documented in the above-described sections.

E.1 Head-end system requirements

The head-end shall be solely responsible for:

1) Generating and broadcasting one or more transport streams (TSs) that conform to MPEG-2 and DVB specifications (EN 300 468 [1] through ETSI ETR 289 [5] and ISO/IEC 13818-1 [7]).

2) Ensuring the MPEG-2 and DVB syntactic and semantic integrity of these TSs.

3) Ensuring that each TS include all standard MPEG-2 PSI and DVB SI tables that are required by MPEG-2 and DVB specifications (specifically, ETSI ETR 154 [2] and ETSI ETR 211 [4]). There is no requirement on the head-end to generate any standard tables that are optional (e.g. EIT Schedule).

4) Ensuring that each PSI and SI table (required or optional) include all descriptors required by MPEG-2 and DVB specifications, respectively (specifically, ETSI ETR 154 [2] and ETSI ETR 211 [4]), and optionally generating any other standard tables and/or descriptors defined by MPEG-2 and/or DVB. See also the next requirement.

5) Including, as required per DVB and CAS-specific conditional access (CA) requirements, CA_descriptors in all PSI CAT and PMT tables. However, the head-end shall not include any private_data_bytes in CA_descriptors whose CA_system_id belongs to any CAS that is interfaced with the head-end (refer to ISO/IEC 13818-1 [7]).

6) Optionally generating any private tables (i.e. with user-defined table_id) with MPEG-2 section syntax. (This is outside the scope of this interface. Commercial agreement should be used to avoid conflict with CAS-generated private tables transmitted on the PDG ⇔ MUX Interface).

7) Optionally generating, for its own purposes, an ordered list of private descriptors (i.e. with user-defined tag value) for insertion in any PSI/SI table; and broadcasting this list of descriptors. Either of two methods may be used to prevent conflict with private descriptors generated by any CAS (clause E.2):

   - logically separating the head-end's private descriptors with a private_data_specifier descriptor, whose private_data_specifier value is used only by the head-end (per commercial agreement). The present document strongly recommends this approach; or

   - accordingly, the head-end may employ commercial agreement or some other unspecified means to prevent conflicts in private descriptor tag usage and interpretation.

8) Scheduling DVB SI services and events. While this is not specific to the C(P)SIG ⇔ (P)SIG Interface, the head-end shall inform the CAS(s) of new and/or changed services and events, per the "triggering" requirement presented below.

9) Configuration and initialization of the head-end (P)SIG processes defined in subclause 8.2.1 of the present document. Specifically, either a single PSISIG, or a {PSIG+SIG} pair, shall be configured.

10) Hosting one or more CAS and their C(P)SIG processes, as defined in subclause 8.2.1, and supporting each CAS per the requirements presented.

11) Maintaining mutual separation and independence of each CAS.

12) Triggering (signalling) each CAS before, or upon, the occurrence of any or all of the following actions:

- new DVB SI EIT Following event;
- new head-end information about a future DVB SI EIT event;
- creation, modification or closure of an ECM stream;
- user-defined (per commercial agreement);
- in order to enable triggering, the CAS shall first tell the head-end which types of triggers it wants to receive, on a per-service basis. In addition, the CAS shall specify how far in advance of the action it wants to receive the trigger (if possible);
- in this context, "service" means either a DVB-defined service or an MPEG-2 program, either being defined by a PAT program_number entry and a PMT section.

13) Fulfilling CAS requests for the insertion of a list of private descriptors in standard PSI and SI tables. The head-end shall therefore include any private descriptor requested by any CAS, provided that (a) the descriptor is syntactically valid, and (b) the integrity of the PSI/SI table(s) can be maintained. The descriptor list insertion may be synchronized with a triggered action (see above), or asynchronous, as requested by the C(P)SIG (see below).

14) Preventing conflict, among CASs and with the head-end, in the usage and interpretation of private descriptors. Either of two methods may be used:

- logically separating each respective CAS's list with a private_data_specifier descriptor, whose private_data_specifier value is used only by the CAS (per commercial agreement). The present document strongly recommends this approach; or
- employing commercial agreement, or some other unspecified means, to prevent conflicts in private descriptor tag usage and interpretation.

15) Fulfilling CAS requests to receive data from any PSI or SI currently transmitted. If the head-end generates EIT Schedule tables, they shall always be returned to the CAS in the clear, even if scrambled for broadcast (see ETSI EN 300 468 [1], subclause 5.1.5).

16) Reporting specified error conditions per the above-defined interactions with the CASs.

17) Ensuring and maintaining all communications, networking, database access and so forth within the head-end, so as to ensure that all other requirements are met. Such intra-head-end interfaces are implementation-dependent, and outside the scope of the present document.

E.2 CAS's C(P)SIG requirements

Each CAS shall be solely responsible for:

1) Informing the head-end as to the types of event- and ECM-related triggers it wants to receive, on a per-service basis. In this context, "service" means either a DVB-defined service or an MPEG-2 program defined by a PAT program_number entry and a PMT section;

2) Processing action triggers received from the head-end;

3) Optionally generating, for its own purposes, an ordered list of private descriptors for insertion in any standard PSI/SI table(s) generated by the head-end:

- each list supplied is associated with a CAS-specific private_data_specifier (per commercial agreement). It is the responsibility of the head-end to decide whether to separate descriptor lists via private_data_specifier descriptors, or by some other unspecified means. The present document strongly recommends the use of private_data_specifier descriptors;
- the CAS may request that the descriptor list insertion be either synchronized with a triggered action, or asynchronous.

4) Sending the descriptor list to the head-end for broadcast:

- a CAS cannot actually send or modify any PSI/SI table by itself. It can only request that the head-end insert its private descriptors in given tables.

5) Maintaining each descriptor list and each set of the CA_descriptor private_data_bytes up-to-date, per CAS requirements. The CAS shall ensure that private descriptors are compatible with the PSI/SI tables in which they are transmitted, at the time they are transmitted;

6) Reporting specified error conditions per the above-defined interactions with the head-end;

7) Ensuring and maintaining all communications, networking, database access and so forth within the CAS, so as to ensure that all other requirements are met. Such intra-CAS interfaces are implementation-dependent, and outside the scope of the present document;

8) Requesting for any PSI/SI table.

Annex F (informative): Example of C(P)SIG ↔ (P)SIG Connection-oriented Solution Configuration

The sample configuration presented here is referred to in subclause 8.3.

Figure F.1 shows this reference configuration at the component and channel level. Figure F.2 depicts all connections at the stream level.

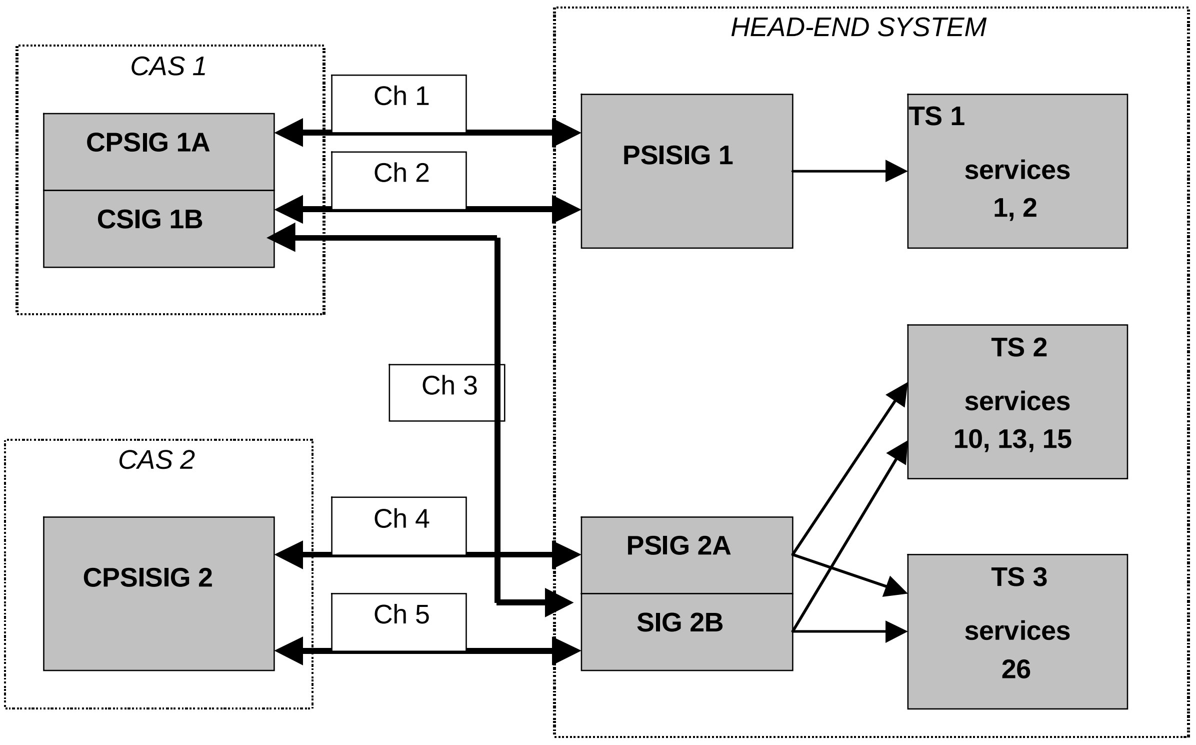


Figure F.1: Example: Channels *(Ch1 - Ch5)* in a head-end system with two CASs

F.1 Head-end processes and configuration data

The head-end in this example includes two (P)SIGs. (P)SIG 1, a PSISIG, serves one transport stream, TS1. (P)SIG 2 consists of two processes, a PSIG and a SIG; they serve two transport streams, TS2 and TS3.

Tables F.1 and F.2 show the information that the head-end *might possibly* require to define the (P)SIG head-end processes *(recall that the present document does not define the format of this data):*

- table F.1 defines TS-related parameter;
- table F.2 defines (P)SIG-related parameters.

For simplicity, this data has been organized into relational tables in near-canonical form (some cells may contain lists of values). The following conventions are used in this and all other configuration tables presented in this annex:

Columns headed by names appearing in *UNDERLINED ITALICS* indicate key fields. A specific system implementation might provide a different set of key fields for any table.

Cells marked with an asterisk (*) indicate data whose values are not of specific interest in understanding this example.

In addition, since the actual representation of this data is beyond the scope of the present document, the following are not to be inferred from this example:

- the completeness of this data: other data might be required;
- the location of this data (e.g. head-end and/or CAS); this would be determined by technical and/or commercial requirements.

Whether this data is static or dynamic; some of each would generally be required.

Table F.1: TS configuration data *(example, not normative)*

TRANSPORT STREAM	TRANSPORT_STREAM _id	ORIGINAL _NETWORK _id	NETWORK_ ID	SERVICE_ IDs
TS1	*	*	*	1, 2
TS2	*	*	*	10, 13, 15
TS3	*	*	*	26

Table F.2: (P)SIG configuration data *(example, not normative)*

(P)SIG NAME	***IP ADDRESS***	TCP PORT NO.	(P)SIG TYPE	TRANSPORT_STREAMs
PSISIG 1	*	*	3	TS1
PSIG 2A	*	*	1	TS2, TS3
SIG 2B	*	*	2	TS2, TS3

F.2 CAS processes and configuration data

Two CASs are hosted by the head-end. It is assumed that the head-end knows the CA_system_id (CASID) and private_data_specifier(s) used by each CAS.

The C(P)SIG of CAS 1 consists of two processes, a CPSIG and a CSIG. The C(P)SIG of CAS 2 is a CPSISIG.

Table F.3 shows the information that the head-end *might possibly* require to define the C(P)SIG CAS processes *(recall that the present document does not define the format of this data).*

Table F.3: C(P)SIG configuration data *(example, not normative)*

C(P)SIG NAME	***CUSTOM_CAS_id***		IP ADDRESS	TCP PORT NO.	C(P)SIG TYPE
	CASID	***extension***			
CPSIG 1A	CASID-1	1	*	*	4
CSIG 1B	CASID-1	2	*	*	8
CPSISIG 2	CASID-2	1	*	*	0xC

F.3 Channels and configuration data

Five channels are present in the system:

- channels *Ch1* and *Ch2* allow CAS 1 to request data from, and insert private data into, TS1. *Ch1* supplies MPEG-2 PSI private data for insertion, and *Ch2* similarly supplies DVB SI private data;

NOTE: CPSIG 1A may *request* DVB SI data via *Ch1*, and CSIG 1B may *request* MPEG-2 PSI data via *Ch2*. The kind of table data returned by the head-end via the table_response message is not restricted by the type of C(P)SIG that issued table_request. This principle holds, clearly, to all the other channels defined.

- similarly, channel *Ch3* allows CAS 1 to exchange data with TS2 and TS3. Only DVB SI private data is supplied for insertion;
- channels *Ch4* and *Ch5* allow CAS 2 to request data from, and insert private data into, TS2 and TS3. *Ch4* supplies MPEG-2 PSI private data, and *Ch5* similarly handles DVB SI private data.

Table F.4 shows the information that the head-end *might possibly* require to establish all the channel connections *(recall that the present document does not define the format of this data).*

Note the interpretation of the trigger_list values used:

- **0x0000003F ('…00111111')** says that the channel supports triggering on six kinds of actions: EIT future events, EIT following events, new ECM streams, ECM stream closure, ECM PID modification, and modification of ECM access criteria;
- **0x0000003D ('…00111101')** says that the channel supports triggering on all kinds of actions listed above, with exception of EIT future events.

Table F.4: Channel configuration data *(example, not normative)*

CHANNEL NAME	CUSTOM_ CHANNEL_ ID	C(P)SIG NAME	(P)SIG NAME	TRIGGER_ LIST	MAX_ STREAMS
Ch1	1	CPSIG 1A	PSISIG 1	0x3F	*
Ch2	2	CSIG 1B	PSISIG 1	0x3F	*
Ch3	3	CSIG 1B	SIG 2B	0x3F	*
Ch4	4	CPSISIG 2	PSIG 2A	0x3D	*
Ch5	5	CPSISIG 2	SIG 2B	0x3F	*

F.4 Streams and configuration data

Eight streams are defined in this sample configuration; they are depicted in Figure F.2.

