ETSI TR 102 035 V1.1.1 (2002-04)

Technical Report

Digital Video Broadcasting (DVB); Implementation Guidelines of the DVB Simulcrypt Standard

Reference
DTR/JTC-DVB-136

Keywords
broadcasting, digital, DVB, video

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.fr

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by Joint Technical Committee (JTC) 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.

Introduction

From now on, the DVB Simulcrypt standard is a broadly implemented specification. This situation is due in particular to a significant property of the Simulcrypt solution, offering the option to use in the same head-end multiple CA systems simultaneously with the same content.

Actually the standard proposes, first, the definition of a head-end architecture reference model identifying logically each of its components. But above all, it specifies interoperable interfaces with those of these components supplied by the CA systems.

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 can be optional).

1 Scope

The present document provides implementation guidelines for the use and implementation of the DVB Simulcrypt standard.

It first tries to draw attention to the technical questions that need to be answered in setting up a DVB Simulcrypt head-end. It also provides guidelines which are intended to be highly recommended rules and as such, aims to facilitate the efficient and reliable implementation of the Simulcrypt model and of its interfaces.

The rules apply to broadcasters, network operators as well as manufacturers.

Clause 4 "Architecture" presents the DVB Simulcrypt Architecture Model as described in TS 103 197 [6].

Clause 5 "Compliance between version 1 and version 2 of the standard" explains how to support compliance between both versions 1 and 2 of ECMG and EMMG/PDG protocols.

Clause 6 "ECMG⇔SCS Protocol" fulfils the ECMG protocol description mainly by giving the state diagram.

Clause 7 "EMMG/PDG⇔MUX protocol" fulfils the EMMG/PDG protocol description by giving the state diagram and by clarifying the bandwidth concept, UDP behaviour and datagram packetization rules.

Clause 8 "C(P)SIG⇔(P)SIG" describes profiles for adapting the C(P)SIG protocol implementation to real and simple needs.

Clause 9 "SIMF based protocols - Role of NMS" allows for reducing the use of the MIB for monitoring only.

Clause 10 "Error management" clarifies error management in DVB Simulcrypt protocols.

Clause 11 "Redundancy management" sums up uniqueness aspects of protocol parameters and describes basic redundancy policies for functions of the DVB Simulcrypt architecture model.

The present document uses the terminology defined in TS 103 197 [6] and should be read in conjunction with that document.

2 References

For the purposes of this Technical Report (TR) the following references apply:

[1] ISO/IEC 13818-1 (1994): "Information technology - Generic coding of moving pictures and associated audio information: Systems".

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

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

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

[5] ETSI TS 101 197: "Digital Video Broadcasting (DVB); DVB SimulCrypt: Head-end architecture and synchronization".

[6] ETSI TS 103 197: "Digital Video Broadcasting (DVB); Head-end implementation of DVB SimulCrypt".

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_system_id: uniquely and globally identifies a particular CA provider, as registered in table 3 'CA_system_ID' of ETR 162

CA_subsystem_ID: Defined in TS 103 197 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 components: components brought by a CA provider for integration into a host head-end system

conditional access descriptor: used to signal either, one particular EMM stream, if found in the CAT, or one ECM stream needed to descramble one or several elementary streams of a particular service if found in a PMT, for one specific CA System identified by its CA_System_id.

NOTE: This CA descriptor can also contains private data whose format is proprietary to the CA System. The syntax of the CA_descriptor is specified in ISO/IEC 13818-1.

channel: application specific representation of an open TCP connection, allowing the association of application specific parameters with such a connection

NOTE: 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

Custom (P)SI Generator (C(P)SIG): Component responsible for generating private PSI descriptors and/or private SI descriptors. It interfaces to the (P)SI Generator.

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

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): component receiving a CW request from the SCS and returning 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): ECM messages but does not support ECM repetition, as defined in TS 103 197

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

generator: component producing data

host: computer system uniquely identified by its IP address, and as such addressable in a computer network.

NOTE: 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 ISO/IEC 13818-1.

NOTE: 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): the role of this head-end component is to perform time multiplexing of input data (AV streams, CA data, private data…) and to output an MPEG-2 transport stream

Network Management System (NMS): component responsible for monitoring and control of SIMF agents

NOTE: 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.

Private Data Generator (PDG): component shown in the DVB-Simulcrypt System Architecture diagram to highlight the fact that the EMMG to MUX interface can be used for EMMs but also for any other private data

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

NOTE: Unless otherwise specified within the present document all "reserved" bits shall be set to "1".

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

NOTE: Unless otherwise specified within the present document all "reserved_future_use" bits shall be set to "1".

resource: set of coherent functions, accessible through a server

NOTE: More than one resource can reside on a single host.

simulcrypt (part 1): Refers to the DVB standard: "Head-end architecture and synchronization" TS 101 197-1.

simulcrypt (final): Refers to the DVB standard: "Head-end implementation of DVB SimulCrypt" TS 103 197.

SCRambler (SCR): component responsible for scrambling data in the MPEG2 Transport stream using the Control Words received from the SCS

server: software entity exporting a resource

NOTE: 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, as defined in EN 300 468

(P)SI Generator ((P)SIG): component responsible for generating the PSI (ISO/IEC 13818-1) and/or the SI (EN 300 468) for the system

NOTE: The generic term (P)SIG refers to a head-end process that serves as a PSI Generator (PSIG), an SI Generator(SIG), or both PSI and SI Generator (PSISIG).

SimulCrypt Synchronizer (SCS): logical component that acquires Control Words, ECMs and synchronizes their play-out for all the Conditional Access Systems connected

stream: independent bi-directional data flow across a channel

NOTE: Multiple streams may flow on a single channel. Stream_IDs (e.g. ECM_stream_ID, Data_stream_ID, etc.) are used to tag messages belonging to a particular stream.

Super_CAS_id: 32-bit identifier formed by the concatenation of the CA_system_id and the CA_subsystem_id (see TS 101 197-1 and TS 103 197)

Transport Stream: Data structure defined in ISO/IEC 13818-1.

NOTE: 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:

AC	Access Criteria
bslbf	bit string, left bit first
C(P)SIG	Custom PSI/SI Generator
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CP	Crypto Period
CW	Control Word
CWG	Control Word Generator
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
ECM	Entitlement Control Message
ECMG	Entitlement Control Message Generator
EIS	Event Info Scheduler
EMM	Entitlement Management Message
EMMG	Entitlement Management Message Generator
IP	Internet Protocol
ISO	International Organization for Standardization
MIB	Management Information Base
MPEG	Moving Pictures Expert Group
Mux	MUltipleX (multiplexer)
MUX	MUltipleXer
NMS	Network Management System
PD	Private Data
PDG	Private Data Generator
PID	Packet IDentifier
PMT	Program Map Table
(P)SI	PSI and or SI
PSI	Program Specific Information
SCR	DVB compliant SCRambler

SCS	SimulCrypt Synchronizer
SI	Service Information
SIG	Service Information Generator
SIM	Simulcrypt Identification Module
SIMF	Simulcrypt Integrated Management Framework
SMS	Short Message Service (GSM)
STB	Set Top Box
TCP	Transport Control Protocol
TS	Transport Stream
UDP	User Datagram Protocol

4 Architecture

The system architecture, in figure 1, shows the logical relationships between the components and which component-to-component interfaces are defined by the DVB Simulcrypt standard. Other components exist in a head-end which are not illustrated i.e. SMS.

