TS 101 197-1 V1.1.1 (1997-06)

Technical Specification

Digital Video Broadcasting (DVB);
DVB SimulCrypt;
Part 1: Head-end architecture and synchronization

European Broadcasting Union

Union Européenne de Radio-Télévision

European Telecommunications Standards Institute

Reference

DTS/JTC-00DVB-62-1 (b3090icr.PDF)

Keywords

Broadcasting, digital, video, DVB, TV

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by the 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 was established in 1990 to co-ordinate the drafting of ETSs in the specific field of broadcasting and related fields. Since 1995 the JTC 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
Case Postale 67
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Digital Video Broadcasting (DVB) Project

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

This Technical Specification (TS) is part 1 of a multi-part document covering Digital Video Broadcasting (DVB); DVB SimulCrypt, as identified below:

Part 1: "Head-end architecture and synchronization";

Part 2: "Extended interoperability and control".

Parts 1 and 2 together address the requirements for interoperability between two or more Conditional Access (CA) systems at a head-end.

Part 1: specifies the system architecture, timing relationships and messaging structures; and

Part 2: specifies 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 SimulCrypt Synchronizer (SCS) could be in the Multiplexer (MUX) or the SCS and MUX could be built independently. Neither architecture is mandated.

1.1 Common scrambling algorithm

The DVB SimulCrypt system is based on the concept of a shared scrambling and descrambling method. No problems were noted so far on the possible constraints which the DVB SimulCrypt architecture might impose on the use of a shared scrambling and descrambling method.

1.2 Language

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

2 Normative references

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] ISO/IEC 13818-1 (1994): "Information Technology - Generic Coding of Moving Pictures and Associated Audio Recommendation H.222.0 (Systems)".

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

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

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

3 Definitions and abbreviations

3.1 Definitions

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

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

CA_system_id: Conditional Access (CA) System IDs are defined in table 3 "CA_system_ID" of ETR 162 [3].

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

channel: An application specific representation of an open Transport Control Protocol (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: A software entity on a host making use of one or more resources offered by a server.

Conditional Access (CA) system: A system to control subscriber access to broadcast services and events e.g. Videoguard, Eurocrypt.

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

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

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

Entitlement Control Message (ECM): Private Conditional Access (CA) information which carries the CW 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 (CA) information which, for example, specifies the authorization levels of subscribers or groups of subscribers for services or events.

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

forbidden: The term "forbidden" when used in the present document indicates that the value shall never be used.

generator: A component producing data.

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

host head-end: A 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]. Systems coding is defined in part 1. Video coding is defined in part 2. Audio coding is defined in part 3.

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

Multiplexer (MUX): See subclause 4.2.10.

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 agreements.

reserved: The 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".

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

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

Scrambler (SCR): See subclause 4.2.11.

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

service: A 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: "SimulCrypt" is a process that facilitates using several Conditional Access (CA) systems in parallel, in conjunction with the DVB common scrambling algorithm, to control access to pay-TV services. SimulCrypt involves the inter-operation of two or more CA streams in a DVB environment. The DVB SimulCrypt addresses specifically the requirements for interoperability between two or more CA systems at a head-end.

SimulCrypt Synchronizer (SCS): The logical component that acquires CW, Entitlement Control Messages (ECM) and synchronizes their play-out for all the CA systems connected.

stream: An 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, etc.) are used to tag messages belonging to a particular stream.

Transport Stream (TS): A TS is a data structure defined in ISO/IEC 13818-1 [1]. It is the basis of DVB related standards.

3.2 Abbreviations

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

AC	Access Criteria
ASCII	American Standard Code for Information Interchange
bslbf	bit string, left bit first
BW	BandWidth
CA	Conditional Access
CAT	Conditional Access Table
CP	Crypto Period
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CW	Control Word
CWG	Control Word Generator
DNS	Domain Name Server
DVB	Digital Video Broadcasting
ECM	Entitlement Control Message
ECMG	Entitlement Control Message Generator
EIS	Event Information Scheduler
EIT	Event Information Table
EMM	Entitlement Management Message
EMMG	Entitlement Management Message Generator
ID	Identifier
IP	Internet Protocol
ISO	International Organization for Standardization
LSB	Least Significant Bit
MJD	Modified Julian Date
MPEG	Moving Pictures Expert Group
MSB	Most Significant Bit
MUX	Multiplexer
NIT	Network Information Table
OSI	Open Systems Interconnect
PAT	Program Association Table
PDG	Private Data Generator
PID	Packet Identifier
PMT	Program Map Table
PSI	Program Specific Information
RFC	Request For Comment
SCR	(DVB compliant) Scrambler
SCS	SimulCrypt Synchronizer
SDT	Service Description Table
SI	Service Information
SIG	Service Information Generator
SMS	Subscriber Management System
ST	Stuffing Table
STB	Set Top Box
tcimsbf	two's complement integer MSB (sign) bit first
TCP	Transport Control Protocol
TDT	Time and Date Table
TLV	Type, Length, Value
TS	Transport Stream
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 i.e. SMS.