Table F.5 shows the information that the head-end *might possibly* require to establish all the stream connections *(recall that the present document does not define the format of this data).*

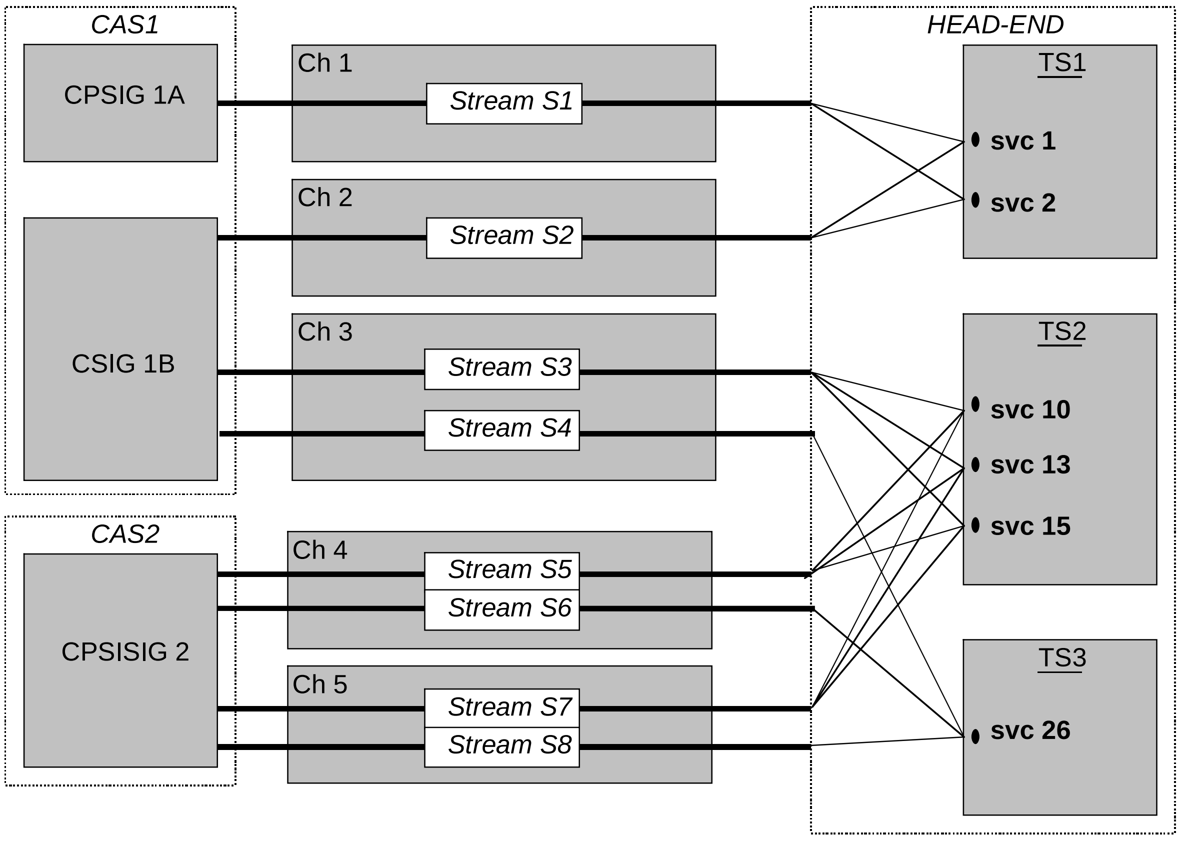


Figure F.2: Example: Stream interconnections for the example in figure F.1

Table F.5: Stream configuration data *(example, not normative)*

STREAM NAME	CUSTOM_ STREAM_id	CUSTOM_ CHANNEL_ ID	TRANSPORT STREAM	SERVICE_ids	SERVICE_ PARAMETERS
S1	1	1	TS1	1, 2	*
S2	1	2	TS1	1, 2	*
S3	2	3	TS2	10, 13, 15	*
S4	3	3	TS3	26	*
S5	2	4	TS2	10, 13, 15	*
S6	3	4	TS3	26	*
S7	2	5	TS2	10, 13, 15	*
S8	3	5	TS3	26	*

Annex G (normative): ASN.1 MIBs description

G.1 SIM MIB

```
SIM-MIB DEFINITIONS::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Counter32, IpAddress,
    BITS FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, RowStatus,
    DisplayString FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP     FROM SNMPv2-CONF;

simMIB MODULE-IDENTITY
    LAST-UPDATED  "9707021700Z "
    ORGANIZATION  "DVB Simulcrypt Technical Group "
    CONTACT-INFO  " ---  "
    DESCRIPTION
         "  The MIB module for defining DVB Simulcrypt Conditional
            Access System configuration information. "
   ::= {1 3 6 1 4 1 2696 1 1}

AdministrativeState::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     " Administrative state as defined by ITU-T Recommendation X.734 [10]. "
    SYNTAX BITS
                        {
                    locked(0),
                    unlocked(1),
        shuttingDown(2)
        }

ECMGSuCasId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "The Super_CAS_id is represented as a 4 bytes unisgned integer. "
    SYNTAX Unsigned32

FlowSuCasId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "The Super_CAS_id is represented as a 4 bytes unisgned integer. "
    SYNTAX Unsigned32

ECMGStreamId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     " The ECM_stream_id is represented as a 2 bytes unsigned integer. "
    SYNTAX  INTEGER (0..65535)

FlowId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     " The flow identifier is represented as a 2 bytes unsigned integer. "
    SYNTAX  INTEGER (0..65535)

ECMGChannelId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
         " The ECM_channel_id is represented as a 2 bytes unsigned integer. "
    SYNTAX  INTEGER (0..65535)

EMMGCommCapability::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Type of communications capability between EMMG/PDG and Multiplexer:
    TCP or UDP or both. "
    SYNTAX BITS
                        {
        both(0),
                    tcp(1),
                  udp(2)
        }
```

```
EMMGCommType::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Type of communications capability between EMMG/PDG and Multiplexer:
    TCP or UDP. "
    SYNTAX BITS
                        {
                    tcp(0),
                  udp(1)
        }

EMMGDataType::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Type of data carried in the EMMG/PDG Multiplexer stream. "
    SYNTAX BITS
                    {
                    emm(0),
                    other(1)
           }

EMMGClientId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
         "  Client_id: The client_id is a 32-bit identifier. It shall
        identify uniquely an EMMG/PDG across all the EMMGs/PDGs
        connected to a given MUX. To facilitate uniqueness of this
        value, the following rules apply:
        * In the case of EMMs or other CA related data, the two first
        bytes of the client_id should be equal to the two bytes of
        the corresponding CA_system_id.
        * In other cases a value allocated by DVB for this purpose
        should be used. "
    SYNTAX Unsigned32

EMMGStreamId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
         "Indicates the Data Stream Id. "
    SYNTAX  INTEGER (0..65535)

EMMGChannelId::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
         "Indicates the Data Channel Id. "
    SYNTAX  INTEGER (0..65535)

TriggerType::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  The type of a trigger in a PSI generator. "
    SYNTAX BITS
                    {
                    dvbEvent(0),
                    futureDvbEvent(1)  ,
            newEcmStream(2),
            flowPidChange(3),
            accessCriteriaChange(4),
            ecmStreamClosure(5)
           }

ECMTriggerType::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  The type of an ECM trigger in a PSI generator. "
    SYNTAX BITS
                    {
                    ecmStreamOpen(0),
                    ecmStreamClose(1)  ,
            ecmStreamChange(2),
            accessCriteriaChange(3)
           }


DescriptorStatus::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  The return status of descriptor insertion. "
    SYNTAX BITS
                    {
                    success(0),
                    unknownTrigger(1)  ,
            unknownLocation(2),
            unsupportedDelay(3),
            unknownContext(4),
            unknownOtherTS(5),
            unknownNetwork(6),
            unknownTS(7),
            unknownES(8),
            unknownBouquet(9),
            unknownEvent(10),
            tableNotSupported(11),
            tableFull(12),
            other(13)
           }

InsertLocation::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Descriptor insertion location. "
    SYNTAX BITS
                    {
                    pmtLoop1(0),
                    pmtLoop2(1)  ,
            cat(2),
            nitLopp1ActualNet(3),
            nitLoop2ActualNet(4),
            nitLopp1OtherNet(5),
            nitLoop2OtherNet(6),
            batLoop1(7),
            batLoop2(8),
            sdtActualTS(9),
            sdtOtherTS(10),
            eitPFActualTS(11),
            eitPFOtherTS(12),
            eitScheduleActualTS(13),
            eitScheduleOtherTS(14)
           }

ProvTableId  ::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Provision table identifier. "
    SYNTAX BITS
                    {
                    pat(0),
                    cat(1)  ,
            pmt(2),
            nitActualNet(3),
            nitOtherNet(4),
            bat(5),
            sdtActualTS(6),
            sdtOtherTS(7),
            eitPFActualTS(8),
            eitPFOtherTS(9),
            eitScheduleActualTS(10),
            eitScheduleOtherTS(11)
           }

DelayType  ::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Delay type. "
    SYNTAX BITS
                    {
                    immediate(0),
                    synchronized(1)
           }



PsigType  ::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "  Psig type. "
    SYNTAX BITS
                    {
                    sig(0),
                    psig(1),
            psisig(2)
           }


CaDescInsMode  ::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     " Conditional Access Descriptor Insertion Type. "
    SYNTAX BITS
                    {
                    psig_insertion(0),
                    no_psig_insertion(1)
           }

FlowType  ::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     " Type of Flow: EMM, ECM, or private data. "
    SYNTAX BITS
                    {
                    ecm(0),
                    emm(1),
            private_data(2)
           }


simMIBObjects           OBJECT IDENTIFIER::= {simMIB 1}
simMIBConformance       OBJECT IDENTIFIER::= {simMIB 2}

simIdent                    OBJECT IDENTIFIER::= {simMIBObjects 1}
simECMG                     OBJECT IDENTIFIER::= {simMIBObjects 2}
simEMMG                     OBJECT IDENTIFIER::= {simMIBObjects 3}
simCPSI             OBJECT IDENTIFIER::= {simMIBObjects 4}
simPSI                      OBJECT IDENTIFIER::= {simMIBObjects 5}

--
-- Ident Group - This group is used for software configuration management of all Simulcrypt
components
-- and includes the following objects:
--

simSofwareVersion OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (80))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "This contains a display string that defines the current version
    of the software for this unit. "
   ::= {simIdent 1}

simMIBVersion OBJECT-TYPE
    SYNTAX      DisplayString  (SIZE (80))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "This contains a display string that defines the current version
    of the MIB. "
   ::= {simIdent 2}

simMIBPrivateVersion OBJECT-TYPE
    SYNTAX      DisplayString  (SIZE (80))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "This contains a display string that defines the current private
    version of the MIB. "
   ::= {simIdent 3}

simAgentVersion OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (80))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "This contains a display string that defines the current version
    of the agent. "
   ::= {simIdent 4}


--
-- ECM Generator Group - This group is used for configuration management and status monitoring of
-- ECM Generators. It identifies each one of the ECM Generators by the IP Address and TCP/UDP Port
-- Number. It also associates  Super_CAS_ids, ECM_channel_ids, and ECM_stream_ids with ECM
-- Generators. It also associates status information and statistics with channels and streams. The
ECM
-- Generator Group consists of three conceptual tables. The first table is the interconnection table
and is
-- used for the Head-end Network Manager to query the IP addresses and the port number to be used by
an
-- SCS to create a channel. It is indexed by a unique EcmgIndex which is an integer assigned by the
-- ECMG agent:
--
simEcmgTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEcmgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  This table specifies the IP addresses and Port numbers of ECM Generators
    to be used by head-end managers to configure SCSs. This table is to be
    used in ECM Generators and ECM Generator proxies. "
   ::= {simECMG 1}

simEcmgEntry OBJECT-TYPE
    SYNTAX      SimEcmgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  Information about a single table entry. Depending on whether
    this is an ECMG agent or ECMG proxy agent different table entries can
    be omitted. "
    INDEX       {simEcmgIndex}
   ::= {simEcmgTable 1}

SimEcmgEntry::= SEQUENCE {
    simEcmgIndex            INTEGER (0..65535),
    simEcmgIpAddress        IpAddress,
    simEcmgTcpPort          INTEGER (0..65535),
    simEcmgSuCasId      ECMGSuCasId,
    simEcmgChannels     Counter32,
    simEcmgCwPrs            Counter32,
    simEcmgErrs         Counter32,
    simEcmgTargetCpsig      INTEGER (0..65535),
    simEcmgCaMib            OBJECT IDENTIFIER
}


simEcmgIndex OBJECT-TYPE
    SYNTAX              INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The ECM Generator Table unique index. "
   ::= {simEcmgEntry 1}

simEcmgIpAddress OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  IP address of the host of the ECMG. "
   ::= {simEcmgEntry 2}

simEcmgTcpPort  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  TCP port of the ECMG. "
   ::= {simEcmgEntry 3}

simEcmgSuCasId  OBJECT-TYPE
    SYNTAX  ECMGSuCasId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The Super_VAS_id is formed by concatenation of the CA_system_id
    (16 bit) and the CA_subsystem_id (16 bit). It defines uniquely a
    set of ECMGs for a given SCS. "
   ::= {simEcmgEntry 4}

simEcmgChannels OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of channels this ECMG is currently maintaining. "
   ::= {simEcmgEntry 5}

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

```
     "  The total number of CW provisioning requests received by this ECMG. "
   ::= {simEcmgEntry 6}

simEcmgErrs OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors for this ECMG. "
   ::= {simEcmgEntry 7}


simEcmgTargetCpsig  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The index into the C(P)SIG table identifying the C(P)SIG associated with this ECMG. "
   ::= {simEcmgEntry 8}

simEcmgCaMib    OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The pointer to a provider proprietary MIB (like ifSpecific in
    the interfaces group of MIB II. "
   ::= {simEcmgEntry 9}

--
-- ECMG Channel Table - Used for monitoring channel information. It is indexed
-- by the ECMG Index from the ECMG table and the ChannelId.
--

simEcmgCTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEcmgCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  This table specifies information relating to ECMG/SCS channels including
    the IP addresses and Port numbers of SCSs communicating
    with the ECMG Generators.  "
   ::= {simECMG 2}

simEcmgCEntry OBJECT-TYPE
    SYNTAX      SimEcmgCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  Information about a single table entry. Depending on whether
    this is an ECMG agent or ECMG proxy agent different table entries can
    be omitted. "
    INDEX       {simEcmgIndex, simEcmgChannelId}
   ::= {simEcmgCTable 1}

SimEcmgCEntry::= SEQUENCE {
    simEcmgChannelId        ECMGChannelId,
    simEcmgCScsIpAddress        IpAddress,
    simEcmgCScsTcpPort      INTEGER (0..65535),
    simEcmgCStreams         Counter32,
    simEcmgCCwPrs           Counter32,
    simEcmgCErrs            Counter32
    }


simEcmgChannelId OBJECT-TYPE
    SYNTAX              ECMGChannelId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The ECMG/SCS Channel identifier. "
   ::= {simEcmgCEntry 1}

simEcmgCScsIpAddress OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  IP address of the SCS. "
   ::= {simEcmgCEntry 2}

simEcmgCScsTcpPort  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  TCP port of the SCS. "
   ::= {simEcmgCEntry 3}

simEcmgCStreams OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of streams this ECMG is currently maintaining on this channel. "
   ::= {simEcmgCEntry 4}

simEcmgCCwPrs   OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of CW provisioning requests received by this ECMG on this channel. "
   ::= {simEcmgCEntry 5}

simEcmgCErrs    OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors for this ECMG on this channel. "
   ::= {simEcmgCEntry 6}


--
--  ECMG Stream Table - Used for monitoring stream information. It is indexed
--  by the ECMG Index from the ECMG table, the ChannelId from the Channel Table
-- and the StreamId.
--

simEcmgSTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEcmgSEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  This table specifies information relating to ECMG/SCS streams.  "
   ::= {simECMG 3}

simEcmgSEntry OBJECT-TYPE
    SYNTAX      SimEcmgSEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "  Information about a single table  entry. Depending on whether
    this is an ECMG agent or ECMG proxy agent different table entries can
    be omitted. "
    INDEX       {simEcmgIndex, simEcmgChannelId, simEcmgStreamId}
   ::= {simEcmgSTable 1}

SimEcmgSEntry::= SEQUENCE {
    simEcmgStreamId     ECMGStreamId,
    simEcmgEcmId        ECMGStreamId,
    simEcmgSLastCp      Unsigned32,
    simEcmgSCwPrs           Counter32,
    simEcmgSErrs            Counter32
    }


simEcmgStreamId OBJECT-TYPE
    SYNTAX              ECMGStreamId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The ECMG/SCS Stream identifier. "
   ::= {simEcmgSEntry 1}

simEcmgEcmId OBJECT-TYPE
    SYNTAX              ECMGStreamId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The unique ECM flow identifier. "
   ::= {simEcmgSEntry 2}

simEcmgSLastCp  OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The number of the crypto period last processed on this stream. "
   ::= {simEcmgSEntry 3}

simEcmgSCwPrs   OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of CW provisioning requests received by this ECMG on this stream. "
   ::= {simEcmgSEntry 4}

simEcmgSErrs    OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors for this ECMG on this stream. "
   ::= {simEcmgSEntry 5}

--
--  EMM/PD Generator Group - This group is used for management of EMM/PD Generators. It identifies
--  each one of the EMM/PD Generators by the IP Address and TCP/UDP Port Number. It also associates
--  client_ids, data_stream_ids, and data_channel_ids  with EMM/PD Generators. It also associates
--  status information and statistics with  streams. The EMMG/PDG Generator  Group consists of  four
--  conceptual tables. The first table is used for information relevant to EMMG/PDG and is indexed
by a
-- unique EmOrPdIndex which is assigned by the EMMG/PDG agent:
--

simEmOrPdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEmOrPdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table defines the EMMG or PDG interfaces to the Mux and is to be
    used in EMMGs/PDGs and optionally the multiplexer. "
   ::= {simEMMG 1}

simEmOrPdEntry OBJECT-TYPE
    SYNTAX      SimEmOrPdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry. Depending on whether
    this is an EMMG/PDG or multiplexer agent different table entries can
    be omitted. "
    INDEX       {simEmOrPdIndex}
   ::= {simEmOrPdTable 1}