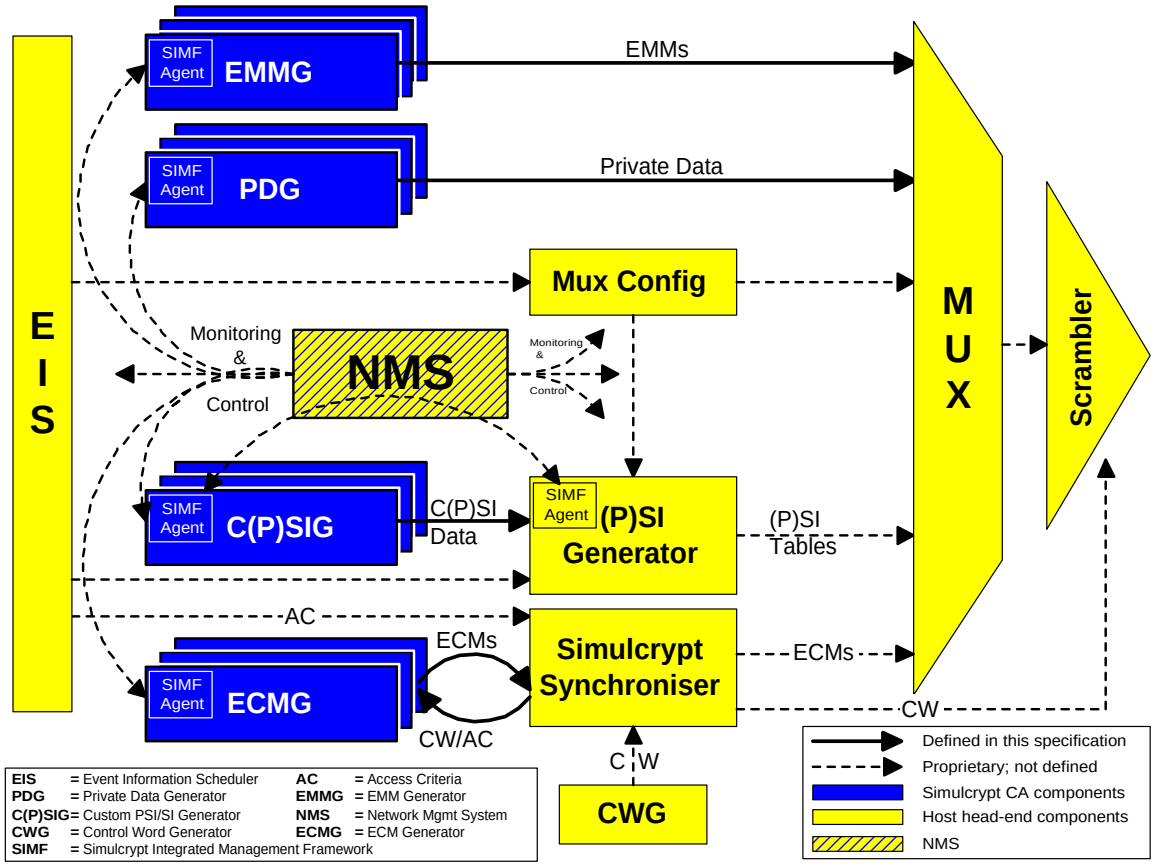


Figure 1: System architecture

The DVB-Simulcrypt system architecture illustrated above is divided into two areas:

- **Host Head-end components:** Those that will need to exist before Simulcrypt CA components can be introduced into a DVB-Simulcrypt head-end;
- **Simulcrypt CA components:** Those typical components which are brought by a new CA provider to introduce his CA into a DVB-Simulcrypt head-end. It must be noted that the EMMGs, PDGs and Custom SI generators are not necessarily required in a DVB-Simulcrypt system.
- The description of the role of these different components can be found in TS 101 197 [5] and in TS 103 197 [6].

5 Compliance between version 1 and version 2 of the standard

Two standards define the head-end implementation of DVB Simulcrypt: Simulcrypt (Part 1) TS 101 197 [5] and Simulcrypt (Final) TS 103 197 [6]. Both documents describe connection-oriented protocols.

According to the protocol_version parameter in each message:

- simulcrypt (Part 1) describes the ECMG protocol and the EMMG protocol in version 1 ("V1");
- simulcrypt (Final) describes the ECMG protocol, the EMMG protocol and the new C(P)SIG protocol in version 2 ("V2").

There is no compliance issue in a C(P)SIG protocol implementation nor in a SIMF implementation because these protocols are described only in the last standard Simulcrypt (final).

However, compliance issues can occur in a SCS/ECMG pair, or in a Mux/EMMG pair, according to their respective protocol versions.

5.1 ECMG protocol

5.1.1 Differences between Version 1 and Version 2

There are four differences between version 1 and version 2 of the ECMG protocol:

- CW length (cf. CP_CW_combination parameter): in V1, CW are 8 byte long; in V2, the CW length is variable;
- ECM_id: this parameter does not exist in V1 and is mandatory in V2;
- CW_Encryption: this parameter does not exist in version 1 and is optional in V2;
- the ECMG protocol error values are not strictly the same; in (Part 1), the protocol error value table is erroneous with two 0x000D codes; codes 0x000F, 0x0010 and 0x0011 in (Final) have different meanings as in (Part 1); in (Final) additional error codes are given.

A V1-SCS cannot connect a V2-ECMG:

- the protocol_version parameter in message header is wrong in messages received by the V2-ECMG and in messages received by the V1-SCS;
- for the V2-ECMG, the ECM_id parameter is missing;
- misunderstanding can occur for some error messages (0x000F to 0x0011) or the V2-ECMG can generate error messages unknown by the V1-SCS (0x0011 to 0x0015).

A V2-SCS cannot connect a V1-ECMG:

- the protocol_version parameter in message header is wrong in messages received by the V1-ECMG and in messages received by the V2-SCS;
- misunderstanding can occur for some error messages (0x000F to 0x0011) or the V2-SCS can generate error messages unknown by the V1-ECMG (0x0011, 0x0012);
- the ECM_id parameter is ignored by the V1-ECMG but is missing in messages received by the V2-SCS;
- if the CW are encrypted by the V2-SCS, the V1-ECMG ignores the CW_Encryption parameter (as an unknown parameter) but the V1-ECMG would process wrong values of CW (it could not decrypt them).

5.1.2 Recommendation for ECMG protocol compliance

A V1 + V2 configuration in a SCS/ECMG pair shall be avoided because it cannot work.

A V1 + V1 configuration or a V2 + V2 configuration in a SCS/ECMG pair is recommended.

If the SCS is compliant with V1 and V2, the SCS connects the ECMG in V2 mode. If the ECMG is V1-compliant, the following policy is recommended:

- such a V1-ECMG shall generate a V1 error message ("Unsupported protocol version");
- then the SCS disconnects, and connects again the ECMG in V1 mode.

If the ECMG is compliant with V1 and V2, whatever the version (1 or 2) of the SCS it is connected by, the following policy is recommended:

- such a V1 and V2 compliant ECMG selects its current version according to the version detected in the ***channel_setup*** message received from the SCS;
- this selection shall be performed for each channel, to allow a V1-SCS and a V2-SCS to connect the same ECMG.

	V1-SCS	V2-SCS	V1/V2-SCS	
V1-ECMG	OK		SCS → V1	Protocol version switch by SCS if error on Channel_Setup
V2-ECMG		OK	SCS → V2	
V1/V2-ECMG	ECMG → V1	ECMG → V2	V1 or V2	
	Protocol version is detected by ECMG at Channel_Setup		(same as ←)	

Figure 2: version 1/version 2 ECMG compliance

5.2 EMMG/PDG protocol

5.2.1 Differences between Version 1 and Version 2

There are four differences between version 1 and version 2 of the EMMG/PDG protocol:

- Data_id: this parameter does not exist in V1 and is mandatory in V2;
- Data_provision message: in V2 the data_channel_id parameter and the data_stream_id parameter are optional, according to the data part protocol TCP or UDP;
- Data_provision message are sent on the same TCP connection as other messages in V1; in V2 they can be sent on a separate UDP link; moreover one UDP link can connect several Muxes in broadcast mode, but associated with several TCP connections;
- the EMMG/PDG protocol error value list has been extended in [Final] (new codes 0x000E to 0x0014).