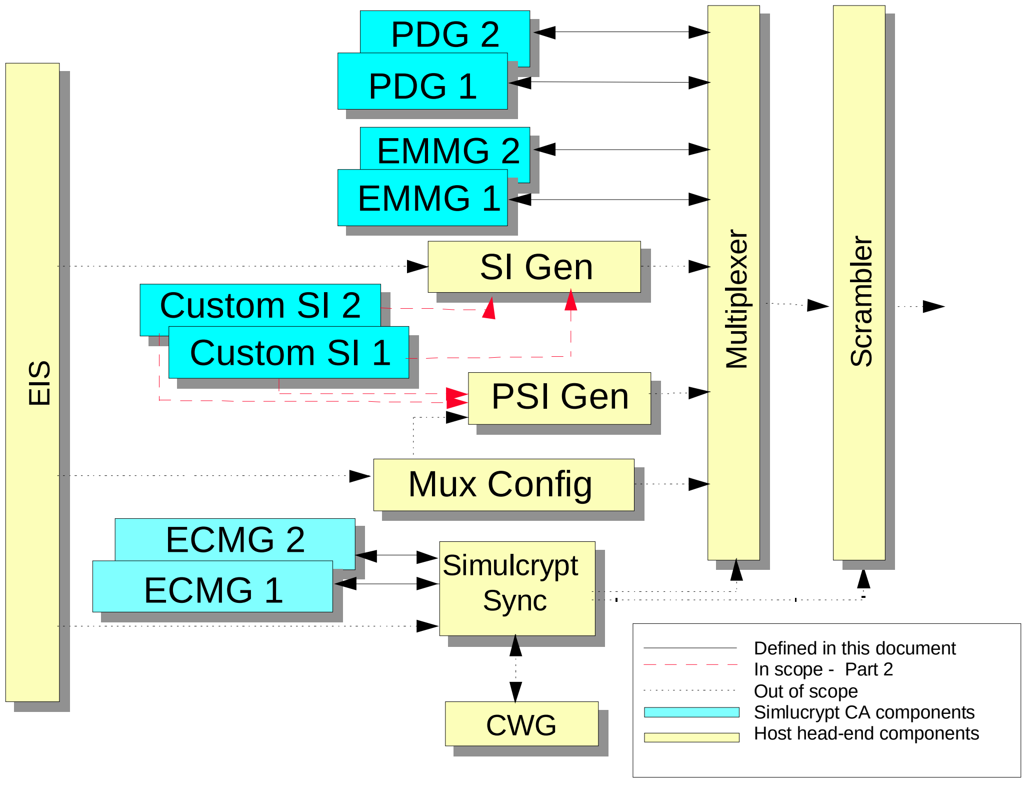


Figure 1: System architecture

The DVB SimulCrypt system architecture illustrated above is divided into two 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: The EMMGs, PDGs and custom SI generators are not necessarily required in a DVB SimulCrypt system.

4.2 Description of components

4.2.1 Event Information Scheduler (EIS)

In the DVB SimulCrypt system architecture diagram (subclause 4.1), the EIS is the functional unit in charge of holding all the 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 (SCS) is to:

- establish TCP connections with ECMGs and set-up one channel per connection;
- set-up streams within channels as needed and allocate ECM_stream_ID values;
- get the CWs 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 CPs 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 CP.

4.2.3 Entitlement Control Message 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 Entitlement Management Message 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 (figure 1) to highlight the fact that the EMMG to MUX interface can be used for EMM's and other CA related private data. The PDG initiates connections to the MUX.

4.2.6 Custom Service Information Generator (SIG)

This component is responsible for generating private SI. It interfaces to both the SI and PSI generators. These interfaces are not defined by the present document.

4.2.7 Multiplexer configuration (MUX Config)

This component is responsible for configuring the MUX and providing a link to the Program Specific Information (PSI) generator for PSI construction and play-out. The interfaces "MUX Config and MUX" and "MUX Config and PSI Generator" are not defined by the present document.

4.2.8 SI generator

This component is responsible for generating the SI (see ETS 300 468 [2]) for the system. The SI server takes its primary data from the EIS and supplementary data from the Custom SI servers supplied by the CA providers. The interfaces between the EIS and SI Server, and SI Server and MUX are not defined by the present document.

4.2.9 Program Specific Information (PSI) generator

This component is responsible for generating the PSI (see ISO/IEC 13818-1 [1]) for the system. The PSI server takes its primary data from MUX Config and supplementary data from the Custom SI servers supplied by the CA providers.

4.2.10 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 (TS). The input data can be transport packets, MPEG sections or raw data. The exact functionality of a MUX is implementation specific. For the purpose of the present document, the MUX shall be able to communicate with the SimulCrypt Synchronizer (SCS), and to accept connections with Entitlement Management Message Generators (EMMG) with the interface defined.

4.2.11 Scrambler (SCR)

The related interfaces are not defined by the present document.

4.2.12 Control Word Generator (CWG)

The related interfaces are not defined by the present document.

5 Description of interfaces

5.1 ECMG <> SCS

5.1.1 Channel specific messages

These messages are defined further in subclause 7.1.3. This interface shall carry the following channel messages:

- Channel_setup;
- Channel_test;
- Channel_status;
- Channel_close;
- Channel_error.

5.1.2 Stream specific messages

These messages are defined further in subclause 7.1.4. The interface shall carry the following stream messages:

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

5.1.3 Interface principles

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).

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 chose with which ECMG the connection will be opened based on either a redundancy policy or resource available.

5.1.3.1 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 set-up 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.

5.1.3.2 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 set-up 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.

5.1.3.3 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.

5.1.3.4 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.

5.1.3.5 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.

5.1.3.6 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.

5.1.3.7 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 EMMG <> MUX

5.2.1 Channel specific messages

These messages are defined further in subclause 7.2.3. The interface shall carry the following channel messages:

- Channel_setup
- Channel_test
- Channel_status
- Channel_close
- Channel_error

5.2.2 Stream specific messages

These messages are defined further in subclause 7.2.4. The interface shall carry the following stream messages:

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

5.2.3 Interface principles

For this interface, the EMMG is the client and the MUX is the server.

5.2.3.1 Channel establishment

The EMMG 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 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 shall close the TCP connection.

5.2.3.2 Stream establishment

The EMMG 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.

Once the connection, channel and stream have been correctly established the EMM will be transferred. It can be transferred as sections or as TS packets. The EMMG indicates at channel set-up which kind of data object will be used.

5.2.3.3 Bandwidth allocation

The interface allows bandwidth negotiation between the EMMG and the MUX. This is not mandatory. During stream set-up the EMMG 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 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.

5.2.3.4 Stream closure

Stream closure is always initiated by the EMMG. This can occur when an EMM 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.

5.2.3.5 Channel closure

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

5.2.3.6 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.

5.2.3.7 Unexpected connection loss

Both EMMG 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.

5.2.3.8 Handling data inconsistencies

If the MUX detects an inconsistency it shall send an error message to the EMMG. If the EMMG receives such a message or detects an inconsistency it should close the connection. The EMMG (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.