SimEmOrPdEntry::= SEQUENCE {
            simEmOrPdIndex      INTEGER (0..65535),
            simEmOrPdDataType       EMMGDataType,
            simEmOrPdClientId       EMMGClientId,
            simEmOrPdCommCapability EMMGCommCapability,
            simEmOrPdErrs           Counter32,
            simEmOrPdTargetCpsig        INTEGER (0..65535),
            simEmOrPdCaMib      OBJECT IDENTIFIER
            }

simEmOrPdIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Unique index into the EMMG or PDG table.  "
   ::= {simEmOrPdEntry 1}

simEmOrPdDataType   OBJECT-TYPE
    SYNTAX  EMMGDataType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Data_type: Type of data handled by this EMMG/PDG.  "
   ::= {simEmOrPdEntry 2}

simEmOrPdClientId   OBJECT-TYPE
    SYNTAX  EMMGClientId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
     "  Client_id: The client_id is a 32-bit identifier. It shall
    identify uniquely an EMMG/PDG across all the EMMGs/PDGs
    connected to a given MUX. To facilitate uniqueness of this
    value, the following rules apply:
    * In the case of EMMs or other CA related data, the two first
    bytes of the client_id should be equal to the two bytes of
    the corresponding CA_system_id.
    * In other cases a value allocated by DVB for this purpose
    should be used. "
   ::= {simEmOrPdEntry 3}

simEmOrPdCommCapability OBJECT-TYPE
    SYNTAX  EMMGCommCapability
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Communication capability between EMMG/PDG and the multiplexer. Currently
    TCP or UDP or both. "
   ::= {simEmOrPdEntry 4}


simEmOrPdErrs   OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors for this EMMG/PDG. "
   ::= {simEmOrPdEntry 5}


simEmOrPdTargetCpsig    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The index into the C(P)SIG table identifying the C(P)SIG associated with this EMMG/PDG. "
   ::= {simEmOrPdEntry 6}


simEmOrPdCaMib  OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Pointer to a vendor proprietary extension to the EMMG/PDG MIB group. "
   ::= {simEmOrPdEntry 7}



--
-- EMMG/PDG Logical Access Point Table - The second EMM Generator/ PD Genartor table is used for
-- configuration of the EMMGs/PDGs. It is uniquely indexed by the EmOrPdLapIndex which is a
-- globally  assigned quantity (with respect to the head-end) and associates globally assigned
Logical
-- Access Points (LAPs) with mux ports.
--

simEmOrPdLapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEmOrPdLapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for configuration of EMM/PD Generators. "
   ::= {simEMMG 2}

simEmOrPdLapEntry OBJECT-TYPE
    SYNTAX      SimEmOrPdLapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry. "
    INDEX       {simEmOrPdLapIndex}
   ::= {simEmOrPdLapTable 1}

SimEmOrPdLapEntry::= SEQUENCE {
            simEmOrPdLapIndex       INTEGER (0..65535),
            simEmOrPdLapAdminState  AdministrativeState,
            simEmOrPdLapCommType    EMMGCommType,
            simEmOrPdLapMuxIpAddress    IpAddress,
            simEmOrPdLapMuxPort     INTEGER (0..65535),
            simEmOrPdLapStatus      RowStatus
            }

simEmOrPdLapIndex OBJECT-TYPE
```

```
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Unique Lpgical Access Point (LAP) identifier.  "
   ::= {simEmOrPdLapEntry 1}

simEmOrPdLapAdminState  OBJECT-TYPE
    SYNTAX AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used by an authorized manager to lock a conceptual row for exclusive
    write and create access. "
   ::= {simEmOrPdLapEntry 2}

simEmOrPdLapCommType    OBJECT-TYPE
    SYNTAX  EMMGCommType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Type of communication between EMMG/PDG and the multiplexer. Currently
    TCP or UDP. "
   ::= {simEmOrPdLapEntry 3}

simEmOrPdLapMuxIpAddress    OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  IP address of the multiplexer for EMMG/PDG communication. "
   ::= {simEmOrPdLapEntry  4}

simEmOrPdLapMuxPort OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Port number (TCP/UDP) of the multiplexer for EMMG/PDG communication. "
   ::= {simEmOrPdLapEntry 5}

simEmOrPdLapStatus  OBJECT-TYPE
    SYNTAX  RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used for table row creation management. "
   ::= {simEmOrPdLapEntry 6}

--
-- EMMG/PDG Logical Access Point Group Table - The third EMM Generator/ PD Genartor table is used
-- for
-- configuration of  the EMMGs/PDG. It associates LAP Groups and LAPs and is
-- uniquely indexed by the EmOrPdLapGroup, and EmOrPdLapIndex.
--

simEmOrPdLapGTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEmOrPdLapGEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for configuration of EMM/PD Generators. "
   ::= {simEMMG 3}

simEmOrPdLapGEntry OBJECT-TYPE
    SYNTAX      SimEmOrPdLapGEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry. "
    INDEX       {simEmOrPdLapGroup, simEmOrPdLapIndex}
   ::= {simEmOrPdLapGTable 1}

SimEmOrPdLapGEntry::= SEQUENCE {
            simEmOrPdLapGroup       INTEGER (0..65535),
            simEmOrPdLapGAdminState AdministrativeState,
            simEmOrPdLapGStatus     RowStatus
            }

simEmOrPdLapGroup OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Logical Access Point (LAP) group.  "
   ::= {simEmOrPdLapGEntry 1}

simEmOrPdLapGAdminState     OBJECT-TYPE
    SYNTAX AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used by an authorized manager to lock a conceptual row for exclusive
    write and create access. "
   ::= {simEmOrPdLapGEntry 2}

simEmOrPdLapGStatus OBJECT-TYPE
    SYNTAX  RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used for table row creation management. "
   ::= {simEmOrPdLapGEntry 3}

--
-- EMMG/PDG Channel Table - Used for monitoring of EMM / PD Generator channels.
--

simEmOrPdCTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEmOrPdCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for monitoring of channels between Muxes and EMMGs/PDGs. "
   ::= {simEMMG 4}

simEmOrPdCEntry OBJECT-TYPE
    SYNTAX      SimEmOrPdCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry. "
    INDEX       {simEmOrPdIndex, simEmOrPdLapIndex, simEmOrPdChannelId}
   ::= {simEmOrPdCTable 1}

SimEmOrPdCEntry::= SEQUENCE {
    simEmOrPdChannelId      EMMGChannelId,
    simEmOrPdCommType       EMMGCommType,
    simEmOrPdCIpAddress     IpAddress,
    simEmOrPdCPort      INTEGER (0..65535),
    simEmOrPdCErrs      Counter32
    }

simEmOrPdChannelId OBJECT-TYPE
    SYNTAX      EMMGChannelId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Data_channel_id: This identifier uniquely identifies a
    EMM/Private Data channel within a client_id. "
   ::= {simEmOrPdCEntry 1}


simEmOrPdCommType OBJECT-TYPE
    SYNTAX      EMMGCommType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Communications type: TCP or UDP. "
   ::= {simEmOrPdCEntry 2}

simEmOrPdCIpAddress OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  IP address of the host of the EMMG or PDG. "
   ::= {simEmOrPdCEntry 3}

simEmOrPdCPort  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Port number *TCP or UDP of the EMMG or PDG. "
   ::= {simEmOrPdCEntry 4}
```

```
simEmOrPdCErrs  OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors on this channel. "
   ::= {simEmOrPdCEntry 5}


--
-- EMMG/PDG Stream Table - Used for monitoring of EMM / PD Generator streams.
--

simEmOrPdSTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimEmOrPdSEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for monitoring of streams between Muxes and EMMGs/PDGs. "
   ::= {simEMMG 5}

simEmOrPdSEntry OBJECT-TYPE
    SYNTAX      SimEmOrPdSEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry. "
    INDEX       {simEmOrPdIndex, simEmOrPdLapIndex, simEmOrPdDataId
}
   ::= {simEmOrPdSTable 1}

SimEmOrPdSEntry::= SEQUENCE {
    simEmOrPdDataId     EMMGStreamId,
    simEmOrPdSChannelId     EMMGChannelId,
    simEmOrPdBwidth     Unsigned32,
    simEmOrPdStreamId       EMMGStreamId,
    simEmOrPdSErrs      Counter32,
    simEmOrPdSBytes     Counter32
    }

simEmOrPdDataId OBJECT-TYPE
    SYNTAX  EMMGStreamId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  DataID: This identifier uniquely identifies a EMM/Private
    Data stream. "
   ::= {simEmOrPdSEntry 1}

simEmOrPdSChannelId OBJECT-TYPE
    SYNTAX  EMMGChannelId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Channel identifier. "
   ::= {simEmOrPdSEntry 2}

simEmOrPdBwidth OBJECT-TYPE
    SYNTAX  Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Bandwidth. "
   ::= {simEmOrPdSEntry 3}

simEmOrPdStreamId   OBJECT-TYPE
    SYNTAX  EMMGStreamId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Data_stream_id: This identifier uniquely identifies a EMM/Private
    Data stream within a channel. "
   ::= {simEmOrPdSEntry 4}

simEmOrPdSErrs  OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors on this stream. "
   ::= {simEmOrPdSEntry 5}
```

```
simEmOrPdSBytes OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of bytes sent by this EMMG/PDG on this stream. "
   ::= {simEmOrPdSEntry 6}

--
--  C(P)SIG) Group - This Group is used for management of some aspects of inteaction between the
--  custom PSI Generators (C(P)SIG)) and the PSI Generator. It consists of three tables. The
--  first table is used for advertising C(P)SIG) information by the C(P)SIG) host. The
--  second table is used for the manager to configure the C(P)SIG). The third  and fourth
--  table are used for C(P)SIG) channel and stream monitoring.
--


simCpsigTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimCpsigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table defines the C(P)SIG) interfaces to the Mux and is to be
    used in the C(P)SIG) and optionally the multiplexer. "
   ::= {simCPSI 1}

simCpsigEntry OBJECT-TYPE
    SYNTAX      SimCpsigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry. Depending on whether
    this is an EMMG/PDG or multiplexer agent different table entries can
    be omitted. "
    INDEX       {simCpsigIndex}
   ::= {simCpsigTable 1}

SimCpsigEntry::= SEQUENCE {
            simCpsigIndex           INTEGER (0..65535),
            simCpsigSuperCasId      ECMGSuCasId,
            simCpsigErrs            Counter32,
            simCpsigChannels        Counter32,
            simCpsigCpsigIpAddress      IpAddress,
            simCpsigCpsigPort       INTEGER(0..65535),
            simCpsigCaMib           OBJECT IDENTIFIER
            }

simCpsigIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Unique index into the EMMG or PDG table.  "
   ::= {simCpsigEntry 1}

simCpsigSuperCasId  OBJECT-TYPE
    SYNTAX  ECMGSuCasId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  super_CAS_id   "
   ::= {simCpsigEntry 2}

simCpsigErrs    OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors for this C(P)SIG). "
   ::= {simCpsigEntry 3}

simCpsigChannels    OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of channels for this C(P)SIG). "
   ::= {simCpsigEntry 4}

simCpsigCpsigIpAddress  OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The IP Address of the  C(P)SIG). "
   ::= {simCpsigEntry 5}

simCpsigCpsigPort   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The TCP port number of the C(P)SIG). "
   ::= {simCpsigEntry 6}

simCpsigCaMib   OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     " Pointer to a vendor proprietary extension to the C(P)SIG) MIB group. "
   ::= {simCpsigEntry 7}
--
-- C(P)SIG) Channel Table - Used for monitoring of C(P)SIG  channels.
--

simCpsigCTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimCpsigCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for monitoring of channels between (P)SIG)s and C(P)SIGs. "
   ::= {simCPSI 2}

simCpsigCEntry OBJECT-TYPE
    SYNTAX      SimCpsigCEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     " Information about a single table  entry. "
    INDEX   {simCpsigIndex, simCpsigChannelId}
   ::= {simCpsigCTable 1}

SimCpsigCEntry::= SEQUENCE {
    simCpsigChannelId       INTEGER (0..65535),
    simCpsigPsigIpAddress       IpAddress,
    simCpsigPsigPort        INTEGER (0..65535),
    simCpsigCErrs           Counter32,
    simCpsigCTstrms     Counter32,
    simCpsigCSstrms     Counter32
    }

simCpsigChannelId OBJECT-TYPE
    SYNTAX      INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Channel id identifies the C(P)SI channel. "
   ::= {simCpsigCEntry 1}

simCpsigPsigIpAddress OBJECT-TYPE
    SYNTAX  IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  IP address of the host of the (P)SIG). "
   ::= {simCpsigCEntry 2}

simCpsigPsigPort    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  TCP Port number of the (P)SIG). "
   ::= {simCpsigCEntry 3}

simCpsigCErrs   OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors on this channel. "
   ::= {simCpsigCEntry 4}

simCpsigCTstrms OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of transport streams on this channel. "
   ::= {simCpsigCEntry 5}

simCpsigCSstrms OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of session on this streamschannel. "
   ::= {simCpsigCEntry 6}


--
-- C(P)SIG Stream Table - Used for monitoring of C(P)SIG) streams.
--

simCpsigStreamTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimCpsigStreamEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is used for monitoring of streams between Muxes and C(P)SIG)s. "
   ::= {simCPSI 3}

simCpsigStreamEntry OBJECT-TYPE
    SYNTAX      SimCpsigStreamEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry. "
    INDEX       {simCpsigIndex, simCpsigChannelId, simCpsigStreamId}
   ::= {simCpsigStreamTable 1}

SimCpsigStreamEntry::= SEQUENCE {
    simCpsigStreamId    INTEGER (0..65535),
    simCpsigStreamTStreamId INTEGER (0..65535),
    simCpsigStreamNid       INTEGER (0..65535),
    simCpsigStreamOnid      INTEGER (0..65535),
    simCpsigStreamMaxCompTime   INTEGER(0..65535),
    simCpsigStreamTriggerEnable TriggerType,
    simCpsigStreamLastTrigger   TriggerType,
    simCpsigStreamLastEventId   INTEGER (0..65535),
    simCpsigStreamLastServiceId INTEGER (0..65535),
    simCpsigStreamLastEsId  INTEGER (0..65535),
    simCpsigStreamLastEcmPid    INTEGER (0..65535),
    simCpsigStreamErrs      Counter32,
    simCpsigStreamBytes     Counter32
    }

simCpsigStreamId    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This identifier uniquely identifies a C(P)SIG) stream "
   ::= {simCpsigStreamEntry 1}

simCpsigStreamTStreamId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  This identifier uniquely identifies a C(P)SIG transport stream "
   ::= {simCpsigStreamEntry 2}

simCpsigStreamNid   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  This identifier uniquely identifies the network ide associated with the stream "
   ::= {simCpsigStreamEntry 3}


simCpsigStreamOnid  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  This identifier uniquely identifies the original network ide associated with the stream "
   ::= {simCpsigStreamEntry 4}


simCpsigStreamMaxCompTime   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Max Computation time by the C(P)SIG. "
   ::= {simCpsigStreamEntry 5}


simCpsigStreamTriggerEnable OBJECT-TYPE
    SYNTAX  TriggerType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Triggers enabled by the C(P)SIG. "
   ::= {simCpsigStreamEntry 6}


simCpsigStreamLastTrigger   OBJECT-TYPE
    SYNTAX  TriggerType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Last trigger processed by the C(P)SIG. "
   ::= {simCpsigStreamEntry 7}


simCpsigStreamLastEventId   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Last event id processed by the C(P)SIG. "
   ::= {simCpsigStreamEntry 8}

simCpsigStreamLastServiceId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Last service id processed by the C(P)SIG. "
   ::= {simCpsigStreamEntry 9}


simCpsigStreamLastEsId  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Last elementary stream id processed by the C(P)SIG. "
   ::= {simCpsigStreamEntry 10}


simCpsigStreamLastEcmPid    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Last ECM pid processed by the C(P)SIG. "
   ::= {simCpsigStreamEntry 11}

simCpsigStreamErrs  OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of communications errors on this stream. "
   ::= {simCpsigStreamEntry 12}

simCpsigStreamBytes OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The total number of bytes sent by this EMMG/PDG on this stream. "
   ::= {simCpsigStreamEntry 13}