A V1-EMMG/PDG cannot connect a V2-Mux:

- the protocol_version parameter in message header is wrong in messages received by the V1-EMMG/PDG and in messages received by the V2-Mux;
- for the V2-Mux, the Data_id parameter is missing;
- the V2-Mux can generate error messages unknown by the V1-EMMG/PDG (0x000E to 0x0014).

A V2-EMMG/PDG cannot connect a V1-Mux:

- the protocol_version parameter in message header is wrong in messages received by the V2-EMMG/PDG and in messages received by the V1-Mux;
- the Data_id parameter is ignored by the V1-Mux but is missing in messages received by the V2-EMMG/PDG;
- the V2-EMMG/PDG can generate error messages unknown by the V1-Mux (0x000E, 0x000F);
- if the V2-EMMG/PDG send them over an UDP link, EMM datagram cannot be caught by the V1-Mux.

5.2.2 Recommendation for EMMG/PDG protocol compliance

A V1 + V2 configuration in a EMMG/PDG/Mux pair shall be avoided because it cannot work.

A V1 + V1 configuration or a V2 + V2 configuration in a EMMG/PDG/Mux pair is recommended.

If the EMMG/PDG is compliant with V1 and V2, the EMMG/PDG connects the Mux in V2 mode. If the Mux is V1 compliant, the following policy is recommended:

- such a V1-Mux shall generate an V1 error message ("Unsupported protocol version");
- then the EMMG/PDG disconnects, and connects again the Mux in V1 mode.

If the Mux is compliant with V1 and V2, whatever is the version 1 or 2 of the EMMG/PDG it is connected by; the following policy is recommended:

- such a V1 and V2 compliant Mux selects its current version according to the version detected in the channel_setup message received from the EMMG/PDG;
- this selection shall be performed for each channel, to allow a V1-EMMG/PDG and a V2-EMMG/PDG to connect the same Mux.

	V1-EMMG/PDG	V2-EMMG/PDG	V1/V2-EMMG/PDG	
V1-MUX	OK		EMMG/PDG → V1	Protocol version switch by EMMG/PDG if error on Channel_Setup
V2-MUX		OK	EMMG/PDG → V2	
V1/V2-MUX	MUX → V1	MUX → V2	V1 or V2	
	Protocol version is detected by Mux at Channel_Setup		(same as ←)	

Figure 3: version 1/version 2 EMMG/PDG compliance

6 ECMG (SCS Protocol)

6.1 State diagram

6.1.1 Channel state machine

This clause presents the channel state machine, which defines the sequence of channel-level messages that shall be used to establish and maintain one channel on one TCP connection.

The channel state machine is found in figure 4. Each state found in this state machine is defined in clauses 6.1.1.1 to 6.1.1.4.

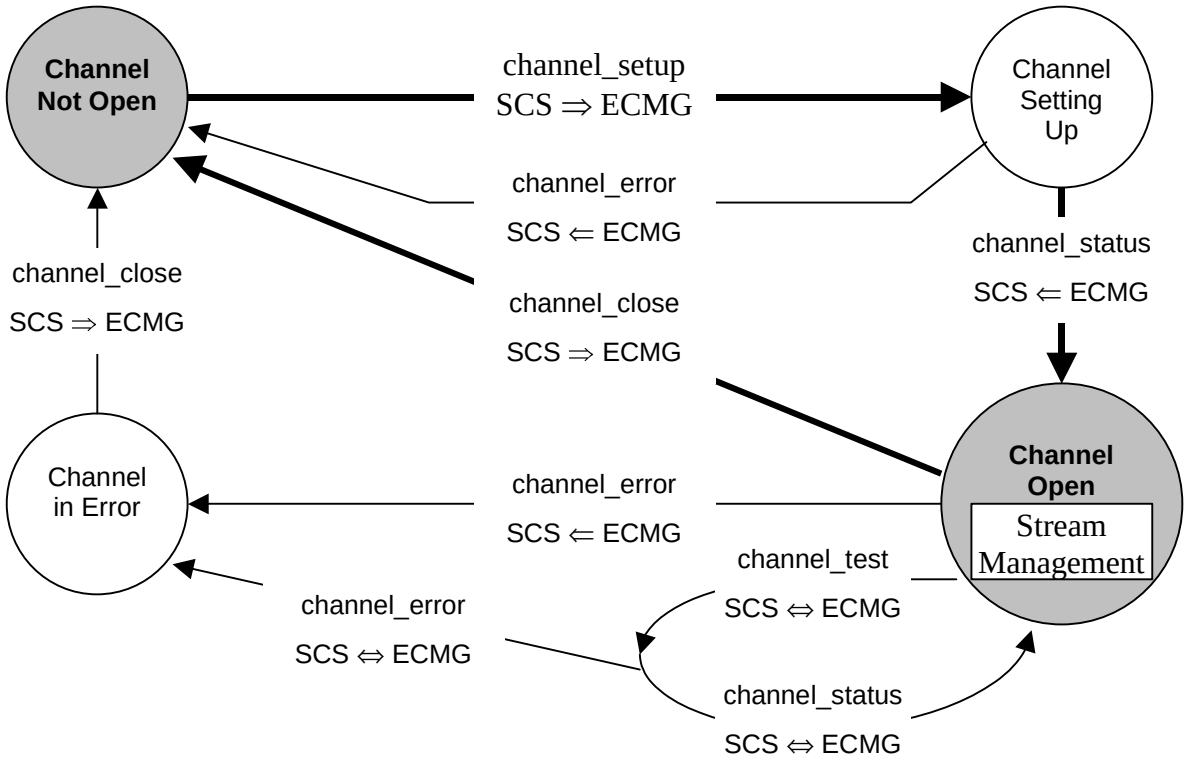


Figure 4 - ECMG ⇔ SCS channel state machine

6.1.1.1 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 SCS initializes a channel by sending a **channel_setup** message to the ECMG on the other end of the channel. **Channel_setup** is the only permissible message in the **Channel Not Open** state. Transmission and receipt of **channel_setup** move the state machine to the **Channel Setting Up** state.

6.1.1.2 Channel Setting Up

From this state, the ECMG 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 ECMG also indicates, via this message, several CAS specific parameters, particularly 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 ECMG could not open the channel; one or more error codes explain the failure. The channel shall be considered by the SCS and by the ECMG as not open. Transmission and receipt of **channel_error** move the state machine to the **Channel Not Open** state.

6.1.1.3 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 clause 3.1.2: 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 SCS or the ECMG 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 ECMG encounters an unrecoverable channel error at any other time, it shall send the SCS 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 SCS wants to close the channel for any reason, it shall send the ECMG 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 the **Channel Open** state.

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

6.1.1.4 Channel In Error

This temporary and short-lived state is used only to represent the fact that the ECMG has encountered and reported an unrecoverable channel error. The SCS shall close the channel.

The SCS sends a **channel_close** message to the ECMG. 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.

6.1.2 Stream state machine

The head-end can establish one or more streams within a channel. This clause 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 SCS 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 clause 6.1.1.3) for a SCS to initiate a stream state machine. The channel state machine, as defined in clause 6.1.1, continues to operate in **Channel Open** state during the operation of the stream state machine. Accordingly, both ECMG and SCS 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 5. Each state found in this state machine is defined in clauses 6.1.2.1 to 6.1.2.5.