5.3 PDG <> MUX

The interface used is the EMMG<>MUX interface.

5.4 Custom SI Generator <> PSI Generator

Proprietary, not defined by the present document.

5.5 Custom SI Generator <> SI Generator

Proprietary, not defined by the present document.

5.6 EIS <> SI Generator

Proprietary, not defined by the present document.

5.7 SI Generator <> MUX

Proprietary, not defined by the present document.

5.8 EIS <> MUX Config

Proprietary, not defined by the present document.

5.9 MUX Config <> PSI Generator

Proprietary, not defined by the present document.

5.10 PSI Generator <> MUX

Proprietary, not defined by the present document.

5.11 MUX <> SCR

Proprietary, not defined by the present document.

5.12 SCR onward

Proprietary, not defined by the present document.

5.13 SCS <> MUX

Proprietary, not defined by the present document.

5.14 SCS <> SCR

Proprietary, not defined by the present document.

5.15 SCS <> CWG

Proprietary, not defined by the present document.

5.16 EIS <> SCS

Proprietary, not defined by the present document.

6 Generic message description

6.1 Generic message structure

On all interfaces, the messages shall have the following generic structure:

```
generic_message
{
   protocol_version   1 bytes
   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
   }
}
```

protocol_version: An 8-bit field identifying the protocol version. It shall have the value 0x01.

message_type: A 16-bit field identifying the type of the message. The list of message type values is defined in subclause 6.2 Unknown message types shall be ignored by the receiving entity.

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

parameter_type: This 16-bit field specifies the type of the following parameter. The list of parameter type values are defined in the interface specific clauses 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: This 16-bit field specifies the number of bytes of the following parameter_value field.

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

6.2 Message_type values

Message_type value	Relevant interface	Message type
0x0000	DVB reserved	DVB reserved
0x0001	ECMG<>SCS	Channel_set-up
0x0002	ECMG<>SCS	Channel_test
0x0003	ECMG<>SCS	Channel_status
0x0004	ECMG<>SCS	Channel_close
0x0005	ECMG<>SCS	Channel_error
0x0006 to 0x0010	DVB reserved	DVB reserved
0x0011	EMMG<>MUX	Channel_set-up
0x0012	EMMG<>MUX	Channel_test
0x0013	EMMG<>MUX	Channel_status
0x0014	EMMG<>MUX	Channel_close
0x0015	EMMG<>MUX	Channel_error
0x0016 to 0x0020	DVB reserved	DVB reserved
0x0101	ECMG<>SCS	Stream_set-up
0x0102	ECMG<>SCS	Stream_test
0x0103	ECMG<>SCS	Stream_status
0x0104	ECMG<>SCS	Stream_close_request
0x0105	ECMG<>SCS	Stream_close_response
0x0106	ECMG<>SCS	Stream_error
0x107 to 0x110	DVB reserved	DVB reserved
0x0111	EMMG<>MUX	Stream_set-up
0x0112	EMMG<>MUX	Stream_test
0x0113	EMMG<>MUX	Stream_status
0x0114	EMMG<>MUX	Stream_close_request
0x0115	EMMG<>MUX	Stream_close_response
0x0116	EMMG<>MUX	Stream_error
0x0117	EMMG<>MUX	Stream_BW_request
0x0118	EMMG<>MUX	Stream_BW_allocation
0x0119 to 0x0120	DVB reserved	DVB reserved
0x0201	ECMG<>SCS	CW_provision
0x0202	ECMG<>SCS	ECM_response
0x0203 to 0x0210	DVB reserved	DVB reserved
0x0211	EMMG<>MUX	data_provision
0x0212 to 0x0220	DVB reserved	DVB reserved
0x0221-0x7FFF	DVB reserved	DVB reserved
0x8000-0xFFFF	User defined	User defined

7 Interface specific message description

7.1 ECMG <> SCS

7.1.1 Parameter_type values

Parameter_type value	Parameter type	Type/units	Length (bytes)
0x0000	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 × 100 ms	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 × 100 ms	2
0x0011	access_criteria_transfer_mode	Boolean	1
0x0012	CP_number	uimsbf	2
0x0013	CP_duration	uimsbf/n × 100 ms	2
0x0014	CP_CW_Combination	---	10 (2 + 8)
	CP	uimsbf	2
	CW	uimsbf	8
0x0015	ECM_datagram	user defined	variable
0x0016 to 0x006F	DVB reserved	DVB reserved	DVB reserved
0x7000	Error_status	see subclause 7.1.5	2
0x7001	Error_information	user defined	variable
0x7002 to 0x7FFF	DVB reserved	-	-
0x8000 to 0xFFFF	User defined		

7.1.2 Parameter semantics

AC_delay_start: This parameter shall be used in place of the delay start parameter for the first CP following a change in AC.

AC_delay_stop: This parameter shall be used in place of the delay stop parameter for the last CP 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 data base, 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 changes. If it equals 1, it indicates that the ECMG requires the access_criteria parameter be present in each CW_provision message.

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 5.1. The CA_subsystem_ID is defined by the user, it is private.

CP_CW_combination: This 10-byte parameter is the concatenation of the CP number the CW is attached to and the CW itself. The parity (odd or even) of the CP number is equal to the parity of the corresponding CW (see ETR 289 [4]).

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

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

CW_per_msg: The number of CWs needed by the ECMG per CW provision message. If this value is "y" and lead_CW is "x", each CW provision message attached to CP "n" will contain all CWs from period (*n* + 1 + x-y) to period (*n* + x). CWs are carried with their CP 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 CW to generate an ECM, it shall by definition specify at least one lead_CW. However, since it may buffer its own CWs, it can set CW_per_msg to one. By doing this, it always receives the CW for the *next* CP and accessing the CW for the current CP from memory (a previous provision message). Alternatively, it may specify 2 CW_per_msg and have both CWs 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 CWs 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 CP, 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 CP. 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 set-up.

delay_stop: This signed integer represents the amount of time between the end of a CP, 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 CP. 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 set-up.

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

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 CP. The ECM datagram shall comply with ETR 289 [4].