--
--  (P)SIG Group - This Group is used for the synchronization and information
--  exchange between the PSI Generator and Custom PSI Generators and
--  between the SI Generator and Custom SI Generators.
--

simPsigTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table advertises the (P)SIG configuration information.  "
   ::= {simPSI 1}

simPsigEntry OBJECT-TYPE
    SYNTAX      SimPsigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry.  "
    INDEX {simPsigIndex}
   ::= {simPsigTable 1}

SimPsigEntry::= SEQUENCE {
            simPsigIndex        INTEGER (0..65535),
            simPsigType     PsigType,
            simPsigTriggerSupport   TriggerType,
            simPsigNetworkId        INTEGER (0..65535),
            simPsigONetworkId       INTEGER (0..65535),
            simPsigTransStreamId        INTEGER (0..65535),
            simPsigTSServices   OCTET STRING (SIZE(0..511))
            }


simPsigIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The unique index into the table. "
   ::= {simPsigEntry 1}

simPsigType     OBJECT-TYPE
    SYNTAX PsigType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Psig type "
   ::= {simPsigEntry 2}


simPsigTriggerSupport  OBJECT-TYPE
    SYNTAX   TriggerType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Identifies which trigger types the PSIG supports. "
   ::= {simPsigEntry 3 }


simPsigNetworkId    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Network identifier.  "
   ::= {simPsigEntry 4}

simPsigONetworkId   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Original Network identifier.  "
   ::= {simPsigEntry 5}


simPsigTransStreamId    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Transport Stream identifier.  "
   ::= {simPsigEntry 6}
```

```
simPsigTSServices   OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..511))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  List of service identifies on the transport stream.  "
   ::= {simPsigEntry 7}

--
--

simPsigConfigTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigConfigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table configures the (P)SIG/(C)PSIG communication.  "
   ::= {simPSI 2}

simPsigConfigEntry OBJECT-TYPE
    SYNTAX      SimPsigConfigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry.  "
    INDEX {simPsigConfigCustCasId, simPsigConfigIndex , simPsigIndex}
   ::= {simPsigConfigTable 1}

SimPsigConfigEntry::= SEQUENCE {
            simPsigConfigIndex      INTEGER (0..65535),
            simPsigConfigAdminState     AdministrativeState,
            simPsigConfigCpsigType      PsigType,
            simPsigConfigCustCasId      ECMGSuCasId,
            simPsigConfigMaxCompTime    INTEGER(0..65535),
            simPsigConfigServiceId      INTEGER (0..65535),
            simPsigConfigTriggerEnable      TriggerType,
            simPsigConfigCADInsMode     CaDescInsMode,
            simPsigConfigEntryStatus    RowStatus
            }


simPsigConfigIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  The unique index into the table. "
   ::= {simPsigConfigEntry 1}

simPsigConfigAdminState     OBJECT-TYPE
    SYNTAX AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used by an authorized manager to lock a conceptual row for exclusive
    write and create access. "
   ::= {simPsigConfigEntry 2}

simPsigConfigCpsigType  OBJECT-TYPE
    SYNTAX PsigType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
    " C(P)SIG type. "
   ::= {simPsigConfigEntry 3}

simPsigConfigCustCasId  OBJECT-TYPE
    SYNTAX  ECMGSuCasId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Custom CAS Identifier. "
   ::= {simPsigConfigEntry 4}

simPsigConfigMaxCompTime OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Maximum Computing Time.  "
   ::= {simPsigConfigEntry 5}

simPsigConfigServiceId  OBJECT-TYPE
```

```
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Service identifier.  "
   ::= {simPsigConfigEntry 6}

simPsigConfigTriggerEnable  OBJECT-TYPE
    SYNTAX  TriggerType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Trigger types enabled.  "
   ::= {simPsigConfigEntry 7}

simPsigConfigCADInsMode OBJECT-TYPE
    SYNTAX  CaDescInsMode
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Conditional Access Descriptor Insert mode.  "
   ::= {simPsigConfigEntry 8}


simPsigConfigEntryStatus    OBJECT-TYPE
    SYNTAX  RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used for table row creation management. "
   ::= {simPsigConfigEntry 9}

--
--

simPsigEcmTrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigEcmTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table contains all the active ECM Triggers. "
   ::= {simPSI 3}

simPsigEcmTrEntry OBJECT-TYPE
    SYNTAX      SimPsigEcmTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry.  "
    INDEX       {simPsigEcmTrIndex}
   ::= {simPsigEcmTrTable 1}

SimPsigEcmTrEntry::= SEQUENCE {
            simPsigEcmTrIndex       INTEGER (0..65535),
            simPsigEcmTrNetworkId       INTEGER (0..65535),
            simPsigEcmTrONetworkId      INTEGER (0..65535),
            simPsigEcmTrTransStreamId   INTEGER (0..65535),
            simPsigEcmTrServiceId       INTEGER (0..65535),
            simPsigEcmTrEsId        INTEGER (0..65535),
            simPsigEcmTrType        ECMTriggerType,
            simPsigEcmTrSuCasId             ECMGSuCasId,
            simPsigEcmTrEcmId       ECMGStreamId,
            simPsigEcmTrEcmPid      INTEGER (0..65535),
            simPsigEcmTrAccessCriteria  OCTET STRING (SIZE(0..127))
            }

simPsigEcmTrIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  The trigger index. "
   ::= {simPsigEcmTrEntry 1}

simPsigEcmTrNetworkId   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier. "
   ::= {simPsigEcmTrEntry 2}

simPsigEcmTrONetworkId  OBJECT-TYPE
```

```
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier. "
   ::= {simPsigEcmTrEntry 3}

simPsigEcmTrTransStreamId   OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier.  "
   ::= {simPsigEcmTrEntry 4}

simPsigEcmTrServiceId   OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Service Identifier.  "
   ::= {simPsigEcmTrEntry 5}

simPsigEcmTrEsId    OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Elementary Stream Identifier.  "
   ::= {simPsigEcmTrEntry 6}

simPsigEcmTrType OBJECT-TYPE
    SYNTAX  ECMTriggerType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  ECM Trigger Type.  "
   ::= {simPsigEcmTrEntry 7}

simPsigEcmTrSuCasId OBJECT-TYPE
    SYNTAX  ECMGSuCasId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  ECM Client Identifier.  "
   ::= {simPsigEcmTrEntry 8}

simPsigEcmTrEcmId OBJECT-TYPE
    SYNTAX  ECMGStreamId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  ECM Stream Identifier.  "
   ::= {simPsigEcmTrEntry 9}

simPsigEcmTrEcmPid OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " ECM PID  "
   ::= {simPsigEcmTrEntry 10}

simPsigEcmTrAccessCriteria  OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..127))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Access criteria.  "
   ::= {simPsigEcmTrEntry 11}

--
--

simPsigFlowTrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigFlowTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table contains all the active Flow Triggers. "
   ::= {simPSI 4}

simPsigFlowTrEntry OBJECT-TYPE
```

```
    SYNTAX      SimPsigFlowTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry.  "
    INDEX       {simPsigFlowTrIndex}
   ::= {simPsigFlowTrTable 1}

SimPsigFlowTrEntry::= SEQUENCE {
            simPsigFlowTrIndex      INTEGER (0..65535),
            simPsigFlowTrType       FlowType,
            simPsigFlowTrSuCasId             FlowSuCasId,
            simPsigFlowTrFlowId     FlowId,
            simPsigFlowTrFlowPID        INTEGER (0..65535)
            }

simPsigFlowTrIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The trigger index. "
   ::= {simPsigFlowTrEntry 1}

simPsigFlowTrType OBJECT-TYPE
    SYNTAX  FlowType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Flow Type.  "
   ::= {simPsigFlowTrEntry 2}

simPsigFlowTrSuCasId OBJECT-TYPE
    SYNTAX  FlowSuCasId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Flow Super CAS identifier.  "
   ::= {simPsigFlowTrEntry 3}

simPsigFlowTrFlowId OBJECT-TYPE
    SYNTAX  FlowId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Flow Stream Identifier.  "
   ::= {simPsigFlowTrEntry 4}

simPsigFlowTrFlowPID OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Flow PID  "
   ::= {simPsigFlowTrEntry 5}

--
--

simPsigEvntTrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigEvntTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table contains all the active EVNT Triggers. "
   ::= {simPSI 5}

simPsigEvntTrEntry OBJECT-TYPE
    SYNTAX      SimPsigEvntTrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry.  "
    INDEX       {simPsigEvntTrIndex}
   ::= {simPsigEvntTrTable 1}

SimPsigEvntTrEntry::= SEQUENCE {
            simPsigEvntTrIndex      INTEGER (0..65535),
            simPsigEvntTrNetworkId      INTEGER (0..65535),
            simPsigEvntTrONetworkId     INTEGER (0..65535),
            simPsigEvntTrTransStreamId  INTEGER (0..65535),
            simPsigEvntTrServiceId      INTEGER (0..65535),
            simPsigEvntTrEventId        INTEGER (0..65535),
            simPsigEvntTrStartTime      OCTET STRING(SIZE (0..4)),
            simPsigEvntTrDuration       OCTET STRING(SIZE (0..2)),
            simPsigEvntTrPrivateData        OCTET STRING(SIZE (0..256))
            }


simPsigEvntTrIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  The event trigger index. "
   ::= {simPsigEvntTrEntry 1}

simPsigEvntTrNetworkId  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier. "
   ::= {simPsigEvntTrEntry 2}

simPsigEvntTrONetworkId     OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier. "
   ::= {simPsigEvntTrEntry 3}

simPsigEvntTrTransStreamId  OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier.  "
   ::= {simPsigEvntTrEntry 4}

simPsigEvntTrServiceId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Event trigger service identifier.  "
   ::= {simPsigEvntTrEntry 5}

simPsigEvntTrEventId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Event Identifier.  "
   ::= {simPsigEvntTrEntry 6}

simPsigEvntTrStartTime OBJECT-TYPE
    SYNTAX  OCTET STRING(SIZE (0..4))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Event start time.  "
   ::= {simPsigEvntTrEntry 7}

simPsigEvntTrDuration OBJECT-TYPE
    SYNTAX  OCTET STRING(SIZE (0..2))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Event duration.  "
   ::= {simPsigEvntTrEntry 8}

simPsigEvntTrPrivateData OBJECT-TYPE
    SYNTAX  OCTET STRING(SIZE (0..256))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  EVNT Channel Identifier.  "
   ::= {simPsigEvntTrEntry 9}

--
--

simPsigDescInsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigDescInsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table contains all the information related to descriptor insertion. "
   ::= {simPSI 6}

simPsigDescInsEntry OBJECT-TYPE
    SYNTAX      SimPsigDescInsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table  entry.  "
    INDEX       {simPsigDescInsIndex}
   ::= {simPsigDescInsTable 1}

SimPsigDescInsEntry::= SEQUENCE {
            simPsigDescInsIndex     INTEGER (0..65535),
            simPsigDescInsAdminState    AdministrativeState,
            simPsigDescInsTrIndex       INTEGER (0..65535),
            simPsigDescInsTrType        TriggerType,
            simPsigDescInsLocationId    InsertLocation,
            simPsigDescInsNetworkId     INTEGER (0..65535),
            simPsigDescInsONetworkId    INTEGER (0..65535),
            simPsigDescInsTransStreamId INTEGER (0..65535),
            simPsigDescInsServiceId     INTEGER (0..65535),
            simPsigDescInsElmStreamId   INTEGER (0..65535),
            simPsigDescInsBouquetId     INTEGER (0..65535),
            simPsigDescInsEventId       INTEGER (0..65535),
            simPsigDescInsONetworkId2loop   INTEGER (0..65535),
            simPsigDescInsNetworkIdOther    INTEGER (0..65535),
            simPsigDescInsTransStreamId2OrO INTEGER (0..65535),
            simPsigDescInsDelayType     DelayType,
            simPsigDescInsDelay     OCTET STRING (SIZE(0..1)),
            simPsigDescPrivDataSpfier   INTEGER (0..65535),
            simPsigDescInsEntryStatus   RowStatus
            }

simPsigDescInsIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The unique index into the table. "
   ::= {simPsigDescInsEntry 1}


simPsigDescInsAdminState    OBJECT-TYPE
    SYNTAX AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used by an authorized manager to lock a conceptual row for exclusive
    write and create access. "
   ::= {simPsigDescInsEntry 2}


simPsigDescInsTrIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The unique index into the corresponding trigger table. "
   ::= {simPsigDescInsEntry 3}


simPsigDescInsTrType OBJECT-TYPE
    SYNTAX   TriggerType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The type of the trigger that caused this descriptor insert. "
   ::= {simPsigDescInsEntry 4}

simPsigDescInsLocationId    OBJECT-TYPE
    SYNTAX  InsertLocation
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The type of target table for insertion. "
   ::= {simPsigDescInsEntry 5}

simPsigDescInsNetworkId     OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier. "
   ::= {simPsigDescInsEntry 6}

simPsigDescInsONetworkId    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier. "
   ::= {simPsigDescInsEntry 7}

simPsigDescInsTransStreamId OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier.  "
   ::= {simPsigDescInsEntry 8}

simPsigDescInsServiceId OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Service Identifier.  "
   ::= {simPsigDescInsEntry 9}

simPsigDescInsElmStreamId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Elementary stream identifier.  "
   ::= {simPsigDescInsEntry 10}

simPsigDescInsBouquetId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Event trigger bouquet identifier.  "
   ::= {simPsigDescInsEntry 11}

simPsigDescInsEventId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  EVNT Identifier.  "
   ::= {simPsigDescInsEntry 12}

simPsigDescInsONetworkId2loop   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier second loop. "
   ::= {simPsigDescInsEntry 13}

simPsigDescInsNetworkIdOther    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier other. "
   ::= {simPsigDescInsEntry 14}

simPsigDescInsTransStreamId2OrO OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier second loop or other.  "
   ::= {simPsigDescInsEntry 15}