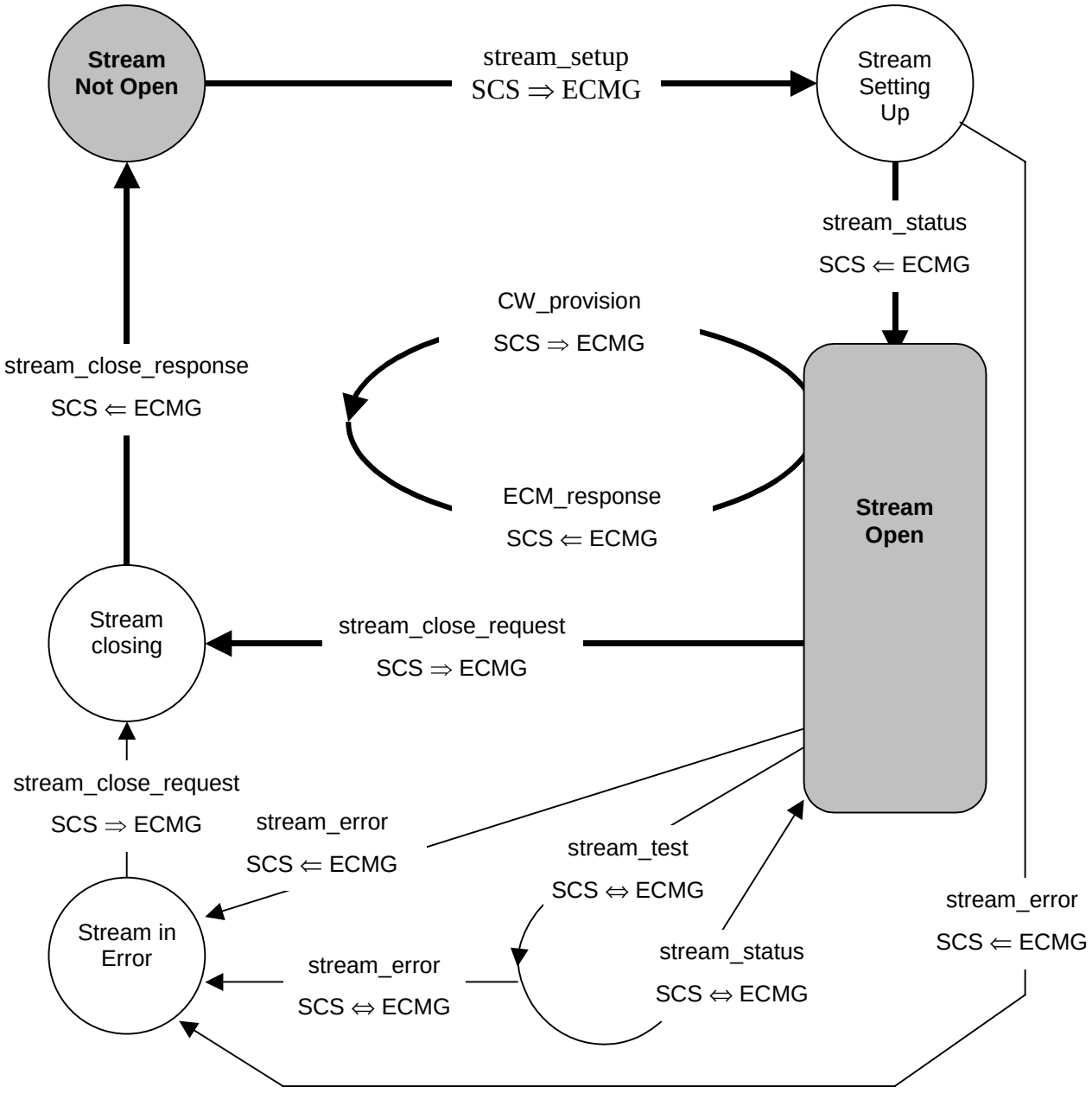


Figure 5: ECMG ⇔ SCS stream state machine

6.1.2.1 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 SCS initializes a stream by sending a **stream_setup message** to the ECMG on the other end of the stream. **Stream_setup** is the only permissible message in the **Stream Not Open** state. Transmission and receipt of **stream_setup** move the state machine to the **Stream Setting Up** state.

6.1.2.2 Stream Setting Up

From this temporary and short-lived state, the **ECMG** 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 ECMG could not open the stream, and that the stream shall be closed by the SCS. One or more error codes explain the failure. Transmission and receipt of **stream_error** move the state machine to the **Stream In Error** state.

6.1.2.3 Stream Open

The stream is open and operational. Five kinds of stream-level messages can be sent while in **Stream Open** state:

- The SCS sends a **CW_provision** message to request the ECMG an ECM. This message carries control word(s), access criteria and cryptoperiod numbers. The ECMG shall respond to each **CW_provision** message with an **ECM_response** message.
- If the SCS 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 SCS or the ECMG 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

6.1.2.4 Stream In Error

This temporary and short-lived state is used only to represent the fact that the SCS or the ECMG has encountered and reported an unrecoverable stream error.

The SCS sends a **stream_close_request** message to the ECMG. Transmission and receipt of **stream_close_request** move the state machine to the Stream Closing state.

6.1.2.5 Stream Closing

This temporary and short-lived state is used only to represent the fact that the SCS has requested closure of the stream.

The ECMG sends a **stream_close_response** message to the SCS, to confirm closure of the stream. Transmission and receipt of **stream_close_response** move the state machine to the **Stream Not Open** state.

The normal and recommended stream closing procedure is the one defined in this clause. However if the ECMG closes the TCP connection, the error management described in clause 10.1 "TCP connection closure by the server" has to be applied.

6.1.3 Summary of messages permissible in each state

Table 1 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 1: message/state cross-reference for the SCS ⇔ ECMG state machines

	Channel states				Stream states (if Channel Open)				
Messages	Not open	Setting up	Open	In error	Not open	Setting up	Open	Closing	In error
channel_setup	X								
channel_status		X			X	X	X	X	X
channel_test			X		X	X	X	X	X
channel_close			X	X	X	X	X	X	X
channel_error		X	X		X	X	X	X	X
stream_setup			→		X				
stream_status			→			X	X		
stream_test			→				X		
stream_close_request			→				X		X
stream_close_response			→					X	
stream_error			→			X	X		
CW_provision			→				X		
ECM_response			→				X		

6.2 Network delay

In any head-end architecture there is the option of connecting CA systems to scramblers at remote sites over a wide area network. In this situation, in order to be more meaningful to the remote scramblers, the *max_comp_time* parameter should take network delay into consideration in addition to the time required to compute the ECM. That is, this parameter should be configured on the CA system to be the sum of the actual time required to compute an ECM plus the worst case network delay expected.

7 EMMG/PDG(MUX protocol)

7.1 State diagram

7.1.1 Channel state machine

This clause presents the channel state machine, which defines the sequence of channel-level messages that shall be used to establish and maintain one channel on one TCP connection.

The channel state machine is found in figure 6. Each state found in this state machine is defined in clauses 7.1.1.1 to 7.1.1.4.