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 a ECM stream within a channel. It is allocated by the SCS prior to stream set-up.

Error_status: See subclause 7.1.5.

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 CWs required in advance to build an ECM. If this value is "x" the ECMG requires CWs up to CP number "*n* + x" to build the ECM attached to CP "*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 CW to generate an ECM, lead_CW would be 1. In other words, it defines the most future CW 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 set-up. 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 set-up. A value of 0 means that this maximum is not known.

min_CP_duration: This parameter is communicated at channel set-up by the ECMG to the SCS to indicate the minimum supported amount of time a CW 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 CPs for the particular stream. It means that all the CPs 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 CPs shall have a duration greater than or equal to the nominal_CP_duration. In addition, the nominal CP duration (chosen by SCS) shall be greater than or equal to:

a) all the min_CP_duration specified by the ECMGs during Channel_set-up;

b) all the max_comp_time values specified by the ECMGs during channel set-up, plus typical network latencies.

section_TSpkt_flag: A value of "0" shall indicate that the ECM carried on the interface is in MPEG-2 section format. A value of "1" shall indicate that these datagrams are in MPEG-2 transport stream packet format.

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

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

7.1.3 Channel specific messages

7.1.3.1 Channel_setup message

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 set-up a channel once the TCP connection has been established, as described in subclause 5.1.3.1. 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).

7.1.3.2 Channel_test message

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.

7.1.3.3 Channel_status message

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 5.1.3.1 and 5.1.3.5). All the parameters listed above are mandatory.

When the message is a response to a set-up, the values of the parameters are those requested by the ECMG. All these parameter values will be valid during the whole life time 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.

7.1.3.4 Channel_close message

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.

7.1.3.5 Channel_error message

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 the sent by the ECMG when a channel level error occurs. A table of possible error conditions can be found in subclause 7.1.5.

7.1.4 Stream specific messages

7.1.4.1 Stream_set-up message

Parameter	Number of instances in message
ECM_channel_ID	1
ECM_stream_ID	1
nominal_CP_duration	1

The Stream_setup message (message type = 0x0101) is sent by the SCS to set-up a stream once the channel has been established, as described in subclause 5.1.3.2.

7.1.4.2 Stream_test message

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.

7.1.4.3 Stream_status message

Parameter	Number of instances in message
ECM_channel_ID	1
ECM_stream_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 set-up, the values of the parameters are those 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.

7.1.4.4 Stream_close_request message

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.

7.1.4.5 Stream_close_response message

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.

7.1.4.6 Stream_error message

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 the sent by the ECMG when a stream level error occurs. A table of possible error conditions can be found in subclause 7.1.5.

7.1.4.7 CW_provision message

Parameter	Number of instances in message
ECM_channel_ID	1
ECM_stream_ID	1
CP_number	1
CP_CW_combination	CW_per_msg.
CP_duration	0 to 1
access_criteria	0 to 1

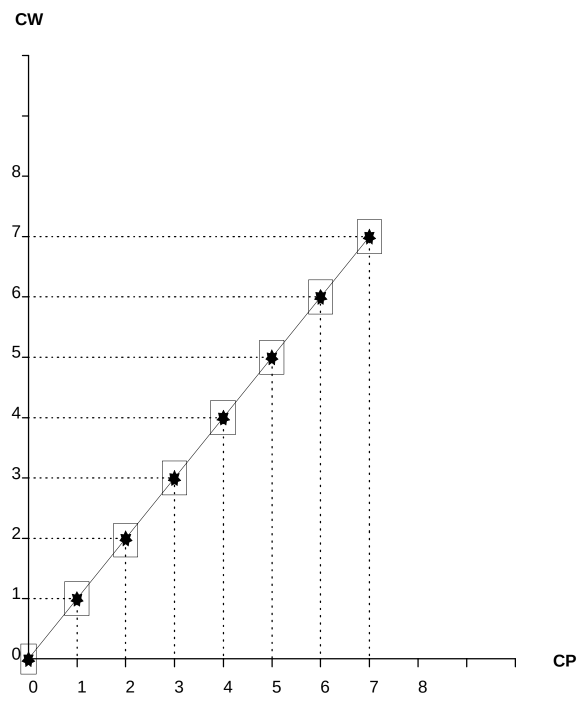
The CW_provision message (message_type 0x201) is sent by the SCS to the ECMG and serves as a request to compute an ECM. The value of the CP_number parameter is the CP number of the requested ECM. The CWs are carried by this message with their associated CP numbers in the CP_CW_combination parameter,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

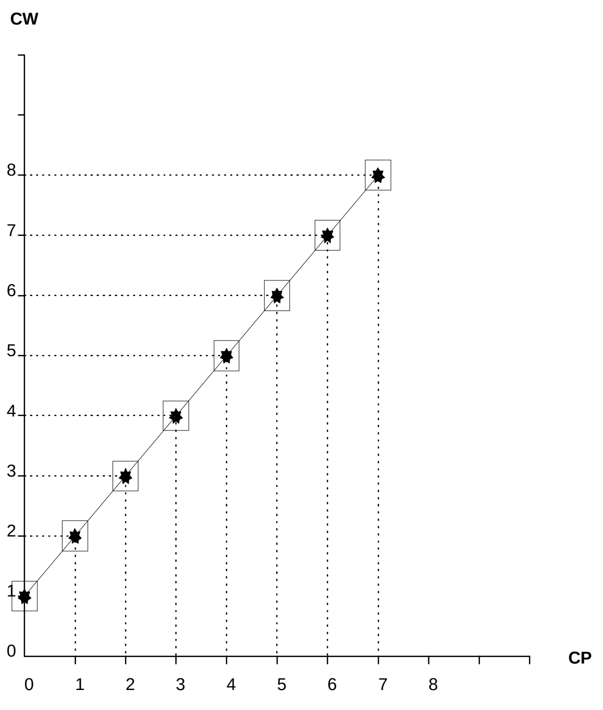
The specific CWs that are passed in the CP_CW_combination to the ECMG via the CW_provision message are derived from the values of lead_CW and CW_per_msg. The following table shows a number of different values these parameters can take to achieve different ECMG requirements.