simPsigDescInsDelayType OBJECT-TYPE
    SYNTAX  DelayType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

```
     " Delay type, immediate or synchronized.  "
   ::= {simPsigDescInsEntry 16}

simPsigDescInsDelay OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..1))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Insert delay "
   ::= {simPsigDescInsEntry 17}

simPsigDescPrivDataSpfier OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Private data specifier "
   ::= {simPsigDescInsEntry 18}

simPsigDescInsEntryStatus OBJECT-TYPE
    SYNTAX  RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Other Transport Stream identifiers. "
   ::= {simPsigDescInsEntry 19}

--
--

simPsigDescInsDescTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigDescInsDescEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table contains all the descriptors to be inserted. "
   ::= {simPSI 7}

simPsigDescInsDescEntry OBJECT-TYPE
    SYNTAX      SimPsigDescInsDescEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry.  "
    INDEX       {simPsigDescInsIndex, simPsigDescInsDescIndex}
   ::= {simPsigDescInsDescTable 1}

SimPsigDescInsDescEntry::= SEQUENCE {
            simPsigDescInsDescIndex     INTEGER (0..65535),
            simPsigDescInsDescAdminState    AdministrativeState,
            simPsigDescInsDescriptor    OCTET STRING (SIZE(0..8191)),
            simPsigDescInsDescriptorStatus  DescriptorStatus,
            simPsigDescInsDescEntryStatus   RowStatus
            }


simPsigDescInsDescIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The unique index into the table. "
   ::= {simPsigDescInsDescEntry 1}


simPsigDescInsDescAdminState    OBJECT-TYPE
    SYNTAX AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Used by an authorized manager to lock a conceptual row for exclusive
    write and create access. "
   ::= {simPsigDescInsDescEntry 2}

simPsigDescInsDescriptor OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..8191))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " The descriptor to be inserted. "
   ::= {simPsigDescInsDescEntry 3}

simPsigDescInsDescriptorStatus OBJECT-TYPE
```

```
    SYNTAX  DescriptorStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " The insertion status of the descriptor. "
   ::= {simPsigDescInsDescEntry 4}

simPsigDescInsDescEntryStatus OBJECT-TYPE
    SYNTAX  RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Other transport stream identifiers. "
   ::= {simPsigDescInsDescEntry 5}

--
--

simPsigTblProvTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigTblProvEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This table is the interface to obtaining all PSI/SI information. "
   ::= {simPSI 8}

simPsigTblProvEntry OBJECT-TYPE
    SYNTAX      SimPsigTblProvEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry.  "
    INDEX       {simPsigTblProvIndex}
   ::= {simPsigTblProvTable 1}

SimPsigTblProvEntry::= SEQUENCE {
            simPsigTblProvIndex     INTEGER (0..65535),
            simPsigTblProvTableId       ProvTableId,
            simPsigTblNetworkId     INTEGER (0..65535),
            simPsigTblONetworkId        INTEGER (0..65535),
            simPsigTblTransStreamId     INTEGER (0..65535),
            simPsigTblServiceId     INTEGER (0..65535),
            simPsigTblBouquetId     INTEGER (0..65535),
            simPsigTblEventId       INTEGER (0..65535),
            simPsigTblONetworkId2loop   INTEGER (0..65535),
            simPsigTblNetworkIdOther    INTEGER (0..65535),
            simPsigTblTransStreamId2OrO INTEGER (0..65535),
            simPsigTblSegmentNr     INTEGER (0..65535),
            simPsigTblProvPart      OCTET STRING (SIZE(0..8191)),
            simPsigTblProvPartNumber    INTEGER (0..65535)
            }

simPsigTblProvIndex OBJECT-TYPE
    SYNTAX   INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The unique index into the table. "
   ::= {simPsigTblProvEntry 1}

simPsigTblProvTableId   OBJECT-TYPE
    SYNTAX ProvTableId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  The table identifier of the table. "
   ::= {simPsigTblProvEntry 2}

simPsigTblNetworkId     OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier. "
   ::= {simPsigTblProvEntry 3}

simPsigTblONetworkId    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier. "
   ::= {simPsigTblProvEntry 4}
```

```
simPsigTblTransStreamId OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier.  "
   ::= {simPsigTblProvEntry 5}

simPsigTblServiceId OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Service Identifier.  "
   ::= {simPsigTblProvEntry 6}

simPsigTblBouquetId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Event trigger bouquet identifier.  "
   ::= {simPsigTblProvEntry 7}

simPsigTblEventId OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Event Identifier  "
   ::= {simPsigTblProvEntry 8}

simPsigTblONetworkId2loop   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Original Network Identifier second loop. "
   ::= {simPsigTblProvEntry 9}

simPsigTblNetworkIdOther    OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  The Network Identifier other. "
   ::= {simPsigTblProvEntry 10}

simPsigTblTransStreamId2OrO OBJECT-TYPE
    SYNTAX  INTEGER(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Transport Stream Identifier second loop or other.  "
   ::= {simPsigTblProvEntry 11}

simPsigTblSegmentNr OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Segment Number  "
   ::= {simPsigTblProvEntry 12}

simPsigTblProvPart OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..8191))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     "  Event Identifier  "
   ::= {simPsigTblProvEntry 13}

simPsigTblProvPartNumber OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
     " Each table is subdivided into parts for SNMP transport if necessary.
      The part number identifies the table part of this entry.  "
   ::= {simPsigTblProvEntry 14}

--
```

```
--

simPsigPIDProvTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SimPsigPIDProvEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  This is the PID provisioning table. "
   ::= {simPSI 9}

simPsigPIDProvEntry OBJECT-TYPE
    SYNTAX      SimPsigPIDProvEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
     "  Information about a single table entry.  "
    INDEX       {simPsigPIDProvSuCasId, simPsigPIDProvFlowId}
   ::= {simPsigPIDProvTable 1}

SimPsigPIDProvEntry::= SEQUENCE {
            simPsigPIDProvFlowType      FlowType,
            simPsigPIDProvSuCasId             FlowSuCasId,
            simPsigPIDProvFlowId        FlowId,
            simPsigPIDProvFlowPID       INTEGER (0..65535)
            }

simPsigPIDProvFlowType OBJECT-TYPE
    SYNTAX  FlowType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Flow Type.  "
   ::= {simPsigPIDProvEntry 1}

simPsigPIDProvSuCasId OBJECT-TYPE
    SYNTAX  FlowSuCasId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Flow Super CAS identifier.  "
   ::= {simPsigPIDProvEntry 2}

simPsigPIDProvFlowId OBJECT-TYPE
    SYNTAX  FlowId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     "  Flow Stream Identifier.  "
   ::= {simPsigPIDProvEntry 3}

simPsigPIDProvFlowPID OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
     " Flow PID  "
   ::= {simPsigPIDProvEntry 4}


--
--
--  Conformance Information
--

simCompliances      OBJECT IDENTIFIER::= {simMIBConformance 1}
simGroups       OBJECT IDENTIFIER::= {simMIBConformance 2}

simEcmgCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent ECMGs "
    MODULE  -- this module
    MANDATORY-GROUPS {simIdentGroup, simEcmgGroup}
    ::= {simCompliances 1}

simEmOrPdCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent EMMGs or PDGs "
    MODULE  -- this module
    MANDATORY-GROUPS {simIdentGroup, simEmOrPdGroup}
    GROUP simEmOrPdLapGGroup
    DESCRIPTION
     " Allows for grouping of LAPs  "
    ::= {simCompliances 2}

simCpsigCompliance  MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent C(P)SIG)s "
    MODULE  -- this module
    MANDATORY-GROUPS {simIdentGroup, simCpsigGroup}
    ::= {simCompliances 3}

simPsigCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent (P)SIG)s "
    MODULE  -- this module
    MANDATORY-GROUPS {simIdentGroup, simPsigGroup}
    ::= {simCompliances 4}

simIdentGroup       OBJECT-GROUP
    OBJECTS {
        simSofwareVersion,
        simMIBVersion,
        simMIBPrivateVersion,
        simAgentVersion
    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing software configuration infomation. "
    ::= {simGroups 1}

simEcmgGroup        OBJECT-GROUP
    OBJECTS {
    simEcmgIndex,
    simEcmgIpAddress,
    simEcmgTcpPort,
    simEcmgSuCasId,
    simEcmgChannels,
    simEcmgCwPrs,
    simEcmgErrs,
    simEcmgTargetCpsig,
    simEcmgCaMib ,
    simEcmgChannelId,
    simEcmgCScsIpAddress,
    simEcmgCScsTcpPort,
    simEcmgCStreams,
    simEcmgCCwPrs,
    simEcmgCErrs,
    simEcmgStreamId,
    simEcmgEcmId,
    simEcmgSLastCp,
    simEcmgSCwPrs,
    simEcmgSErrs
    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing ECMG infomation. "
    ::= {simGroups 2}


simEmOrPdGroup      OBJECT-GROUP
    OBJECTS {
    simEmOrPdIndex,
    simEmOrPdDataType,
    simEmOrPdClientId,
    simEmOrPdCommCapability,
    simEmOrPdErrs,
    simEmOrPdTargetCpsig ,
    simEmOrPdCaMib,
    simEmOrPdLapIndex,
    simEmOrPdLapAdminState,
    simEmOrPdLapCommType,
    simEmOrPdLapMuxIpAddress,
    simEmOrPdLapMuxPort,
    simEmOrPdLapStatus,
    simEmOrPdChannelId,
    simEmOrPdCommType,
    simEmOrPdCIpAddress,
    simEmOrPdCPort,
    simEmOrPdCErrs,
    simEmOrPdDataId,
    simEmOrPdSChannelId,
    simEmOrPdBwidth,
    simEmOrPdStreamId,
    simEmOrPdSErrs,
    simEmOrPdSBytes
    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing EMMG/PDG infomation. "
    ::= {simGroups 3}

simEmOrPdLapGGroup  OBJECT-GROUP
    OBJECTS {
    simEmOrPdLapGroup,
    simEmOrPdLapGAdminState,
    simEmOrPdLapGStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing LAPG infomation. "
    ::= {simGroups 4}

simCpsigGroup   OBJECT-GROUP
    OBJECTS {
    simCpsigIndex,
    simCpsigSuperCasId,
    simCpsigErrs,
    simCpsigChannels,
    simCpsigCpsigIpAddress,
    simCpsigCpsigPort,
    simCpsigCaMib,
    simCpsigChannelId,
    simCpsigPsigIpAddress,
    simCpsigPsigPort,
    simCpsigCErrs,
    simCpsigCTstrms,
    simCpsigCSstrms,
    simCpsigStreamTStreamId,
    simCpsigStreamNid   ,
    simCpsigStreamOnid      ,
    simCpsigStreamMaxCompTime   ,
    simCpsigStreamTriggerEnable ,
    simCpsigStreamLastTrigger   ,
    simCpsigStreamLastEventId   ,
    simCpsigStreamLastServiceId ,
    simCpsigStreamLastEsId  ,
    simCpsigStreamLastEcmPid    ,
    simCpsigStreamErrs      ,
    simCpsigStreamBytes
    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing C(P)SIG) infomation. "
    ::= {simGroups 5}


simPsigGroup    OBJECT-GROUP
    OBJECTS {
        simPsigIndex,
        simPsigType,
        simPsigTriggerSupport,
        simPsigNetworkId,
        simPsigONetworkId,
        simPsigTransStreamId,
        simPsigTSServices,
        simPsigConfigAdminState,
        simPsigConfigCustCasId,
        simPsigConfigMaxCompTime,
        simPsigConfigServiceId,
        simPsigConfigTriggerEnable,
        simPsigConfigEntryStatus,
        simPsigConfigCpsigType,
        simPsigConfigCADInsMode,
        simPsigEcmTrNetworkId,
        simPsigEcmTrONetworkId,
        simPsigEcmTrTransStreamId,
        simPsigEcmTrServiceId,
        simPsigEcmTrEsId,
        simPsigEcmTrType,
        simPsigEcmTrSuCasId,
        simPsigEcmTrEcmId,
        simPsigEcmTrEcmPid,
        simPsigEcmTrAccessCriteria,
        simPsigEvntTrNetworkId,
        simPsigEvntTrONetworkId,
        simPsigEvntTrTransStreamId,
        simPsigEvntTrServiceId,
        simPsigEvntTrEventId,
        simPsigEvntTrStartTime,
        simPsigEvntTrDuration,
        simPsigEvntTrPrivateData,
        simPsigFlowTrIndex,
        simPsigFlowTrType,
        simPsigFlowTrSuCasId,
        simPsigFlowTrFlowId,
        simPsigFlowTrFlowPID,
        simPsigDescInsIndex,
        simPsigDescInsAdminState,
        simPsigDescInsTrIndex,
        simPsigDescInsTrType,
        simPsigDescInsLocationId,
        simPsigDescInsNetworkId,
        simPsigDescInsONetworkId,
        simPsigDescInsTransStreamId,
        simPsigDescInsServiceId,
        simPsigDescInsElmStreamId,
        simPsigDescInsBouquetId,
        simPsigDescInsEventId,
        simPsigDescInsNetworkIdOther,
        simPsigDescInsONetworkId2loop,
        simPsigDescInsTransStreamId2OrO,
        simPsigDescInsDelayType,
        simPsigDescInsDelay,
        simPsigDescPrivDataSpfier,
        simPsigDescInsEntryStatus,
        simPsigDescInsDescIndex,
        simPsigDescInsDescAdminState,
        simPsigDescInsDescriptor,
        simPsigDescInsDescriptorStatus,
        simPsigDescInsDescEntryStatus,
        simPsigTblProvIndex,
        simPsigTblProvTableId       ,
        simPsigTblNetworkId,
        simPsigTblONetworkId    ,
        simPsigTblTransStreamId     ,
        simPsigTblServiceId ,
        simPsigTblBouquetId,
        simPsigTblEventId   ,
        simPsigTblNetworkIdOther,
        simPsigTblONetworkId2loop,
        simPsigTblTransStreamId2OrO,
        simPsigTblSegmentNr,
        simPsigTblProvPart,
        simPsigTblProvPartNumber,
        simPsigPIDProvFlowType,
        simPsigPIDProvSuCasId,
        simPsigPIDProvFlowId,
        simPsigPIDProvFlowPID

    }
    STATUS current
    DESCRIPTION
     " A collection of objects providing (P)SIG) infomation. "
    ::= {simGroups 6}
END
```