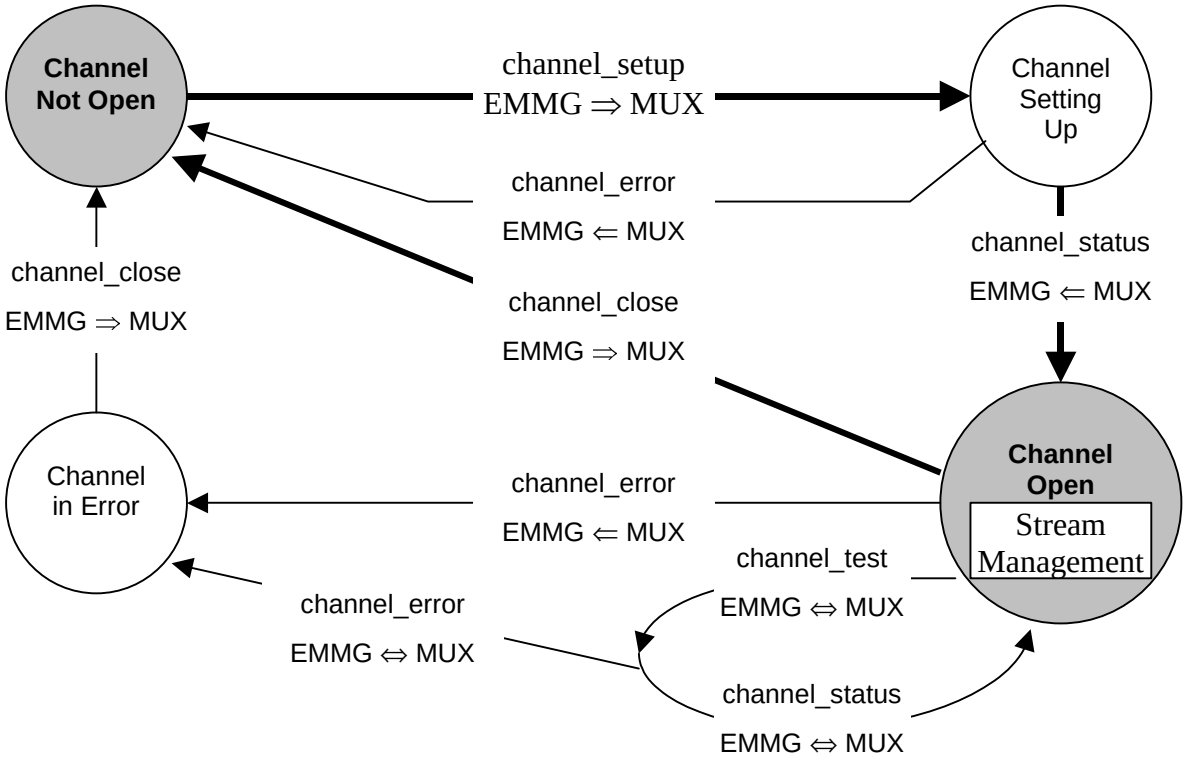


Figure 6: EMMG ⇔ MUX channel state machine

7.1.1.1 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 EMMG initializes a channel by sending a **channel_setup** message to the MUX on the other end of the channel. **Channel_setup** is the only permissible message in the **Channel Not Open** state. Transmission and receipt of **channel_setup** move the state machine to the **Channel Setting Up** state.

7.1.1.2 Channel Setting Up

From this state, the MUX 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. Transmission and receipt of **channel_status** move the state machine to the **Channel Open** state.

The **channel_error** message acknowledges that the MUX could not open the channel; one or more error codes explain the failure. The channel shall be considered by the EMMG and by the MUX as not open. Transmission and receipt of **channel_error** move the state machine to the **Channel Not Open** state.

7.1.1.3 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 clause 7.1.2: 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 EMMG or the MUX 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 MUX encounters an unrecoverable channel error at any other time, it shall send the EMMG 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 EMMG wants to close the channel for any reason, it shall send the MUX 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 the **Channel Open** state.

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

7.1.1.4 Channel In Error

This temporary and short-lived state is used only to represent the fact that the MUX has encountered and reported an unrecoverable channel error. The EMMG shall close the channel.

The EMMG sends a **channel_close** message to the MUX. 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.

7.1.2 Stream state machine

The head-end can establish one or more streams within a channel. This clause 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 EMMG 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 clause 7.1.1.3) for a EMMG to initiate a stream state machine. The channel state machine, as defined in clause 7.1.1, continues to operate in **Channel Open** state during the operation of the stream state machine. Accordingly, both MUX and EMMG 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 7. Each state found in this state machine is defined in clauses 7.1.2.1 to 7.1.2.5.

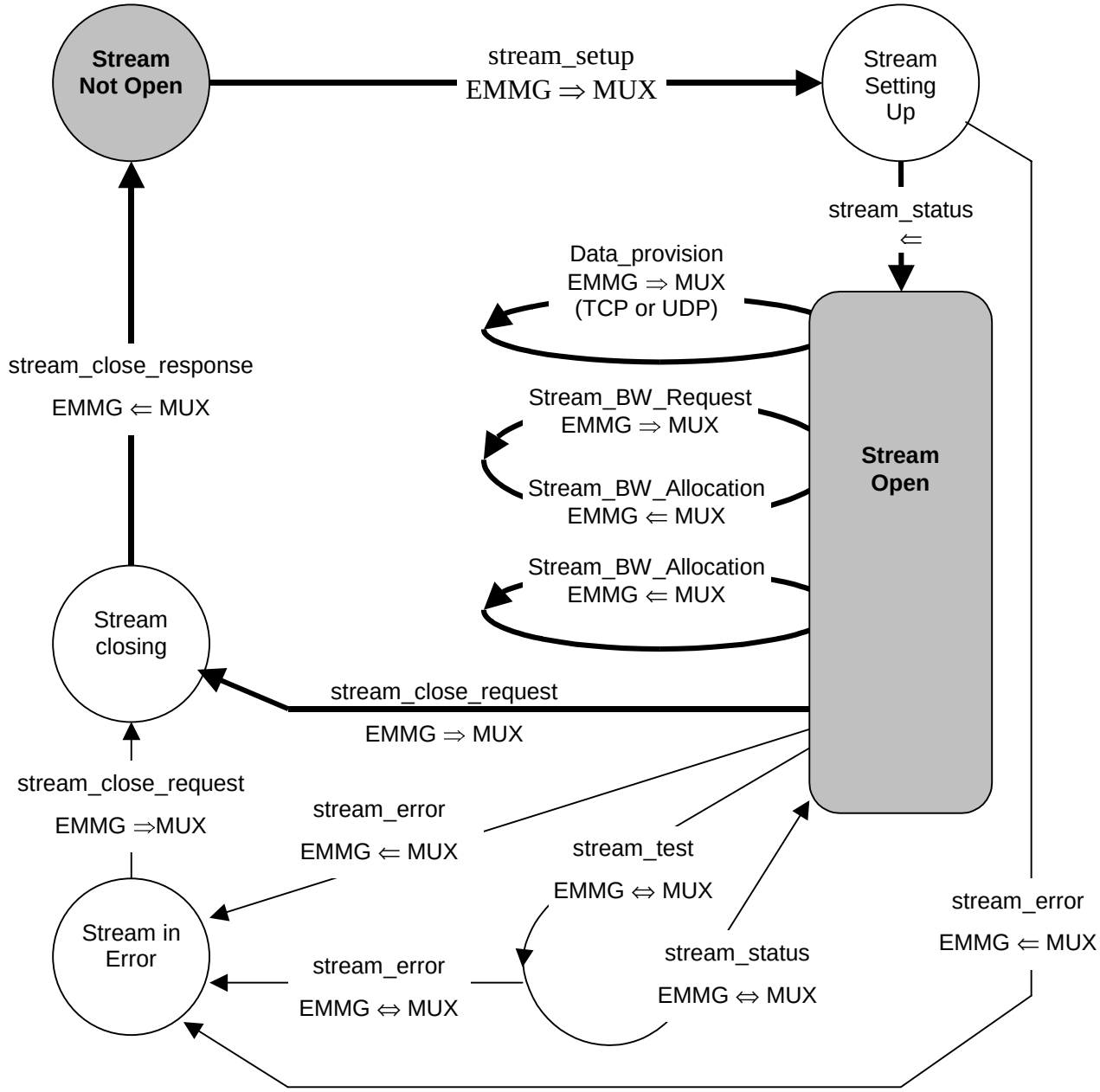


Figure 7: EMMG ⇔ MUX stream state machine

7.1.2.1 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 EMMG initializes a stream by sending a **stream_setup message** to the MUX on the other end of the stream. **Stream_setup** is the only permissible message in the **Stream Not Open** state. Transmission and receipt of **stream_setup** move the state machine to the **Stream Setting Up** state.

7.1.2.2 Stream Setting Up

From this temporary and short-lived state, the **MUX** 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 MUX could not open the stream, and that the stream shall be closed by the EMMG. One or more error codes explain the failure. Transmission and receipt of **stream_error** move the state machine to the **Stream In Error** state.