Example	Requirements	lead_CW	CW_per_msg
1	One (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	Three (3) CWs per ECM per CP	1	3

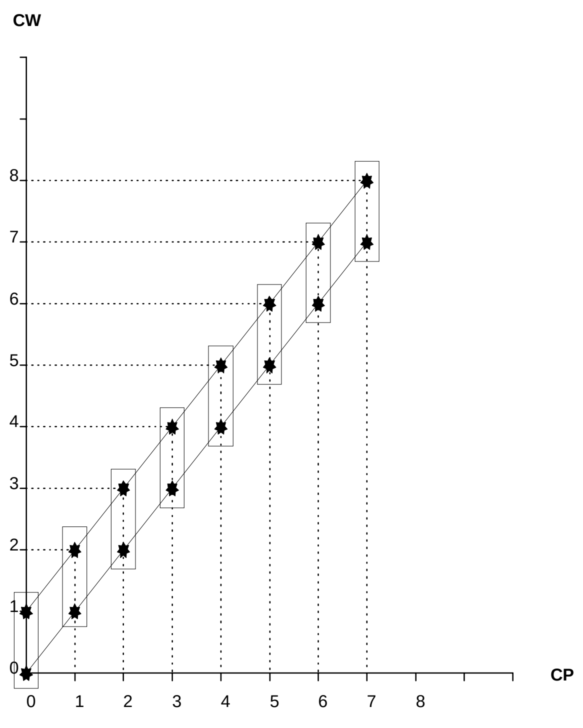
The graphs in figure 2 depict which CWs shall be passed for a specific CP, based on the different methods listed above. For any given CP, X-axis, the corresponding CW is portrayed on the Y-axis. In the CW_provision message, the boxed CWs are the set of CP_CW_combination that shall be passed.



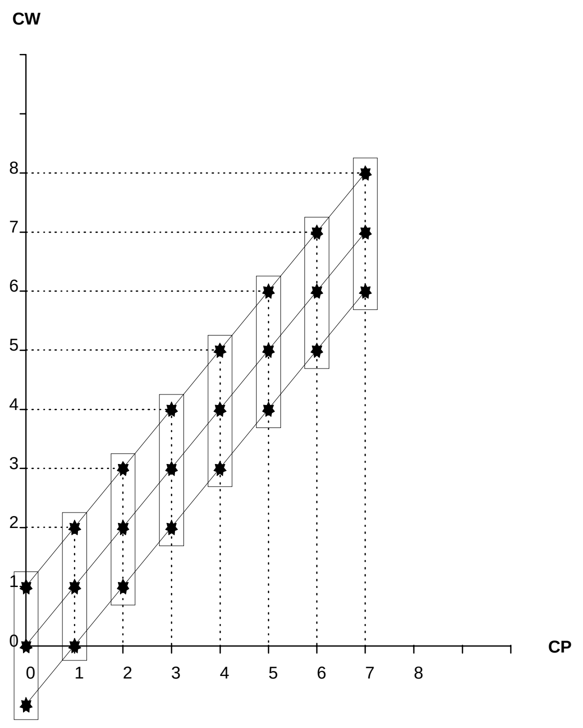
Example 1



Example 2



Example 3



Example 4

Figure 2: Lead_CW to CW_per_msg relationship

7.1.4.8 ECM_response message

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 set-up, and the typical network delays.

7.1.5 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 life time 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 below can not occur in normal operation. They are mainly provided to facilitate the integration and debugging phase.

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 CWs to compute ECM
0x000C	ECMG out of storage capacity
0x000D	ECMG out of computational resources
0x000E	Unknown parameter_type value
0x000D	Inconsistent length for DVB parameter
0x000F	Missing mandatory DVB parameter
0x0010	Invalid value for DVB parameter
0x0011 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

7.2 EMMG <> MUX and PDG <>MUX

7.2.1 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 to 0x6FFF	DVB Reserved	-	-
0x7000	Error_status	see subclause 7.2.5	2
0x7001	Error_information	user defined	variable
0x7002 to 0x7FFF	DVB reserved	-	-
0x8000 to 0xFFFF	User defined	-	-

7.2.2 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 bit rate. The EMMG/PDG will not exceed the negotiated bit 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 2 first bytes of the client_id should be equal to the 2 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 a EMM/Private Data stream within a channel.

Data_channel_ID: This identifier uniquely identifies a EMM/Private Data channel within a client_ID.

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.

Data_type: Type of data carried in the datagram in the stream. The field shall have the value 0x00 for EMM data. Values for other private data types are user defined.

Error_status: See subclause 7.2.5.

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: A value of "0" shall indicate that the EMM or private datagrams carried on the interface are in MPEG-2 section format. A value of "1" shall indicate that these datagrams are in MPEG-2 transport stream packet format.

7.2.3 Channel specific messages

7.2.3.1 Channel_setup message

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 set-up a channel once the TCP connection has been established, as described in subclause 5.2.3.1. It shall contain the client_ID parameter indicating to the MUX the EMMG/PDG that is opening the channel.

7.2.3.2 Channel_test message

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.

7.2.3.3 Channel_status message

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 5.2.3.1 and 5.2.3.6). All the parameters listed above are mandatory.

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

7.2.3.4 Channel_close message

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.

7.2.3.5 Channel_error message

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 the sent by the MUX when a channel level error occurs. A table of possible error conditions can be found in subclause 7.2.5.

7.2.4 Stream specific messages

7.2.4.1 Stream_setup message

Parameter	Number of instances in message
client_ID	1
data_channel_ID	1
data_stream_ID	1
data_type	1

The Stream_setup message (message_type = 0x0111) is sent by the EMMG/PDG to set-up a stream once the channel has been established, as described in subclause 5.2.3.2.

7.2.4.2 Stream_test message

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.

7.2.4.3 Stream_status message

Parameter	Number of instances in message
client_ID	1
data_channel_ID	1
data_stream_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.

7.2.4.4 Stream_close_request message

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.

7.2.4.5 Stream_close_response message

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 EMMG/PDG indicating the stream that is being closed.