G.2 SEM MIB

```
SEM-MIB DEFINITIONS::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Integer32, Unsigned32, BITS, IpAddress, TimeTicks,
    NOTIFICATION-TYPE           FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, RowStatus,
    DisplayString, TruthValue, DateAndTime FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP    FROM SNMPv2-CONF;


semMIB MODULE-IDENTITY
    LAST-UPDATED  "9707021700Z "
    ORGANIZATION  "DVB Simulcrypt Technical Group "
    CONTACT-INFO  " ---  "
    DESCRIPTION
         "The MIB module for defining DVB Simulcrypt Conditional
    Access System event information. "
   ::= {1 3 6 1 4 1 2696 1 2}

EntryName::= TEXTUAL-CONVENTION
    STATUS  current
    DESCRIPTION
         " Entry name convention for tables. "
    SYNTAX OCTET STRING (SIZE (8))

EventType::= TEXTUAL-CONVENTION
    STATUS  current
    DESCRIPTION
         "An event type is indicated if the bit corresponding to its power of
        two is set, i.e. if the mask is 3 then communications and
        qualityOfService event types are indicated. "
    SYNTAX BITS {
              communications(0),
              qualityOfService(1),
              processingError(2),
              equipmentAlarm(3),
              environmental(4),
               attributeValueChange(5),
               stateChange(6),
               timeDomainViolation(7),
               securitySrvcOrMechnsmViolation(8),
               relationshipChange(9),
               operationalViolation(10),
               integrityViolation(11),
               physicalViolation(12),
               thresholdCrossing(13),
               thresholdClearing(14)
        }

ProbableCause::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Defined in ITU-T Recommendation X.733 [9] "
    SYNTAX BITS {
            simulcryptSpecific(0),
            adapterError(1),
            applicationSubsystemFailure(2),
            bandwidthReduced(3),
            callEstablishmentError(4),
            communicationsProtocolError(5),
            communicationsSubsystemFailure(6),
            configurationOrCustomizationError(7),
            congestion(8),
            corruptData(9),
            cpuCyclesLimitExceeded(10),
            dataSetOrModemError(11),
            degradedSignal(12),
            dTEDCEInterfaceError(13),
            enclosureDoorOpen(14),
            equipmentMalfunction(15),
            excessiveVibration(16),
            fileError(17),
            fireDetected(18),
            floodDetected(19),
            framingError(20),
            heatOrVentOrCoolSystemProblem(21),
            humidityUnacceptable(22),
            inputOutputDeviceError(23),
            inputDeviceError(24),
            lANError(25),
            leakDetected(26),
            localNodeTransmissionError(27),
            lossOfFrame(28),
            lossOfSignal(29),
            materialSupplyExhausted(30),
            multiplexerProblem(31),
            outOfMemory(32),
            ouputDeviceError(33),
            performanceDegraded(34),
            powerProblem(35),
            pressureUnacceptable(36),
            processorProblem(37),
            pumpFailure(38),
            queueSizeExceeded(39),
            receiveFailure(40),
            receiverFailure(41),
            remoteNodeTransmissionError(42),
            resourceAtOrNearingCapacity(43),
            responseTimeExcessive(44),
            retransmissionRateExcessive(45),
            softwareError(46),
            softProgramAbnormallyTerminated(47),
            softwareProgramError(48),
            storageCapacityProblem(49),
            temperatureUnacceptable(50),
            thresholdCrossed(51),
            timingProblem(52),
            toxicLeakDetected(53),
            transmitFailure(54),
            transmitterFailure(55),
            underlyingResourceUnavailable(56),
            versionMismatch(57),
            authenticationFailure(58),
            breachOfConfidentiality(59),
            cableTamper(60),
            delayedInformation(61),
            denialOfService(62),
            duplicateInformation(63),
            informationMissing(64),
            informationModificationDetected(65),
            informationOutOfSequence(66),
            intrusionDetection(67),
            keyExpired(68),
            nonRepudiationFailure(69),
            outOfHoursActivity(70),
            outOfService(71),
            proceduralError(72),
            unauthorizedAccessAttempt(73),
            unexpectedInformation(74),
            unspecifiedReason(75)
            }


PerceivedSeverity::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
             "This convention defines six severity levels, which provide
             an indication of how it is perceived that the capability
             of the managed object has been affected. Those severity
             levels which represent service affecting conditions
             ordered from most severe to lease severe are Critical,
             Major, Minor, and Warning.

             Perceived Severity is defined in ITU-T Recommendation X.733 [9]. "
    SYNTAX BITS
                    {
                        cleared(0),
                        indeterminate(1),
                        warning(2),
                        minor(3),
                        major(4),
                        critical(5)
                   }

TrendIndication::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
         "Indicates the trend of an event as defined in ITU-T Recommendation X.733 [9]. "
    SYNTAX BITS
             {      lessSevere(0),
                        noChange(1),
                        moreSevere(2)
            }

BackedUpStatus::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
             "The backed up status as defined in ITU-T Recommendation X.733 [9] "
    SYNTAX BITS
            {
                    backedUp(0),
                        notBackedUp(1)
           }

AdministrativeState::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Administrative state as defined by ITU-T Recommendation X.734 [10]. "
    SYNTAX BITS
                    {
                    locked(0),
                    unlocked(1),
                        shuttingDown(2)
           }

OperationalState::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Operational state as defined by ITU-T Recommendation X.734 [10]. "
    SYNTAX BITS
            {
                    enabled(0),
                        disabled(1)
           }

AvailabilityStatus::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Availability Status as defined by ITU-T Recommendation X.734 [10]. "
    SYNTAX BITS
                    {
        available(0),
                        inTest(1),
                        failed(2),
                        powerOff(3),
                        offLine(4),
                        offDuty(5),
                            dependency(6),
                        degraded(7),
                        notInstalled(8),
                            logFull(9)
                   }


AlarmStatus::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Alarm Status as defined by ITU-T Recommendation X.734 [10]. "
    SYNTAX BITS
                    {
            underRepair(0),
                        critical(1),
                        major(2),
                        minor(3),
                        alarmOutstanding(4),
                        cleared(5)
                   }


EventSensitivity::=  TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
     "Is the event caused by crossing of a threshold (edge)or by exceeding a threshold
    (level). "
    SYNTAX BITS
                    {
            edgeSensitive(0),
                        levelSensitive(1)
                   }





semMIBObjects           OBJECT IDENTIFIER::= {semMIB 1}
semMIBConformance       OBJECT IDENTIFIER::= {semMIB 2}
semMIBNotificationPrefix    OBJECT IDENTIFIER::= {semMIB 3}

semEvent              OBJECT IDENTIFIER::= {semMIBObjects 1}
semEfd                OBJECT IDENTIFIER::= {semMIBObjects 2}

--
-- Event Section
--

semEventTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SemEventEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A table of management event information. "
```

```
   ::= {semEvent 1}

semEventEntry OBJECT-TYPE
    SYNTAX      SemEventEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "Information about a single management event. "
    INDEX       {semEventName}
   ::= {semEventTable 1}

SemEventEntry::= SEQUENCE {
    semEventName                EntryName,
    semEventAdminState      AdministrativeState,
    semEventAlarmStatus     AlarmStatus,
    semEventType                    EventType,
    semEventText                DisplayString,
    semEventChangedObjectId     OBJECT IDENTIFIER,
    semEventToStateChange       Unsigned32,
    semEventRisingThreshold     Integer32,
    semEventFallingThreshold    Integer32,
    semEventProbableCause       ProbableCause,
    semEventPerceivedSeverity   PerceivedSeverity,
    semEventTrendIndication    TrendIndication,
    semEventBackedUpStatus      BackedUpStatus,
    semEventBackUpObject            OBJECT IDENTIFIER,
    semEventSpecificProblems   OBJECT IDENTIFIER,
    semEventFrequency           Integer32,
    semEventSensitivity         EventSensitivity,
    semEventStatus              RowStatus
}


semEventName OBJECT-TYPE
    SYNTAX      EntryName
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The unique name of the target event. "
   ::= {semEventEntry 1}

semEventAdminState OBJECT-TYPE
    SYNTAX      AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The Administrative State of the event entry. "
   ::= {semEventEntry 2}

semEventAlarmStatus OBJECT-TYPE
    SYNTAX      AlarmStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The Alarm Status of an event. "
   ::= {semEventEntry 3}


semEventType OBJECT-TYPE
   SYNTAX      EventType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         " Indicates the type of the event. "
    ::= {semEventEntry 4}



semEventText OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "A description of the event's function and use. "
    DEFVAL {''H}
   ::= {semEventEntry 5}


semEventChangedObjectId OBJECT-TYPE
SYNTAX          OBJECT IDENTIFIER
MAX-ACCESS      read-create
    STATUS          current
    DESCRIPTION
```

```
          "The object identifier of the MIB object to check to see
         if the event should fire.

         This may be wildcarded by truncating all or part of the
         instance portion, in which case the condition is obtained
         as if with a GetNext function, checking multiple values
         if they exist. "
    ::= {semEventEntry 6}


semEventToStateChange  OBJECT-TYPE
SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "  If semEvent ChangedObjectId is a state/status/variable,
            this variable identifies the state that causes the
            event to be generated. "
   ::= {semEventEntry 7}

semEventRisingThreshold OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "  A threshold value to check against if semEventType is
            'threshold'. In this case if the value of the object at
            semEventValueID is greater than or equal to this threshold
            and the value at the last sampling interval was less than
            this threshold, one semEventRisingEvent is triggered.
            If semEventType is not 'threshold', this object is not
            instantiated. "
    DEFVAL {0}
   ::= {semEventEntry 8}

semEventFallingThreshold OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "  A threshold value to check against if semEventType is
            'threshold'. In this case if the value of the object at
            semEventValueID is less than or equal to this threshold
            and the value at the last sampling interval was greater than
            this threshold, one semEventFallingEvent is triggered.
            If semEventType is not 'threshold', this object is not
            instantiated. "
    DEFVAL {0}
   ::= {semEventEntry 9}

semEventProbableCause OBJECT-TYPE
    SYNTAX ProbableCause
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "This variable defines further probable cause for
         the last event of this type. "
   ::= {semEventEntry 10}

semEventPerceivedSeverity  OBJECT-TYPE
    SYNTAX      PerceivedSeverity
    MAX-ACCESS  read-only
STATUS      current
    DESCRIPTION
             "This parameter defines the pereceived severity of the
         last event of this type. "
   ::= {semEventEntry 11}

semEventTrendIndication   OBJECT-TYPE
    SYNTAX  TrendIndication
    MAX-ACCESS  read-only
    STATUS      current
        DESCRIPTION
             "Indicates the trend of the last event of this type. "
   ::= {semEventEntry 12}

semEventBackedUpStatus  OBJECT-TYPE
    SYNTAX      BackedUpStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "The backed up status. "
   ::= {semEventEntry 13}
```

```
semEventBackUpObject  OBJECT-TYPE
SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "  If the backed up status is backedUp then this variable
            ontains the object identifier of the object containing
            back up object. "
   ::= {semEventEntry 14}

semEventSpecificProblems       OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         " This variable identifies the object responsible for the event.  "
   ::= {semEventEntry 15}

semEventFrequency OBJECT-TYPE
    SYNTAX      Integer32 (1..65535)
    UNITS        "seconds "
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "  The number of seconds to wait between event condition
            checks. To encourage consistency in sampling, the
            interval is measured from the beginning of one check to
            the beginning of the next. "
    DEFVAL {600}
   ::= {semEventEntry 16}

semEventSensitivity OBJECT-TYPE
    SYNTAX      EventSensitivity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The event sensitivity which identifies whether the event is level or edge
    sensitive. "
   ::= {semEventEntry 17}

semEventStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The control that allows creation/deletion of entries. "
   ::= {semEventEntry 18}


--
-- Event Forwarding Discriminator (EFD) Status  Section
--

semEfdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SemEfdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A table of management EFDs. "
   ::= {semEfd 1}

semEfdEntry OBJECT-TYPE
    SYNTAX      SemEfdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "Information about a single EFD. "
    INDEX       {semEfdName , semEfdTarget}
   ::= {semEfdTable 1}

SemEfdEntry::= SEQUENCE {
            semEfdName                  EntryName,
        semEfdAdminState        AdministrativeState,
        semEfdOperState         OperationalState,
        semEfdAvailStatus       AvailabilityStatus,
        semEfdStartTime         DateAndTime,
        semEfdStopTime          DateAndTime,
        semEfdDailyStartTime        TimeTicks,
        semEfdDailyStopTime         TimeTicks,
        semEfdWeeklyMask        OCTET STRING,
        semEfdTypes         EventType,
        semEfdCause             ProbableCause,
        semEfdSeverity          PerceivedSeverity,
        semEfdSpecificProblems      OBJECT IDENTIFIER,
        semEfdTrendIndication       TrendIndication,
        semEfdChangedObjectId       OBJECT IDENTIFIER,
        semEfdToStateChange         Unsigned32,
            semEfdNotification          OBJECT IDENTIFIER,
        semEfdOr            TruthValue,
        semEfdTarget                IpAddress,
        semEfdText                  DisplayString,
        semEfdStatus            RowStatus
   }

semEfdName OBJECT-TYPE
    SYNTAX      EntryName
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The unique name of the EFD. "
   ::= {semEfdEntry 1}

semEfdAdminState  OBJECT-TYPE
    SYNTAX  AdministrativeState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  The current Adminsitrative state of the EFD. "
   ::= {semEfdEntry 2}

semEfdOperState  OBJECT-TYPE
SYNTAX OperationalState
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  The current Operational state of the EFD. "
   ::= {semEfdEntry 3}