7.1.2.3 Stream Open

The stream is open and operational. Seven kinds of stream-level messages can be sent while in **Stream Open** state:

- The EMMG sends a **data_provision** message to the MUX. This message is not acknowledged by the MUX. This message can be sent in the current stream on the same TCP connection or over an UDP link.
- The EMMG may send a **stream_BW_request** message to the MUX, in order to be allocated a bandwidth in transport stream for the EMMs. In this message the EMMG gives the requested bandwidth. The MUX shall respond to each **stream_BW_request** message with a **stream_BW_allocation** message in which the allocated bandwidth is given. This does not change the state of the stream state machine. **Stream_BW_request** may be sent at any time from this state.
- The MUX may send a **stream_BW_allocation** message to the EMMG, in order to allocate a bandwidth to the EMMG. In this message the MUX gives the allocated bandwidth. This message is not acknowledged by the EMMG. **Stream_BW_allocation** may be sent at any time from this state or as response to a **stream_BW_request** message. This does not change the state of the stream state machine.
- If the EMMG 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 EMMG or the MUX 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

7.1.2.4 Stream In Error

This temporary and short-lived state is used only to represent the fact that the EMMG or the MUX has encountered and reported an unrecoverable stream error.

The EMMG sends a **stream_close_request** message to the MUX. Transmission and receipt of **stream_close_request** move the state machine to the Stream Closing state.

7.1.2.5 Stream Closing

This temporary and short-lived state is used only to represent the fact that the EMMG has requested closure of the stream.

The MUX sends a **stream_close_response** message to the EMMG, to confirm closure of the stream. Transmission and receipt of **stream_close_response** move the state machine to the **Stream Not Open** state.

The normal and recommended stream closing procedure is the one defined in this clause. However if the MUX closes the TCP connection, the error management described in clause 10.1 "TCP connection closure by the server" has to be applied.

7.1.3 Summary of messages permissible in each state

Table 2 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 2: message/state cross-reference for the EMMG ⇔ MUX state machines

	Channel states				Stream states (if Channel Open)				
Messages	Not open	Setting up	Open	In error	Not open	Setting up	Open	Closing	In error
channel_setup	X								
channel_status		X			X	X	X	X	X
channel_test			X		X	X	X	X	X
channel_close			X	X	X	X	X	X	X
channel_error		X	X		X	X	X	X	X
stream_setup			→		X				
stream_status			→			X	X		
stream_test			→				X		
stream_close_request			→				X		X
stream_close_response			→					X	
stream_error			→			X	X		
stream_BW_request			→				X		
stream_BW_allocation			→				X		
data _provision			→				X		

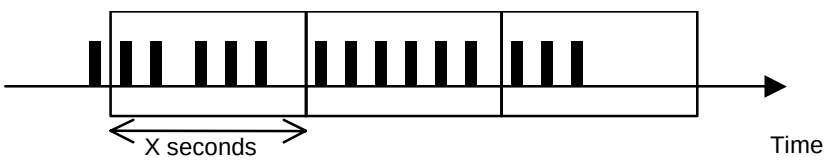
7.2 Datagram insertion timing

7.2.1 Bandwidth definition

The EMMG/PDG protocol includes a bandwidth negotiation mechanism between the CAS and the head-end (see TS 103 197 [6], clause 6.2.1.4). According to this mechanism, on CAS request or on its own initiative, the head-end allocates the bandwidth. The bandwidth parameter is given in kbit/s (see TS 103 197 [6], clause 6.2.2).

Whatever the datagram format is (TS packet or section), the bandwidth parameter in the EMMG/PDG protocol is the bandwidth occupied in the transport stream by EMM/Private Data datagrams, considering 188 byte TS packets (i.e. including header, = 1 504 bits per packet).

The EMMG/PDG protocol bandwidth is evaluated in a X second window applied to the transport stream. The origin of these "bandwidth windows" is arbitrary. The window length X shall be proposed by the Mux vendor and agreed by CAS vendor.



The exceeded bandwidth error is generated according to a sliding average over some bandwidth windows.

7.2.2 EMM/Private Data datagram insertion

When it receives a **data_provision** message the Mux shall insert the datagrams in the TS according to the following rules:

- For the same stream datagrams shall be inserted in TS in the order of their arrival at the Mux.
- Processing of content datagrams in the Mux can introduce a constant delay between their arrival at the Mux and their insertion into the TS. This constant delay should be lower than 250 ms.
- For the same stream, by referring to the relative time position of datagrams at the input of the Mux, the jitter generated by the Mux in TS shall be lower than 5 ms.

These rules are depicted in figure 8.

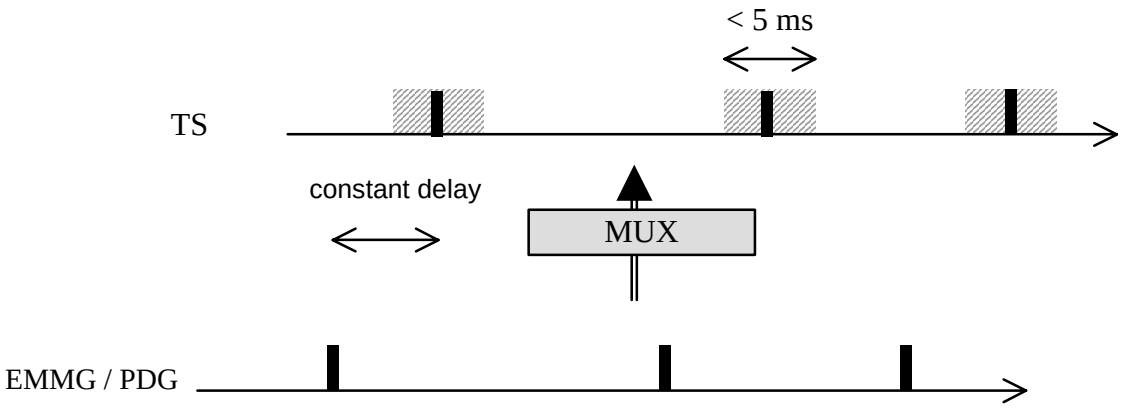


Figure 8: Timing of datagram insertion into TS

7.3 UDP provision without TCP control layer

According to TS 103 197 [6], when EMMG/PDG datagram provision is performed over UDP, the TCP control layer is mandatory between each MUX and the EMMG/PDG.

In particular, a MUX is not allowed to process any datagrams received when a valid TCP control layer connection was not previously established with the EMMG/PDG whose client_id is given in these datagrams.

However if one or several TCP control links are broken or missing the EMMG/PDG shall continue to send data as long as there is at least one valid control path. In the same time the EMMG/PDG can try to re-establish the broken TCP control link(s), possibly in applying specific procedure of Mux redundancy.

7.4 Datagram packetization

Datagram packetization shall comply with packetization rules defined in relevant MPEG2 and DVB standards.

When multiple EMM/PD sections are sent by the EMMG/PDG in a single **data_provision** message (i.e. in section format) the MUX shall use as many TS packets and shall pack as many sections into each TS packet as possible.

If a CAS needs particular section packetization rules, for STB constraints compliance, the EMMG/PDG shall perform the section packetization prior to providing the MUX with datagrams in TS packet format.

The use of a proprietary protocol between the EMMG/PDG and the MUX for defining section packetization rules does not comply with TS 103 197 [6].

8 C(P)SIG⇔(P)SIG

8.1 C(P)SIG protocol profiles

The C(P)SIG protocol allows a CAS for providing descriptors to a head-end. The CAS is responsible for deciding whether a descriptor has to be inserted or deleted, for defining what this descriptor is, for indicating where and when this descriptor has to be inserted. The head-end is responsible for inserting the descriptor in PSI or SI tables, for providing the CAS with (P)SI table contents and for warning the CAS about events occurring in services, in ECM streams or in EMM/PD streams.