7.2.4.6 Stream_error message

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 the sent by the MUX when a stream level error occurs. A table of possible error conditions can be found in subclause 7.2.5.

7.2.4.7 Stream_BW_request message

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.

7.2.4.8 Stream_BW_allocation message

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).

7.2.4.9 Data_provision message

Parameter	Number of instances in message
client_ID	1
data_channel_ID	1
data_stream_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).

7.2.5 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 life time of a TCP connection.

There are two different error messages on those 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 the table below can not occur in normal operation. They are mainly provided to facilitate the integration and debugging phase.

error_status value	Error type
0x0000	Reserved
0x0001	Invalid message
0x0002	Unsupported protocol version
0x0003	Unknown 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
0x000F to 0x7FFE	Reserved
0x7000	Unknown error
0x7001	Unrecoverable error
0x8000 to 0xFFFF	MUX specific/CA system specific/User defined

8 Timing and play-out issues

8.1 Timing issues

In all systems there is a Crypto Period (CP) when data is scrambled with a particular Control Word (CW).

For STBs to regenerate the CW in time, the ECM play-out shall 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 play-out time.

The timing diagram in Figure 3 illustrates this relationship between CW generation, ECM generation, ECM play-out and CP.

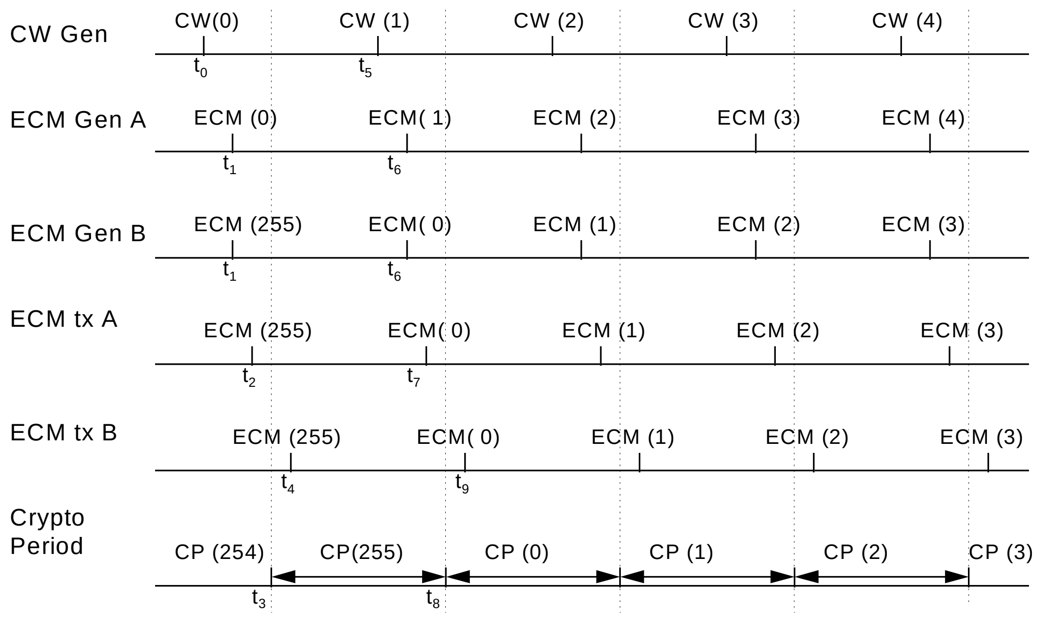


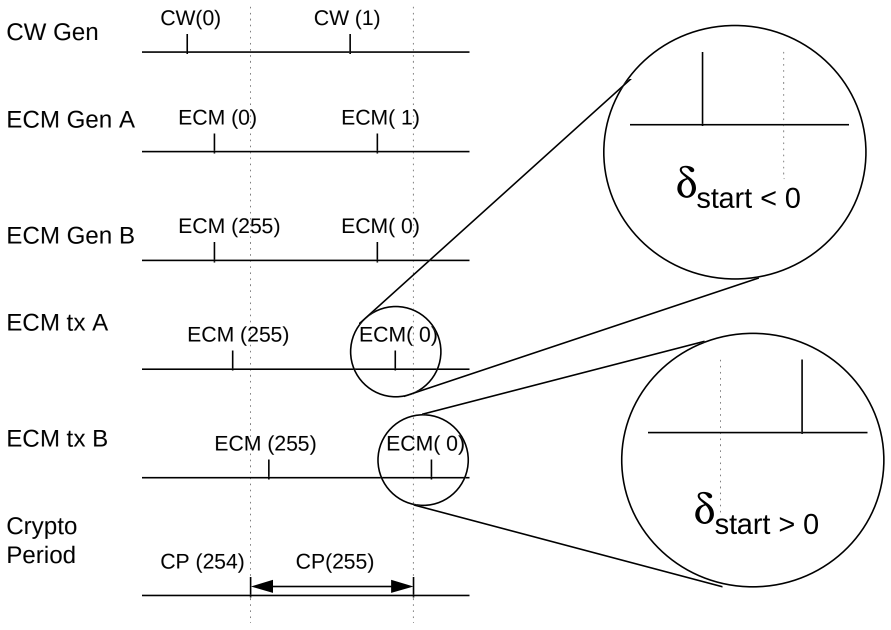
Figure 3: ECM timing diagram

At the beginning of CP(0), at t_8, the scrambler begins using CW(0) 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(0), at t_0, it is able to produce ECM(0), at t_1. This ECM (0) is transmitted sufficiently prior to the beginning of CP(0), to ensure that the STB can obtain CW(0) before CP(0).

CA system B achieves the same result by producing its ECM for two CWs. As soon as ECM Gen. B receives CW(0), at t_0, it is able to produce ECM(255), at t_1. ECM(255), which encompasses CW(255) and CW(0), is transmitted from the middle of CP(255), at t_4. This ensures that the STB can obtain CW(0) before CP(0) begins. After CP(0) begins, at t_9, CW(0) and CW(1) are available in ECM(0).

8.2 Delay start



NOTE: delay_start is equal to delay_stop.