semEfdAvailStatus    OBJECT-TYPE
    SYNTAX AvailabilityStatus
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "    This object controls the Availability status of the EFD
            which reflects the scheduling. "
   ::= {semEfdEntry 4}

semEfdStartTime  OBJECT-TYPE
    SYNTAX  DateAndTime
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  This variable defines the date and time at which an unlocked and
            enabled EFD starts functioning, i.e. changes its
            availability status from offDuty to available. "
   ::= {semEfdEntry 5}

semEfdStopTime  OBJECT-TYPE
    SYNTAX  DateAndTime
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  This variable defines the date and time at which an unlocked and
            enabled EFD stops functioning, i.e. changes its
            availability status from available to offDuty. "
   ::= {semEfdEntry 6}

semEfdDailyStartTime  OBJECT-TYPE
    SYNTAX  TimeTicks
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  This variable defines the daily start time at which an unlocked and
            enabled EFD starts functioning, i.e. changes its
            availability status from offDuty to available. "
   ::= {semEfdEntry 7}

semEfdDailyStopTime  OBJECT-TYPE
    SYNTAX  TimeTicks
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  This variable defines the daily stop time at which an unlocked and
            enabled EFD stops functioning, i.e. changes its
            availability status from available to offDuty. "
   ::= {semEfdEntry 8}

semEfdWeeklyMask  OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE (1))
    MAX-ACCESS  read-create
STATUS      current
    DESCRIPTION
             "  This variable defines the weekly schedule at which an unlocked and
            enabled EFD may start functioning, i.e. changes its
            availability status from available to offDuty. A day is
            scheduled if the cooresponding power of 2, i.e. 2**3 for
            Wednesday is in the mask. "
   ::= {semEfdEntry 9}

semEfdTypes OBJECT-TYPE
    SYNTAX      EventType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  The event types that this EFD may generate notifications for. "
   ::= {semEfdEntry 10}

semEfdCause OBJECT-TYPE
    SYNTAX      ProbableCause
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         " Any event with a different probable cause is ignored. "
   ::= {semEfdEntry 11}

semEfdSeverity  OBJECT-TYPE
    SYNTAX      PerceivedSeverity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        " Any event with severity equal to or less the discriminated level
          is ignored. "

   ::= {semEfdEntry 12}

semEfdSpecificProblems     OBJECT-TYPE
   SYNTAX       OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  Any event not generated by the identified object is ignored. "
   ::= {semEfdEntry 13}

semEfdTrendIndication   OBJECT-TYPE
    SYNTAX      TrendIndication
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
             "  Any events with a trend less or equal to specified value are
            ignored. "
   ::= {semEfdEntry 14}

semEfdChangedObjectId OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "Any events not caused by a change of the value of this object
        are ignored. "
   ::= {semEfdEntry 15}

semEfdToStateChange  OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         " Any state changes to a different state than the one indicated are
      ignored. "
   ::= {semEfdEntry 16}

semEfdNotification OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The object identifier from the NOTIFICATION-TYPE for the
        notification. "
   ::= {semEfdEntry 17}

semEfdOr OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         " Indicates whether if this EFD matches the event, the matching
    process shall be continued with the next EFD. "
   ::= {semEfdEntry 18}

semEfdTarget OBJECT-TYPE
    SYNTAX      IpAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "Notifications Targets.  A value of 0
        indicates the local system. "
::= {semEfdEntry 19}

semEfdText OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "A description of the EFD's function and use. "
   ::= {semEfdEntry 20}


semEfdStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The control that allows creation/deletion of entries. "
   ::= {semEfdEntry 21}

--
-- Notifications
--


semMIBNotifications OBJECT IDENTIFIER::= {semMIBNotificationPrefix 0}



semEventAlarm NOTIFICATION-TYPE
    OBJECTS
        {   semEventName,
            semEventType,
            semEventProbableCause,
            semEventSpecificProblems,
            semEventPerceivedSeverity,
            semEventTrendIndication,
            semEventText
        }
    STATUS  current
    DESCRIPTION
             " Alarm notification. "
   ::= {semMIBNotifications 1}


   semEventStateChange NOTIFICATION-TYPE
    OBJECTS
        {   semEventName,
            semEventToStateChange,
            semEventChangedObjectId
        }
    STATUS  current
    DESCRIPTION
         " State change notification. "
   ::= {semMIBNotifications 2}

   semEventObjectValueChange NOTIFICATION-TYPE
    OBJECTS
        {   semEventName,
            semEventChangedObjectId
        }
    STATUS  current
    DESCRIPTION
             " Object value change. "
   ::= {semMIBNotifications 3}
```

```
--
--  Conformance Information
--

semCompliances      OBJECT IDENTIFIER::= {semMIBConformance 1}
semGroups       OBJECT IDENTIFIER::= {semMIBConformance 2}

semECMGCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent ECMGs. A hosting entity must also support either
    threshold, state change, or value change events. "
    MODULE  -- this module
    MANDATORY-GROUPS {semMandatoryNotifications}
    GROUP semThresholdEventGroup
    DESCRIPTION
     " This group is required  if thresold events are supported. "
    GROUP semThresholdEventOptGroup
    DESCRIPTION
     " This group is optional  if thresold events are supported. "
    GROUP semThresholdEfdGroup
    DESCRIPTION
     " This group is required if threshold events are supported. "
    GROUP semThresholdEfdOptGroup
    DESCRIPTION
     " This group is optional if threshold events are supported. "
    GROUP semStateChangeEventGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEventOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semStateChangeEfdGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semValueChangeEventGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEventOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semValueChangeEfdGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semOptionalNotifications
    DESCRIPTION
     " This Group is required if state change or value change notifications
    are supported. "
    ::= {semCompliances 1}

semEmOrPdCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent EMMG/PDGs. A hosting entity must also support either
    threshold, state change, or value change events. "
    MODULE  -- this module
    MANDATORY-GROUPS {semMandatoryNotifications}
    GROUP semThresholdEventGroup
    DESCRIPTION
     " This group is required  if thresold events are supported. "
    GROUP semThresholdEventOptGroup
    DESCRIPTION
     " This group is optional  if thresold events are supported. "
    GROUP semThresholdEfdGroup
    DESCRIPTION
     " This group is required if threshold events are supported. "
    GROUP semThresholdEfdOptGroup
    DESCRIPTION
     " This group is optional if threshold events are supported. "
    GROUP semStateChangeEventGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEventOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semStateChangeEfdGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semValueChangeEventGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEventOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semValueChangeEfdGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semOptionalNotifications
    DESCRIPTION
     " This Group is required if state change or value change notifications
    are supported. "
    ::= {semCompliances 2}

semCpsigCompliance  MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent CPSIGs. A hosting entity must also support either
    threshold, state change, or value change events. "
    MODULE  -- this module
    MANDATORY-GROUPS {semMandatoryNotifications}
    GROUP semThresholdEventGroup
    DESCRIPTION
     " This group is required  if thresold events are supported. "
    GROUP semThresholdEventOptGroup
    DESCRIPTION
     " This group is optional  if thresold events are supported. "
    GROUP semThresholdEfdGroup
    DESCRIPTION
     " This group is required if threshold events are supported. "
    GROUP semThresholdEfdOptGroup
    DESCRIPTION
     " This group is optional if threshold events are supported. "
    GROUP semStateChangeEventGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEventOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semStateChangeEfdGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semValueChangeEventGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEventOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semValueChangeEfdGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semOptionalNotifications
    DESCRIPTION
     " This Group is required if state change or value change notifications
    are supported. "
    ::= {semCompliances 3}

semCsigCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent CSIGs. A hosting entity must also support either
    threshold, state change, or value change events. "
    MODULE  -- this module
    MANDATORY-GROUPS {semMandatoryNotifications}
    GROUP semThresholdEventGroup
    DESCRIPTION
     " This group is required  if thresold events are supported. "
    GROUP semThresholdEventOptGroup
    DESCRIPTION
     " This group is optional  if thresold events are supported. "
    GROUP semThresholdEfdGroup
    DESCRIPTION
     " This group is required if threshold events are supported. "
    GROUP semThresholdEfdOptGroup
    DESCRIPTION
     " This group is optional if threshold events are supported. "
    GROUP semStateChangeEventGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEventOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semStateChangeEfdGroup
    DESCRIPTION
     " This Group is required if stateChange events are supported. "
    GROUP semStateChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional if stateChange events are supported. "
    GROUP semValueChangeEventGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEventOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semValueChangeEfdGroup
    DESCRIPTION
     " This Group is required valueChange events are supported. "
    GROUP semValueChangeEfdOptGroup
    DESCRIPTION
     " This Group is optional valueChange events are supported. "
    GROUP semOptionalNotifications
    DESCRIPTION
     " This Group is required if state change or value change notifications
    are supported. "
    ::= {semCompliances 4}

semMandatoryNotifications   NOTIFICATION-GROUP
    NOTIFICATIONS {
        semEventAlarm
    }
    STATUS current
    DESCRIPTION
     " A collection of objects defining mandatory notifications. "
    ::= {semGroups 1}

semOptionalNotifications    NOTIFICATION-GROUP
    NOTIFICATIONS {
        semEventStateChange,
        semEventObjectValueChange
    }
    STATUS current
    DESCRIPTION
     " A collection of objects defining optional notifications. "
    ::= {semGroups 2}

semThresholdEventGroup  OBJECT-GROUP
    OBJECTS {
        semEventName,
        semEventAdminState,
        semEventType,
        semEventText,
        semEventChangedObjectId,
        semEventRisingThreshold,
        semEventFallingThreshold,
        semEventProbableCause,
        semEventPerceivedSeverity,
        semEventTrendIndication,
        semEventFrequency,
        semEventStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying threshold events. "
    ::= {semGroups 3}

semThresholdEventOptGroup   OBJECT-GROUP
```

```
    OBJECTS {
        semEventAlarmStatus,
        semEventBackedUpStatus,
        semEventBackUpObject,
        semEventSpecificProblems,
        semEventSensitivity
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying threshold events. "
    ::= {semGroups 4}

semStateChangeEventGroup    OBJECT-GROUP
    OBJECTS {
        semEventName,
        semEventAdminState,
        semEventType,
        semEventText,
        semEventChangedObjectId,
        semEventToStateChange,
        semEventFrequency,
        semEventStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying state change events. "
    ::= {semGroups 5}


semStateChangeEventOptGroup OBJECT-GROUP
    OBJECTS {
        semEventAlarmStatus,
        semEventProbableCause,
        semEventPerceivedSeverity,
        semEventTrendIndication,
        semEventBackedUpStatus,
        semEventBackUpObject,
        semEventSpecificProblems,
        semEventSensitivity
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying state change events. "
    ::= {semGroups 6}

semValueChangeEventGroup    OBJECT-GROUP
    OBJECTS {
        semEventName,
        semEventAdminState,
        semEventType,
        semEventText,
        semEventChangedObjectId,
        semEventFrequency,
        semEventStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying value change events. "
    ::= {semGroups 7}


semValueChangeEventOptGroup OBJECT-GROUP
    OBJECTS {
        semEventAlarmStatus,
        semEventProbableCause,
        semEventPerceivedSeverity,
        semEventTrendIndication,
        semEventBackedUpStatus,
        semEventBackUpObject,
        semEventSpecificProblems,
        semEventSensitivity
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying value change events. "
    ::= {semGroups 8}

semThresholdEfdGroup    OBJECT-GROUP
    OBJECTS {
            semEfdName,
        semEfdAdminState,
        semEfdOperState,
        semEfdAvailStatus,
        semEfdTypes,
        semEfdCause ,
        semEfdSeverity,
        semEfdTrendIndication ,
        semEfdChangedObjectId,
            semEfdNotification,
        semEfdOr,
        semEfdTarget,
        semEfdText,
        semEfdStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying threshold EFDs. "
    ::= {semGroups 9}

semThresholdEfdOptGroup OBJECT-GROUP
    OBJECTS {
        semEfdStartTime,
        semEfdStopTime,
        semEfdDailyStartTime,
        semEfdDailyStopTime,
        semEfdWeeklyMask,
        semEfdSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying threshold EFDs. "
    ::= {semGroups 10}

semStateChangeEfdGroup  OBJECT-GROUP
    OBJECTS {
            semEfdName,
        semEfdAdminState,
        semEfdOperState,
        semEfdAvailStatus,
        semEfdTypes,
        semEfdToStateChange,
            semEfdNotification,
        semEfdOr,
        semEfdTarget,
        semEfdText,
        semEfdStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying state change EFDs. "
    ::= {semGroups 11}

semStateChangeEfdOptGroup   OBJECT-GROUP
    OBJECTS {
        semEfdStartTime,
        semEfdStopTime,
        semEfdDailyStartTime,
        semEfdDailyStopTime,
        semEfdCause ,
        semEfdSeverity,
        semEfdTrendIndication ,
        semEfdChangedObjectId,
        semEfdWeeklyMask,
        semEfdSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying state change EFDs. "
    ::= {semGroups 12}

semValueChangeEfdGroup  OBJECT-GROUP
    OBJECTS {
            semEfdName,
        semEfdAdminState,
        semEfdOperState,
        semEfdAvailStatus,
        semEfdTypes,
        semEfdChangedObjectId,
            semEfdNotification,
        semEfdOr,
        semEfdTarget,
        semEfdText,
        semEfdStatus
    }
    STATUS current
    DESCRIPTION
```

```
     " A collection of  objects specifying value change EFDs. "
    ::= {semGroups 13}

semValueChangeEfdOptGroup   OBJECT-GROUP
    OBJECTS {
        semEfdStartTime,
        semEfdStopTime,
        semEfdDailyStartTime,
        semEfdDailyStopTime,
        semEfdCause ,
        semEfdSeverity,
        semEfdTrendIndication ,
        semEfdWeeklyMask,
        semEfdSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying value change EFDs. "
    ::= {semGroups 14}

END
```