Any CAS-related descriptor (CA-descriptors or private descriptors) can be provided by the CAS to the head-end and any PSI or SI table can be addressed as target of descriptor insertion. To allow this the C(P)SIG protocol combines five transaction types between head-end and CAS: trigger, (P)SI table provisioning, service change, flow PID provisioning, custom (P)SI descriptor insert.

To make easier the implementation of the C(P)SIG protocol, some profiles are defined allowing a CAS and a head-end for using a subset of C(P)SIG protocol features:

- low-level profile: CA-descriptors in PMT and in CAT;
- high-level profile: all C(P)SIG protocol features.

8.2 Low-LEVEL profile definition

In this low-level profile:

- only CA_descriptors for ECM in PMT and for EMM in CAT are addressed by a CAS and by a head-end;
- only the connection-oriented protocol is supported;
- only CPSIG and PSIG functional boxes are implemented.

8.2.1 CA-descriptor for ECM streams

In the low-level profile two transaction types are used:

- trigger transaction type: the CAS is warned about the existence of ECM streams, about access condition change and about PID change (ECM setup, ECM closure, AC Change, PID Change);
- descriptor Insertion transaction type: the CAS provides the head-end with CA-descriptors for PMT.

8.2.1.1 Messages

The following messages in TS 103 197 [6], clause 8.3 are not used:

- **table_request, table_response** for Table Provisioning transaction type (see TS 103 197 [6], clause 8.2.4);
- **stream_service_change** for Service Change transaction type (see TS 103 197 [6], clause 8.2.6);
- **PID_provision_request, PID_provision_response** for PID Provisioning transaction type (see TS 103 197 [6], clause 8.2.7).

8.2.1.2 Trigger transaction type (see TS 103 197 [6], clause 8.2.3)

ECM stream setup, ECM stream closure, flow PID change, access criteria change triggers are used, corresponding to the following values of trigger types:

From Table 30 of TS 103 197 [6]: Trigger types

trigger cause	trigger_list bit #	trigger type
ECM stream set up	2	0x00000004
access criteria change	3	0x00000008
ECM stream closure	4	0x00000010
flow PID change	5	0x00000020
combination of any previous	3 and 5	0x00000028

In **trigger** message (see TS 103 197 [6], clause 8.3.4.11), the parameter table becomes:

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

The head-end shall triggers the CPSIG:

- for ECM setup: each time a stream is opened on the ECMG protocol;
- for ECM closure: each time a stream is closed on the ECMG protocol;
- for ECM setup: for each already active ECM stream when it connects the C(P)SIG.

8.2.1.3 Descriptor Insertion transaction type (see TS 103 197 [6], clause 8.2.5)

The descriptor Insertion transaction type is limited as follows:

- only PMT 1st loop and PMT 2nd loop are addressed: in table 32 of TS 103 197 [6] only location_id values 0x02 and 0x03 are used;
- only CA_Descriptors are provided/inserted (no private_data_specifier parameter in TS 103 197 [6], clause 8.3.4.14);
- both values of CA_descriptor_insertion_mode are allowed (see TS 103 197 [6], clause 8.3.3.2);
- both insertion delay type (immediate and synchronized) are allowed (see TS 103 197 [6], clause 8.3.4.14).

In **descriptor_insert_request** message (see TS 103 197 [6], clause 8.3.4.14), the parameter table becomes:

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 (= 0x02 or 0x03)
service_id	1 (see table 32)
ES_id	0 or 1 (see table 32)
descriptor	0 to n

8.2.2 CA-descriptor for EMM streams

In the low-level profile three transaction types are used:

- trigger transaction type: the CAS is warned about PID change;
- flow PID Provisioning transaction type: the CAS requests for the PID allocated to an EMM flow;
- descriptor Insertion transaction type: the CAS provides the head-end with CA-descriptors for CAT.

8.2.2.1 Messages

In TS 103 197 [6], clause 8.3 the following messages are not used:

- **table_request, table_response** for Table Provisioning transaction type (see TS 103 197 [6], clause 8.2.4);
- **stream_service_change** for Service Change transaction type (see TS 103 197 [6], clause 8.2.6).

8.2.2.2 Trigger transaction type (see TS 103 197 [6], clause 8.2.3)

Only flow PID change trigger is used. Accordingly only the following value of trigger type is used:

From Table 30 of TS 103 197 [6]: Trigger types

trigger cause	trigger_list bit #	trigger type
flow PID change	5	0x00000020

In **trigger** message (see TS 103 197 [6], clause 8.3.4.11), the parameter table becomes:

Parameter	Number of instances in message PID change
transaction_id	1
custom_channel_id	1
custom_stream_id	1
trigger_id	1
trigger_type	1
flow_PID_change_related_data	1

8.2.2.3 Descriptor Insertion transaction type (see TS 103 197 [6], clause 8.2.5)

The descriptor Insertion transaction type is limited as follows:

- only CAT is addressed: in table 32 of TS 103 197 [6] only location_id value 0x01 is used;
- only CA_Descriptors are provided/inserted (no private_data_specifier parameter in TS 103 197 [6], clause 8.3.4.14);
- both values of CA_descriptor_insertion_mode are allowed (see TS 103 197 [6], clause 8.3.3.2);
- both insertion delay type (immediate and synchronized) are allowed (see TS 103 197 [6], clause 8.3.4.14).

In **descriptor_insert_request** message (see TS 103 197 [6], clause 8.3.4.14), the parameter table becomes:

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 (= 0x01 for CAT)
descriptor	0 to n

8.2.2.4 Flow PID Provisioning transaction type (see TS 103 197 [6], clause 8.2.7)

The flow_type shall be 0x00 for EMM (see TS 103 197 [6], clauses 8.3.4.16 and 8.3.4.17).

8.3 High-level profile definition

In this high-level profile, all features of the C(P)SIG protocol as described in the specification TS 103 197[6] are available.

9 SIMF based protocols - Role of NMS

9.1 MIB use for monitoring only

The SIM module (see TS 103 197 [6], clause 7.3) is defined to allow a NMS for configuring and monitoring a CAS device considering the Simulcrypt specific parameters.

One can meet real operational cases where a CAS device (ECMG, EMMG, CPSIG) is directly configured by the CAS provider independently from any NMS, but where the current status of the CAS device is wanted to be displayed by NMS.

According to TS 103 197 [6], clause 7.3, all access rights in SIM can be further restricted by individual MIB views if so desired in particular implementations.

In particular, a CAS device may restrict access to all its parameters to read-only mode, for allowing the NMS to access only to the current status of the CAS device.

10 Error management

10.1 Error processing in all connection-oriented protocols

Some error processing are described in TS 103 197 [6], the others are recommended below.

Error case	Condition	Processing
"unknown command" See message_type definition in TS 103 197 [6], clause 4.4.1	A message_type value is not defined in the protocol. A message_type value use does not comply with the state diagram of the protocol.	"Unknown message types shall be ignored by the receiving entity".
"unknown parameter" See parameter_type definition in TS 103 197 [6], clause 4.4.1	A parameter_type value is not defined in the protocol. F.i.: parameter 0x000F in ECMG protocol. A parameter_type value does not comply with the current protocol version. F.i.: ECM_id in V1-ECMG protocol. A parameter_type value use does not comply with the message.	"The data associated with that parameter is discarded and the remaining message is processed".
"inconsistencies"	A parameter value, which should remain fixed during the current protocol phase, has changed. Particularly: The protocol_version has changed after channel_setup. The channel_id value is not the same as the one in channel_setup on the current TCP connection (except in channel_setup message). The stream_id value in a message is unknown on the current channel (except in stream_setup message). This condition is also applicable to parameters specific to each protocol (see clause 10.2, 10.3 and 10.4).	Recommended: The receiving entity sends an error message and the client (SCS, EMMG, (P)SIG) closes the channel and the connection. Other behaviour: The receiving entity sends an error message and processes the message with the previous value of the concerned parameter. To be avoided: The receiving entity ignores the value change and processes the message with the new value of the concerned parameter.
"unknown channel_id value"	A channel_id value is not known by the receiving entity (except in a channel_setup message).	Recommended: the receiving entity sends an error message on the received channel_id. Other behaviour: the message is ignored.
"TCP connection closure by the server"	The server closes the TCP connection.	The client closes the TCP connection on its side and considers that all current channels and streams are closed. Then the client shall connect again the server or its backup.
"missing mandatory DVB parameter"	A mandatory DVB parameter is missing in a message sent by a server or by a client.	The message is rejected by the receiver and the receiver sends an error message.