Figure 4: Delay_start

Delay_start: This signed integer represents the amount of time between the start of a CP, 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 CP. 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 set-up.

8.3 Delay stop

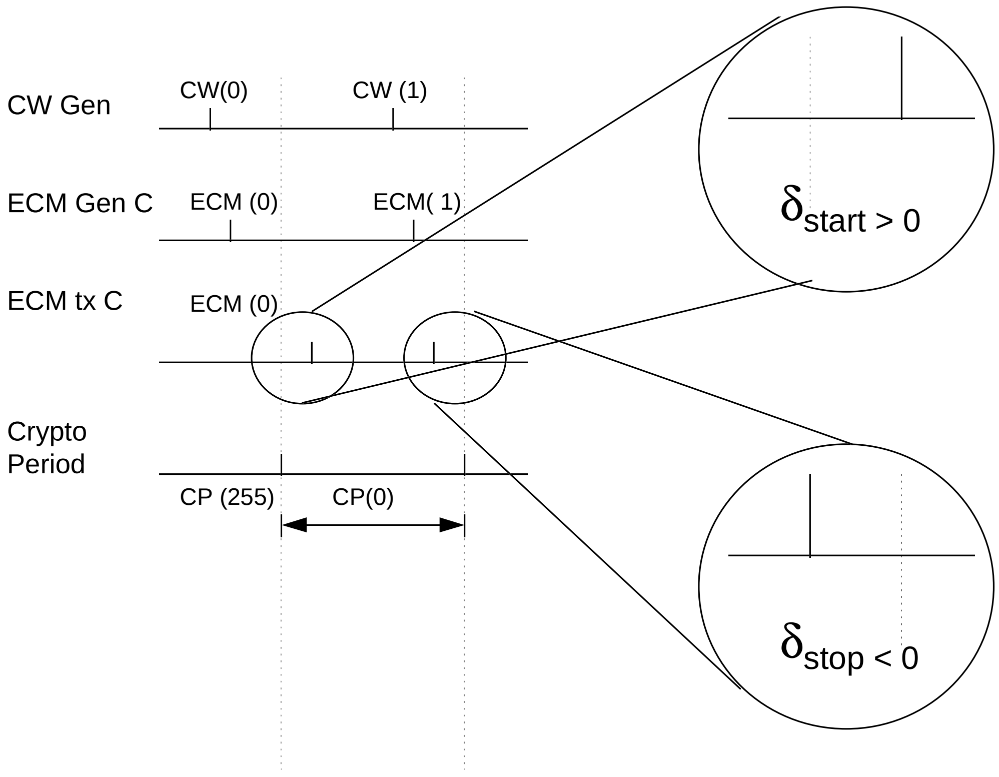


Figure 5: Delay_stop

Delay_stop: This signed integer represents the amount of time between the end of a CP, 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 CP. 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 set-up.

8.4 Play-out issues

8.4.1 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 play-out of this ECM at the time calculated with the delay start parameter;
- the SCS shall stop the play-out of the previous ECM of the same stream at the same time.

In other words the play-out of two ECMs of the same stream can never overlap.

The play-out 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 messdage), the SCS can extend the duration of the current CP (e.g. to attempt to reconnect or switch to a backup device). In such as case the play-out of ECMs is extended accordingly.

8.4.2 EMMs and private data

The MUX should play-out EMMs/Private datagrams in the order in which they arrive.

8.5 Event realignment

If a new event starts in the middle of a CP (either from a program or a change in access criteria), CPs may need to be re-aligned. It is the SCS' responsibility to make sure that during this re-alignment, no CP drops below the nominal_CP_duration.

In other words, if 21:00:00 starts a new event and the nominal CP duration is 20 seconds, and a CP starts at 20:59:30, then the SCS is not allowed to make a 10-second CP between 20:59:50 and 21:00:00.

Instead, if it needs to align CPs with the start of the events, it shall lengthen the previous CP to 30 seconds, so that it ends exactly at 21:00:00.

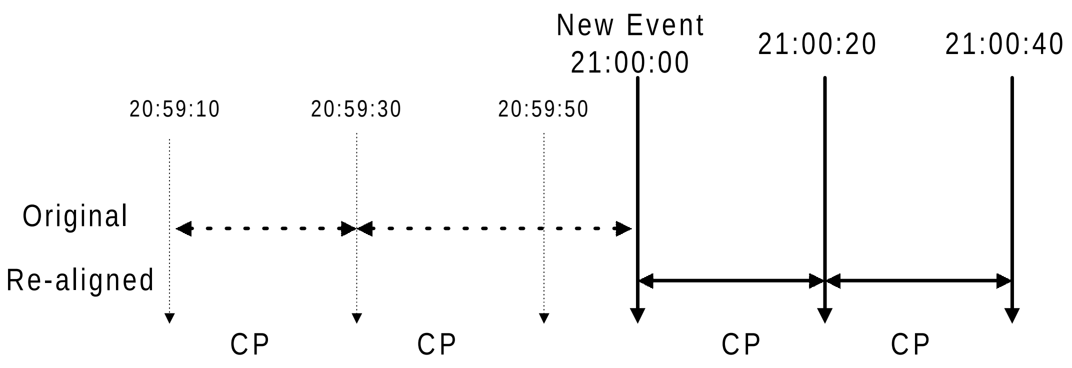


Figure 6: Event Realignment

9 System layering

9.1 Introduction

Each subclause describes a single system layer as defined within the OSI model. The Presentation Layer is not described in the context of the present document.

9.2 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.

9.3 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.

9.4 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). Hosts within IP are uniquely identified by their IP address.

9.5 Transport Layer

The Transport Layer provides the facilities to allow two hosts, either directly or indirectly connected, to exchange data in a reliable and sequenced manner. 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.

TCP uses the "port" concept together with the IP address as provided by the network layer to support the individually addressable end-points of communication. For a description of TCP, refer to RFC 793.

9.6 Session Layer

The data exchange facility as provided by the Session Layer to the Application Layer has the following features:

- **Connection based:** all communication takes place between two uniquely defined communication end-points (no "broadcasts");
- **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.
- **Unformatte**d: 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 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.