G.3 SLM MIB

```
SLM-MIB DEFINITIONS::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Unsigned32, BITS, TimeTicks   FROM SNMPv2-SMI
    RowStatus, DisplayString, DateAndTime FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP     FROM SNMPv2-CONF
    EventType, ProbableCause, PerceivedSeverity, TrendIndication,
    AdministrativeState, OperationalState,
    AvailabilityStatus, EntryName FROM SEM-MIB;


slmMIB MODULE-IDENTITY
    LAST-UPDATED  "9708071700Z "
    ORGANIZATION  "DVB Simulcrypt Technical Group "
    CONTACT-INFO  " ---  "
    DESCRIPTION
         "The MIB module for defining DVB Simulcrypt Conditional
    Access System logs information. "
   ::= {1 3 6 1 4 1 2696 1 3}

slmMIBObjects       OBJECT IDENTIFIER::= {slmMIB 1}
slmMIBConformance   OBJECT IDENTIFIER::= {slmMIB 2}

slmLogControl              OBJECT IDENTIFIER::= {slmMIBObjects 1}
slmLogs                   OBJECT IDENTIFIER::= {slmMIBObjects 2}

--
-- Log Control Group
--

slmLogDefinitionTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF SlmLogDefinitionEntry
        MAX-ACCESS      not-accessible
        STATUS      current
        DESCRIPTION
             "A list of Log Table Entry Definitions. Identifies log table
         and the types of events to be logged into that table. "
        REFERENCE    " --  "
   ::= {slmLogControl 1}


slmLogDefinitionEntry OBJECT-TYPE
        SYNTAX      SlmLogDefinitionEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
             "An entry (conceptual row) in the Log Definition Table. "
        REFERENCE    " --  "
    INDEX       {slmLogDefinitionName }
   ::= {slmLogDefinitionTable 1}


SlmLogDefinitionEntry::= SEQUENCE
    {
        slmLogDefinitionName        EntryName,
        slmLogDefinitionId          OBJECT IDENTIFIER,
        slmLogDefinitionAdminState      AdministrativeState,
        slmLogDefinitionOperState       OperationalState,
        slmLogDefinitionAvailStatus AvailabilityStatus,
        slmLogDefinitionFullAction      BITS,
        slmLogDefinitionMaxLogSize      INTEGER,
        slmLogDefinitionCurrentLogSize  INTEGER,
        slmLogDefinitionNumberOfRecords INTEGER
   }

slmLogDefinitionName OBJECT-TYPE
        SYNTAX      EntryName
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
             " The variable used for identifying log Definition table entries.  "
        REFERENCE    " --  "
   ::= {slmLogDefinitionEntry 1}

slmLogDefinitionId OBJECT-TYPE
        SYNTAX      OBJECT IDENTIFIER
        MAX-ACCESS      read-create
        STATUS      current
        DESCRIPTION
             " The variable used for identifying log tables.  "
        REFERENCE    " --  "
   ::= {slmLogDefinitionEntry 2}

slmLogDefinitionAdminState  OBJECT-TYPE
        SYNTAX      AdministrativeState
        MAX-ACCESS      read-create
        STATUS      current
        DESCRIPTION
             "  The current Adminsitrative state of the LOG as defined in
                ISO/IEC 10164-2 [8]  "
   ::= {slmLogDefinitionEntry 3}

slmLogDefinitionOperState  OBJECT-TYPE
        SYNTAX      OperationalState
        MAX-ACCESS      read-create
        STATUS      current
        DESCRIPTION
             "  The current Operational state of the LOG as defined in
                ISO/IEC 10164-2 [8]  "
   ::= {slmLogDefinitionEntry 4}

slmLogDefinitionAvailStatus    OBJECT-TYPE
        SYNTAX     AvailabilityStatus
        MAX-ACCESS      read-only
        STATUS      current
        DESCRIPTION
             "  This object Definitions the Availability status of the LOG
                as defined in ISO/IEC 10164-2 [8] and state-machine "
   ::= {slmLogDefinitionEntry 5}

slmLogDefinitionFullAction    OBJECT-TYPE
        SYNTAX      BITS
                    {
                        wrap(0),
                        halt(1)
                   }
        MAX-ACCESS      read-create
        STATUS      current
        DESCRIPTION
             "  This object Definitions the action to be taken when the maximum
                size of the log has been reached. "
   ::= {slmLogDefinitionEntry 6}

slmLogDefinitionMaxLogSize    OBJECT-TYPE
        SYNTAX      INTEGER (1..4294967295)
        MAX-ACCESS      read-only
        STATUS      current
        DESCRIPTION
             "  This object specifies the maximum size of a log in number
                of octets. A size of 2**32 - 1 specifies that there is no limit. "
   ::= {slmLogDefinitionEntry 7}

slmLogDefinitionCurrentLogSize    OBJECT-TYPE
        SYNTAX      INTEGER (1..4294967295)
        MAX-ACCESS      read-only
        STATUS      current
        DESCRIPTION
             "  This object specifies the current size of a log in number
```

```
                of octets. "
   ::= {slmLogDefinitionEntry 8}

slmLogDefinitionNumberOfRecords    OBJECT-TYPE
        SYNTAX      INTEGER (1..4294967295)
        MAX-ACCESS      read-only
        STATUS      current
        DESCRIPTION
             "  This object specifies the number of records in the log. "
        REFERENCE    "ITU-T Recommendation X.735 [11] "
   ::= {slmLogDefinitionEntry 9}

slmLogControlTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF SlmLogControlEntry
        MAX-ACCESS      not-accessible
        STATUS      current
        DESCRIPTION
             "A list of Log Table Filter Definitions. Since
         multiple types of events can be logged into the same table,
         multiple entries in the log table could be defining the same
         log table. "
        REFERENCE    " --  "
   ::= {slmLogControl 2}


slmLogControlEntry OBJECT-TYPE
        SYNTAX      SlmLogControlEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
             "An entry (conceptual row) in the Log Control Table. "
        REFERENCE    " --  "
    INDEX       {slmLogDefinitionName, slmLogControlName }
   ::= {slmLogControlTable 1}


SlmLogControlEntry::= SEQUENCE
    {
        slmLogControlName       EntryName,
        slmLogControlStartTime      DateAndTime,
        slmLogControlStopTime       DateAndTime,
        slmLogControlDailyStartTime TimeTicks,
        slmLogControlDailyStopTime  TimeTicks,
        slmLogControlWeeklyMask     OCTET STRING,
        slmLogControlTypes          EventType,
        slmLogControlCause      ProbableCause,
        slmLogControlSeverity       PerceivedSeverity,
        slmLogControlSpecificProblems   OBJECT IDENTIFIER,
        slmLogControlToStateChange  Unsigned32,
        slmLogControlTrendIndication    TrendIndication,
        slmLogControlChangedObjectId    OBJECT IDENTIFIER,
        slmLogControlStatus RowStatus
   }
slmLogControlName OBJECT-TYPE
        SYNTAX      EntryName
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
             " The variable used for identifying log control table entries.  "
        REFERENCE    " --  "
   ::= {slmLogControlEntry 1}

slmLogControlStartTime  OBJECT-TYPE
        SYNTAX  DateAndTime
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the date and time at which an unlocked and
            enabled log control entry starts functioning, i.e. changes its
            availability status from offDuty to available. "
   ::= {slmLogControlEntry 2}

slmLogControlStopTime  OBJECT-TYPE
        SYNTAX  DateAndTime
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the date and time at which an unlocked and
            enabled log control stops functioning, i.e. changes its
            availability status from available to offDuty. "
   ::= {slmLogControlEntry 3}

slmLogControlDailyStartTime  OBJECT-TYPE
```

```
        SYNTAX  TimeTicks
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the daily start time at which an unlocked and
            enabled log control entry starts functioning, i.e. changes its
            availability status from offDuty to available. "
   ::= {slmLogControlEntry 4}

slmLogControlDailyStopTime  OBJECT-TYPE
        SYNTAX  TimeTicks
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the daily start time at which an unlocked and
            enabled log control entry stops functioning, i.e. changes its
            availability status from available to offDuty. "
   ::= {slmLogControlEntry 5}

slmLogControlWeeklyMask  OBJECT-TYPE
        SYNTAX  OCTET STRING (SIZE (1))
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the weekly schedule at which an unlocked and
            enabled log control entry may start functioning, i.e. changes its
            availability status from available to offDuty. A day is
            scheduled if the cooresponding power of 2, i.e. 2**3 for
            Wednesday is in the mask. "
   ::= {slmLogControlEntry 6}


slmLogControlTypes OBJECT-TYPE
        SYNTAX      EventType
        MAX-ACCESS      read-create
        STATUS      current
        DESCRIPTION
             "  This variable defines the type of events being logged. "
   ::= {slmLogControlEntry 7}

slmLogControlCause OBJECT-TYPE
        SYNTAX      ProbableCause
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
         " Any event with a different probable cause is ignored. "
   ::= {slmLogControlEntry 8}

slmLogControlSeverity  OBJECT-TYPE
        SYNTAX      PerceivedSeverity
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             " Any event with severity equal to or less the discriminated level
          is ignored. "

   ::= {slmLogControlEntry 9}

slmLogControlSpecificProblems     OBJECT-TYPE
        SYNTAX      OBJECT IDENTIFIER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  Any event not generated by the identified object is ignored. "
   ::= {slmLogControlEntry 10}


slmLogControlToStateChange  OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
         " Any state changes to a different state than the one indicated are
      ignored. "
   ::= {slmLogControlEntry 11}

slmLogControlTrendIndication   OBJECT-TYPE
        SYNTAX      TrendIndication
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
             "  Any events with a trend less or equal to specified value are
            ignored. "
   ::= {slmLogControlEntry 12}

slmLogControlChangedObjectId OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
         "Any events not caused by a change of the value of this object
     are ignored. "
   ::= {slmLogControlEntry 13}

slmLogControlStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
         "The control that allows creation/deletion of entries.
        Once made active an entry may not be modified except to. "
   ::= {slmLogControlEntry 14}

--
-- Alarm  Log Group
--


slmAlarmLogTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SlmAlarmLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A table of logged alarm events. "
   ::= {slmLogs 1}

slmAlarmLogEntry OBJECT-TYPE
    SYNTAX      SlmAlarmLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A single alarm log. "
    INDEX       {slmAlarmLogTime}
   ::= {slmAlarmLogTable 1}

SlmAlarmLogEntry::= SEQUENCE {
    slmAlarmLogName     EntryName,
    slmAlarmLogTime     TimeTicks,
    slmAlarmLogText     DisplayString,
    slmAlarmLogType     EventType,
    slmAlarmLogCause        ProbableCause,
    slmAlarmLogSeverity     PerceivedSeverity,
    slmAlarmLogSpecificProblems OBJECT IDENTIFIER,
    slmAlarmLogTrendIndication  TrendIndication,
    slmAlarmLogChangedObjectId  OBJECT IDENTIFIER
}

slmAlarmLogName OBJECT-TYPE
    SYNTAX      EntryName
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The  name of the logged event. "
   ::= {slmAlarmLogEntry 1}

slmAlarmLogTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The  name of the logged event. "
   ::= {slmAlarmLogEntry 2}

slmAlarmLogText OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "A description of the event's function and use. "
    DEFVAL {''H}
   ::= {slmAlarmLogEntry 3}

slmAlarmLogType OBJECT-TYPE
        SYNTAX      EventType
    MAX-ACCESS  read-only
    STATUS      current
```

```
        DESCRIPTION
         " Indicates the type of the event. "
    ::= {slmAlarmLogEntry 4}

slmAlarmLogCause OBJECT-TYPE
    SYNTAX ProbableCause
    MAX-ACCESS  read-only
    STATUS      current
        DESCRIPTION
         "This variable defines further probable cause for
         the event. "
   ::= {slmAlarmLogEntry 5}

slmAlarmLogSeverity  OBJECT-TYPE
        SYNTAX      PerceivedSeverity
    MAX-ACCESS  read-only
    STATUS      current
        DESCRIPTION
             "This parameter defines the Perceived severity of the
         event. "
   ::= {slmAlarmLogEntry 6}

slmAlarmLogSpecificProblems       OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         " This variable identifies the object responsible for the event.  "
   ::= {slmAlarmLogEntry 7}

slmAlarmLogTrendIndication   OBJECT-TYPE
    SYNTAX  TrendIndication
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "Indicates the trend of the event. "
   ::= {slmAlarmLogEntry 8}

slmAlarmLogChangedObjectId  OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         " The object identifier of the object which changed value. "
   ::= {slmAlarmLogEntry 9}


--
-- State Logs Group
--

slmStateChangeLogTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SlmStateChangeLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A table of logged stateChange events. "
   ::= {slmLogs 2}

slmStateChangeLogEntry OBJECT-TYPE
    SYNTAX      SlmStateChangeLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A single stateChange log. "
    INDEX       {slmStateChangeLogTime}
   ::= {slmStateChangeLogTable 1}

SlmStateChangeLogEntry::= SEQUENCE {
    slmStateChangeLogName           EntryName,
    slmStateChangeLogTime               TimeTicks,
    slmStateChangeLogText                   DisplayString,
    slmStateChangeLogToStateChange          Unsigned32,
    slmStateChangeLogChangedObjectId        OBJECT IDENTIFIER
}

slmStateChangeLogName OBJECT-TYPE
    SYNTAX      EntryName
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The name of the logged event. "
   ::= {slmStateChangeLogEntry 1}
```

```
slmStateChangeLogTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The name of the logged event. "
   ::= {slmStateChangeLogEntry 2}

slmStateChangeLogText OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "A description of the event's function and use. "
    DEFVAL {''H}
   ::= {slmStateChangeLogEntry 3}

slmStateChangeLogToStateChange  OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "If ChangedObjectId is a state/status/variable,
         this variable identifies the state that caused the
         event to be generated. "
   ::= {slmStateChangeLogEntry 4}

slmStateChangeLogChangedObjectId OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The object identifier of the MIB object that caused the event. "
   ::= {slmStateChangeLogEntry 5}

--
-- Object Value Change Logs Group
--

slmValueChangeLogTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SlmValueChangeLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A table of logged valuechange events. "
   ::= {slmLogs 3}

slmValueChangeLogEntry OBJECT-TYPE
    SYNTAX      SlmValueChangeLogEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "A single valuechange log. "
    INDEX       {slmValueChangeLogTime}
   ::= {slmValueChangeLogTable 1}

SlmValueChangeLogEntry::= SEQUENCE {
    slmValueChangeLogName           EntryName,
    slmValueChangeLogTime               TimeTicks,
    slmValueChangeLogText                   DisplayString,
    slmValueChangeLogChangedObjectId        OBJECT IDENTIFIER
}

slmValueChangeLogName OBJECT-TYPE
    SYNTAX      EntryName
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The  name of the logged event. "
   ::= {slmValueChangeLogEntry 1}

slmValueChangeLogTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The  name of the logged event. "
   ::= {slmValueChangeLogEntry 2}

slmValueChangeLogText OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "A description of the event's function and use. "
    DEFVAL {''H}
   ::= {slmValueChangeLogEntry 3}

slmValueChangeLogChangedObjectId OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The object identifier of the MIB object that caused the event. "
   ::= {slmValueChangeLogEntry 4}



--
--  Conformance Information
--

slmCompliances      OBJECT IDENTIFIER::= {slmMIBConformance 1}
slmGroups       OBJECT IDENTIFIER::= {slmMIBConformance 2}

slmECMGCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent ECMGs. A hosting entity must also support either
    threshold, state change, or value change logs. "
    MODULE  -- this module
    GROUP slmThresholdLogControlGroup
    DESCRIPTION
     " This group is required if threshold logs are supported. "
    GROUP slmThresholdLogControlOptGroup
    DESCRIPTION
     " This group is optional if threshold logs are supported. "
    GROUP slmStateChangeLogControlGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmStateChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional if stateChange logs are supported. "
    GROUP slmValueChangeLogControlGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    GROUP slmValueChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional valueChange logs are supported. "
    GROUP slmAlarmLogGroup
    DESCRIPTION
     " This group is required  if alarm logs are supported. "
    GROUP slmAlarmLogOptGroup
    DESCRIPTION
     " This group is optional  if alarm logs are supported. "
    GROUP slmStateChangeLogGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmValueChangeLogGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    ::= {slmCompliances 1}

slmEmOrPdCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent EMMG/PDGs. A hosting entity must also support either
    threshold, state change, or value change logs. "
    MODULE  -- this module
    GROUP slmThresholdLogControlGroup
    DESCRIPTION
     " This group is required if threshold logs are supported. "
    GROUP slmThresholdLogControlOptGroup
    DESCRIPTION
     " This group is optional if threshold logs are supported. "
    GROUP slmStateChangeLogControlGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmStateChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional if stateChange logs are supported. "
    GROUP slmValueChangeLogControlGroup
    DESCRIPTION
```

```
     " This Group is required valueChange logs are supported. "
    GROUP slmValueChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional valueChange logs are supported. "
    GROUP slmAlarmLogGroup
    DESCRIPTION
     " This group is required  if alarm logs are supported. "
    GROUP slmAlarmLogOptGroup
    DESCRIPTION
     " This group is optional  if alarm logs are supported. "
    GROUP slmStateChangeLogGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmValueChangeLogGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    ::= {slmCompliances 2}

slmCpsigCompliance  MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent CPSIGs. A hosting entity must also support either
    threshold, state change, or value change logs. "
    MODULE  -- this module
    GROUP slmThresholdLogControlGroup
    DESCRIPTION
     " This group is required if threshold logs are supported. "
    GROUP slmThresholdLogControlOptGroup
    DESCRIPTION
     " This group is optional if threshold logs are supported. "
    GROUP slmStateChangeLogControlGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmStateChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional if stateChange logs are supported. "
    GROUP slmValueChangeLogControlGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    GROUP slmValueChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional valueChange logs are supported. "
    GROUP slmAlarmLogGroup
    DESCRIPTION
     " This group is required  if alarm logs are supported. "
    GROUP slmAlarmLogOptGroup
    DESCRIPTION
     " This group is optional  if alarm logs are supported. "
    GROUP slmStateChangeLogGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmValueChangeLogGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    ::= {slmCompliances 3}

slmCsigCompliance   MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
     " The compliance statement for SNMP Entities which host or
    represent CSIGs. A hosting entity must also support either
    threshold, state change, or value change logs. "
    MODULE  -- this module
    GROUP slmThresholdLogControlGroup
    DESCRIPTION
     " This group is required if threshold logs are supported. "
    GROUP slmThresholdLogControlOptGroup
    DESCRIPTION
     " This group is optional if threshold logs are supported. "
    GROUP slmStateChangeLogControlGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmStateChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional if stateChange logs are supported. "
    GROUP slmValueChangeLogControlGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    GROUP slmValueChangeLogControlOptGroup
    DESCRIPTION
     " This Group is optional valueChange logs are supported. "
    GROUP slmAlarmLogGroup
    DESCRIPTION
     " This group is required if alarm logs are supported. "
    GROUP slmAlarmLogOptGroup
    DESCRIPTION
     " This group is optional if alarm logs are supported. "
    GROUP slmStateChangeLogGroup
    DESCRIPTION
     " This Group is required if stateChange logs are supported. "
    GROUP slmValueChangeLogGroup
    DESCRIPTION
     " This Group is required valueChange logs are supported. "
    ::= {slmCompliances 4}

slmAlarmLogGroup    OBJECT-GROUP
    OBJECTS {
        slmAlarmLogName,
        slmAlarmLogTime,
        slmAlarmLogText,
        slmAlarmLogType,
        slmAlarmLogChangedObjectId
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying alarm logs. "
    ::= {slmGroups 1}

slmAlarmLogOptGroup OBJECT-GROUP
    OBJECTS {
        slmAlarmLogCause,
        slmAlarmLogSeverity,
        slmAlarmLogTrendIndication,
        slmAlarmLogSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying alarm logs. "
    ::= {slmGroups 2}

slmStateChangeLogGroup  OBJECT-GROUP
    OBJECTS {
        slmStateChangeLogName,
        slmStateChangeLogTime,
        slmStateChangeLogText,
        slmStateChangeLogChangedObjectId,
        slmStateChangeLogToStateChange
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying state change logs. "
    ::= {slmGroups 3}

slmValueChangeLogGroup  OBJECT-GROUP
    OBJECTS {
        slmValueChangeLogName,
        slmValueChangeLogTime,
        slmValueChangeLogText,
        slmValueChangeLogChangedObjectId
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying value change logs. "
    ::= {slmGroups 4}

slmThresholdLogControlGroup OBJECT-GROUP
    OBJECTS {
            slmLogControlName,
        slmLogControlTypes,
        slmLogControlCause ,
        slmLogControlSeverity,
        slmLogControlTrendIndication ,
        slmLogControlChangedObjectId,
        slmLogControlStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying threshold LOGCONTROLs. "
    ::= {slmGroups 5}

slmThresholdLogControlOptGroup  OBJECT-GROUP
    OBJECTS {
        slmLogControlStartTime,
        slmLogControlStopTime,
        slmLogControlDailyStartTime,
        slmLogControlDailyStopTime,
        slmLogControlWeeklyMask,
        slmLogControlSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying threshold LOGCONTROLs. "
    ::= {slmGroups 6}

slmStateChangeLogControlGroup   OBJECT-GROUP
    OBJECTS {
            slmLogControlName,
        slmLogControlTypes,
        slmLogControlToStateChange,
        slmLogControlChangedObjectId,
        slmLogControlStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying state change LOGCONTROLs. "
    ::= {slmGroups 7}

slmStateChangeLogControlOptGroup    OBJECT-GROUP
    OBJECTS {
        slmLogControlStartTime,
        slmLogControlStopTime,
        slmLogControlDailyStartTime,
        slmLogControlDailyStopTime,
        slmLogControlWeeklyMask,
        slmLogControlSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying state change LOGCONTROLs. "
    ::= {slmGroups 8}

slmValueChangeLogControlGroup   OBJECT-GROUP
    OBJECTS {
            slmLogControlName,
        slmLogControlTypes,
        slmLogControlChangedObjectId,
        slmLogControlStatus
    }
    STATUS current
    DESCRIPTION
     " A collection of objects specifying value change LOGCONTROLs. "
    ::= {slmGroups 9}

slmValueChangeLogControlOptGroup    OBJECT-GROUP
    OBJECTS {
        slmLogControlStartTime,
        slmLogControlStopTime,
        slmLogControlDailyStartTime,
        slmLogControlDailyStopTime,
        slmLogControlWeeklyMask,
        slmLogControlSpecificProblems
    }
    STATUS current
    DESCRIPTION
     " A collection of optional objects specifying value change LOGCONTROLs. "
    ::= {slmGroups 10}

END
```

Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

- FIPS 46-2 (1993): "Data Encryption Standard (DES)".
- FIPS 81 (1980): "DES Modes of Operation".
- ANSI X3.92 (1981): "Data Encryption Algorithm".
- ANSI X3.106 (1983): "Data Encryption Algorithm - Modes of Operation".
- Schneier, Bruce: "Applied Cryptography, Second Edition".
- Menezes, vanOorschot, Vanstone: "Handbook of Applied Cryptography".
- RFC 1157 (1990): "A Simple Network Management Protocol (SNMP)", J. Case, M. Fedor, M. Schoffstall, J. Davin.
- CCITT Recommendation X.209 (1988): "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".
- ISO/IEC 8825 (1990): "Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)".
- DAVIC "Systems Management Architecture", Baseline Document Number 13b63r11.

History

Document history		
V1.1.1	June 2000	Publication