10.2 Specific inconsistencies in ECMG protocol

ECM_id parameter value in a **stream_status** message is different from the one given in the **stream_setup** message for the same stream_id (V2 only).

In a **channel_status** message the parameter values relative to the ECMG are not the same as those given by the ECMG in the **channel_status** message as response to the **channel_setup** message: section_TSPkt_flag, delay_start, delay_stop, AC_delay_start, AC_delay_stop, transition_delay_start, transition_delay_stop, ECM_rep_period, max_streams, min_CP_duration, lead_CW, CW_per_msg, max_comp_time.

In a **stream_status** message the access_criteria_transfer_mode parameter value is not the same as the one given by the ECMG in the **stream_status** message as response to the **stream_setup** message.

The value of CP_number parameter in a **ECM_response** message is not the same as the one in the associated **CW_provision** message.

The datagram length is not a multiple of 188 bytes when section_TSpkt_flag is set to "TS packet" (=1). This case could give as well the "Inconsistent length of a DVB parameter" error.

10.3 Specific inconsistencies in EMMG protocol

The client_id value is not the same as the one in **channel_setup** on the current TCP connection.

Data_id parameter value in a **stream_status** message or in a **data_provision** message is different from the one given in the **stream_setup** message for the same stream_id (V2 only).

In a **channel_status** message the section_TSPkt_flag parameter values is not the same as the one given by the EMMG in the **channel_setup** message.

In a **stream_status** message the data_type parameter value is not the same as the one given by the EMMG in the **stream_setup** message.

The datagram is not 188 byte long when section_TSpkt_flag is set to "TS packet" (=1). This case could give as well the "Inconsistent length of a DVB parameter" error.

10.4 Specific inconsistencies in C(P)SIG protocol

The protocol_version value is not 2.

The transaction_id parameter value in a response is different from the one of the associated request. If this parameter value is out of range, the "Invalid value for a DVB parameter" error is issued.

In a **channel_status** message the (P)SIG or C(P)SIG specific parameter values are not the same as those given in the **channel_setup** message nor in the first **channel_status** response.

In a **stream_status** message the list of service_id parameter values is not the same as the one given by the PSIG in the **stream_setup** message.

(P)SIG and C(P)SIG are not compliant.

11 Redundancy management

11.1 Uniqueness concept

This clause sums up uniqueness rules for some parameters as given in TS 103 197 [6].

11.1.1 ECMG protocol

Super_CAS_id

(TS 103 197 [6], clause 5.3) The Super_CAS_ID […] shall identify uniquely a (set of) ECMG(s) for a given SCS […]

ECM_channel_id

(TS 103 197 [6], clause 5.1.2).There is always one (and only one) channel per TCP connection.

(TS 103 197 [6], clause 5.3) The ECM_channel_ID is allocated by the SCS and uniquely identifies an ECM channel across all connected ECMGs

ECM_stream_id

(TS 103 197 [6], clause 5.3) This identifier uniquely identifies a ECM stream within a channel. It is allocated by the SCS prior to stream set-up.

ECM_id

(TS 103 197 [6], clause 5.3) 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 clause 8.2.7.

(TS 103 197 [6], clause 8.2.7) […] 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.

[…]

11.1.2 EMMG/PDG protocol

client_id

(TS 103 197 [6], clause 6.2.3) The client_id […] 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_channel_id

(TS 103 197 [6], clause 6.2.3) This identifier uniquely identifies a EMM/Private Data channel within a client_ID.

"One and only one channel per TCP connection" is not explicitly written in the standard, but:

> (channel establishment clause 6.2.1.2) […] In case of a rejection or a failure during channel set-up 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;
>
> (channel closure clause 6.2.1.6) […] This is done by means of a Channel_close message sent by the EMMG/PDG. Subsequently, the connection shall be closed by both sides.

data_stream_id

(TS 103 197 [6], clause 6.2.3) This identifier uniquely identifies a EMM/Private Data stream within a channel.

data_id

(TS 103 197 [6], clause 6.2.3) 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 clause 8.2.7.

(TS 103 197 [6], clause 8.2.7: see above).

11.2 Basic redundancy scenarios

11.2.1 Definitions

Basic redundancy of a device

"Basic redundancy of a device" means that a spare device is available but not yet in operational use when a failure occurs. A spare device can be in various states from staying on a shelf up to being ready for moving to operational use.

CAS Manager

The CAS Manager is the function allowing the operator for configuring/monitoring a CAS device.

head-end Manager

The head-end Manager is the function allowing the operator for configuring/monitoring a head-end device.

11.2.2 ECMG ⇔ SCS

11.2.2.1 SCS redundancy

The SCS redundancy is managed by the head-end Manager.

When the SCS fails or is stopped, current links between SCS and ECMG are broken: stream(s), channel(s) and TCP connection(s) are closed.

After TCP connection closure, the ECMG shall be ready for being connected by any SCS.

The spare SCS is given internally by the head-end the current context of ECMG generation. In particular, it is given the IP address and the TCP port of the ECMG. Then the head-end Manager activates this SCS which establishes all needed connections (TCP, channel(s), stream(s)) to the ECMG.

11.2.2.2 ECMG redundancy

The ECMG redundancy is managed by both head-end Manager and CAS Manager.

When the ECMG fails or is stopped, current links between SCS and ECMG are broken: stream(s), channel(s) and TCP connection(s) are closed.

After TCP connection closure, the SCS is given by the head-end Manager the IP address and the TCP port of a spare ECMG; the IP address and the TCP port may remain the same if the spare ECMG replaces physically the failing one. Then the SCS connects this spare ECMG according to the broken ECM generation context.

The spare ECMG shall be activated by the CAS Manager before being connected by the SCS.

Until the spare ECM can provide the SCS with ECMs, the SCS shall preserve the scrambling status of the service by choosing and applying a temporary policy, as repeating the last valid CWs and ECMs or as working still normally with the other ECMGs in Simulcrypt configuration.

11.2.3 EMMG ⇔ MUX

11.2.3.1 EMMG redundancy

The EMMG redundancy is managed by the CAS Manager.

When the EMMG fails or is stopped, current links between EMMG and MUX are broken: stream(s), channel(s) and TCP connection(s) are closed. If any the UDP port remains available.

After TCP connection closure, the MUX shall be ready for being connected by any EMMG.

The spare EMMG is given by the CAS Manager the current context of EMMG generation. In particular, it is given the IP address + TCP port, and the IP address + UDP port if used, of the MUX. Then the CAS Manager activates this EMMG which establishes all needed connections (TCP, channel(s), stream(s)) to the MUX.

11.2.3.2 MUX redundancy

The MUX redundancy is managed by both CAS Manager and head-end Manager.

When the MUX fails or is stopped, current links between EMMG and MUX are broken: stream(s), channel(s) and TCP connection(s) are closed.

After TCP connection closure, the EMMG is given by the CAS Manager the IP address, the TCP port, and the IP address + UDP port if used, of a spare MUX; IP addresses and ports may remain the same if the spare MUX replaces physically the failing one. Then the EMMG connects this spare MUX according to the broken EMM generation context.

The spare MUX shall be activated by the head-end manager before being connected by the EMMG.

List of figures

History

Document history		
V1.1.1	April 2002	Publication