9.7 System layering overview/Communications protocol stack

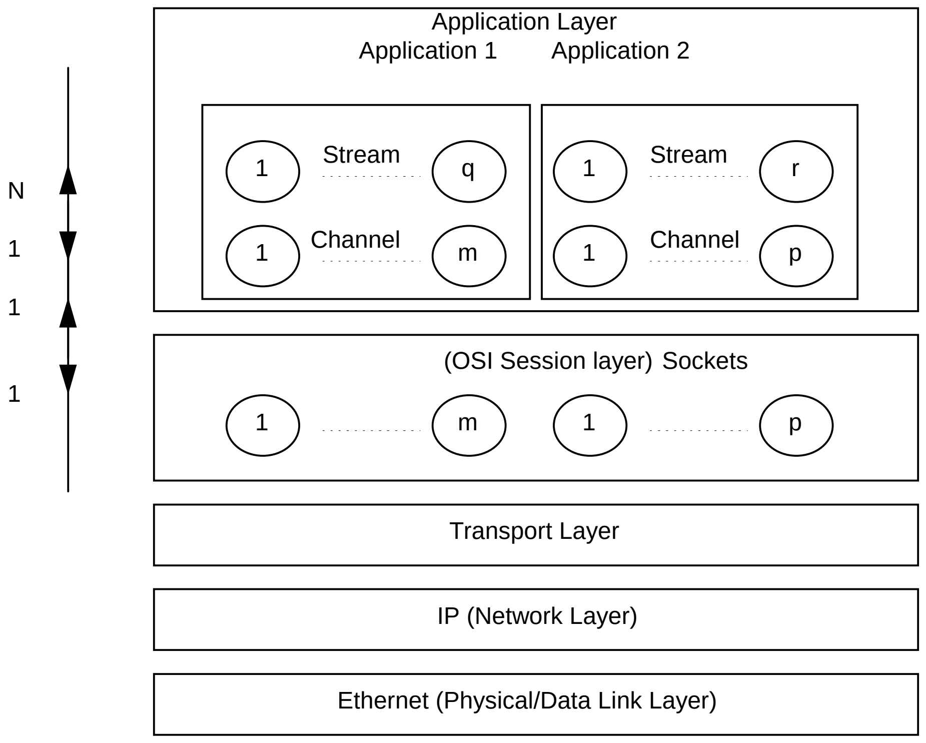


Figure 7: Systems layering overview diagram

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.

9.8 TCP 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 Domain Name Server (DNS). The DNS can be consulted by the client to retrieve the required information.

TCP 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 A (informative): SCS coexistence

A.1 Introduction

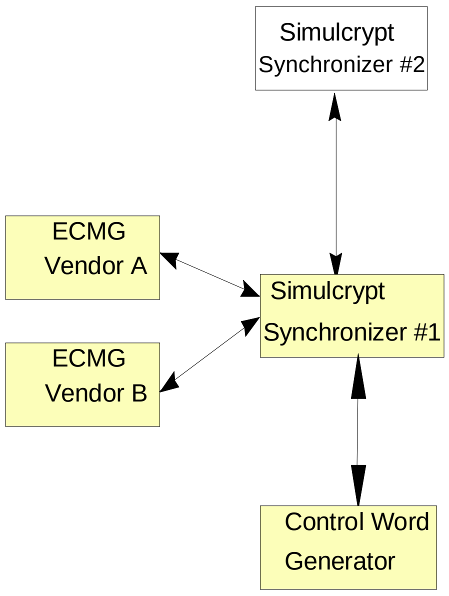
This annex describes how the ECMG ⇔ SCS message interfaces could be used to communicate with another SCS (e.g. in a hot-standby configuration). In a SimulCrypt environment all the information that an SCS needs to communicate with a MUX, is encompassed by the "Channel Status", "Stream Status", "CW Provision" and "ECM Response" messages. This information can be passed to another SCS which can then use this information to maintain its internal status.

A.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. The 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 play-out.

In the example below, 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.



Annex B (informative): Control Word (CW) generation and testing

B.1 Introduction

The Control Word Generator (CWG) is an integral part of the DVB SimulCrypt system which generates low-level cryptographic keys which are used directly to scramble content. It should supply CWs 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 CWs with deterministic properties.

Since generating highly random CWs 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 CWs with high confidence in their randomness qualities.

B.2 Background

The generated CWs should approach as nearly as practicable true random sequences. In general, the criteria for producing cryptographically-secure random sequences include:

1) they shall *appear* random, that is they shall seem to an observer to possess the qualities of true random sequences;

2) 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;

3) 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 CWs, only that great care shall 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.

B.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 to 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 criterion 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 (CWG) should preferably use a physical source such as thermal noise, diode noise, MISC or the equivalent to generate random sequences. Pseudo-random techniques should be used advisedly and only after exhaustive testing to insure at least criteria 1 and 2 above are met.

B.4 Control Word (CW) 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 (see subcaluses B.4.1 and B.4.2).

B.4.1 Bias 1/0

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);
- XOR-ing bits together will exponentially converge to $p(0) = 0{,}5$

A 2-bit example: $p(0) = (0{,}5 + e)^2 + (0{,}5 - e)^2 = 0{,}5 + 2\,e^2$

A 4-bit example: $p(0) = 0{,}5 + 8\,e^4$

Recommendation:

1/0 bias detection tests should be run on the generated sequences, and corrections should be made when needed.

B.4.2 Auto-correlation

Auto-correlation 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 auto-correlation property shall be minimized. There are many algorithms available to measure auto-correlation properties, and most specify the test to be run on small (100-kbit) 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:

Auto-correlation tests should be run on sequences at intervals sufficient to ensure with reasonable certainty that no deterministic properties exist.

B.5 Testing locations

Although it is recommended that the above tests be conducted at the output of the CW generator, CA operators should consider conducting similar tests at the input of the of their ECMGs to confirm the randomness of the CWs they receive.

History

Document history		
V1.1.1	June 1997	Publication

ISBN 2-7437-1453-0
Dépôt légal : Juin 1997