ETSI TS 102 006 V1.2.1 (2002-10)

Technical Specification

Digital Video Broadcasting (DVB); Specification for System Software Update in DVB Systems

Reference
RTS/JTC-DVB-125-2

Keywords
broadcasting, data, 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.org

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

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

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

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

The present document incorporates both the simple and the extended profiles of the former DVB Data Download Specification, TS 102 006-1, therefore TS 102 006-1 V1.1.1 has been withdrawn and replaced by the present document.

Introduction

The present document defines agreements on which to base interoperability for *system software update* services and receivers. These have been selected to minimize interdependencies between the parties involved. In particular:

- It defines the signalling information that can be used to locate the transport stream containing the *system software update service* in a network via the NIT or BAT as appropriate.
- It defines the signalling information used to locate the *system software update service* in a transport stream (via the PMT).
- It defines the options for transmitting the actual *system software update service* in either a proprietary data transfer format, or a standardized 2-layer DVB data carousel (called standard update carousel from here on).

- It defines a Update Notification Table (UNT) that can be used to enhance the system software update functionality in an upward compatible way. The table provides a standard mechanism for carrying additional information, e.g. update scheduling information, extensive selection and targeting information, action notification, filtering descriptors.

- It defines a recommended format for exchanging the *system software update* data from receiver manufacturer to the network (or multiplex) operator for subsequent transmission. In case multiple receiver manufacturers share the same standard update carousel this format allows such a multi-vendor carousel to be composed from individual manufacturers contributions in a simple way.

The present document has to be seen in context with ETR 162 [3] and EN 300 468 [4] because it describes additional descriptors used for *system software update*.

1 Scope

Receiver software is increasingly complex. In order to guarantee the functionality of a receiver as well as increasing its functionality once deployed in the field a software update service is required. The present document specifies a standard mechanism for signalling a software update service and the means to carry the data for such a software update service. It builds on [1], [3] and [4] for signalling and [2] for data carriage.

The present document does not define the mandatory character of this protocol in a specific context, and it does not exclude the use of proprietary mechanisms for doing a software update. This allows a network to support horizontal market model receivers (e.g. MHP receivers). Equally it allows receivers requiring a software update service to be deployed in a network independent way.

2 References

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

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ISO/IEC 13818-6: "Information technology; Generic coding of moving pictures and associated audio information; Part 6: Extensions for DSM-CC" 1998-09-01.

[2] ETSI EN 301 192 (V1.2.1): "Digital Video Broadcasting (DVB); DVB specification for data broadcasting".

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

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

[5] IEEE 802-1990: "IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture" (http://standards.ieee.org/catalog/olis/802-1990.pdf).

[6] ISO/IEC 8859-1: "Information technology; 8-bit single-byte coded graphic character sets; Part 1: Latin alphabet No. 1".

[7] ISO 639-2: "Codes for the representation of names of languages; Part 2: Alpha-3 code".

[8] ETSI TS 101 197-1: "Digital Video Broadcasting (DVB); DVB SimulCrypt; Part 1: Head-end architecture and synchronization".

3 Definitions and Abbreviations

3.1 Definitions

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

(Receiver) manufacturer: organization which assume prime responsibility for updating the software of a receiver once deployed in the field

NOTE: Depending on legal arrangements this can also apply to service providers and other entities.

system software update: update of receiver software transmitted over the DVB systems

3.2 Abbreviations

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

BAT	Bouquet Association Table
DDB	Download Data Block
DII	Download Info Indication
DSI	Download Server Initiate
bslbf	bit string, left bit first
DSM-CC	Digital Storage Media - Command and Control
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
ISO	International Organization for Standardization
LSB	Least Significant Bit
MHP	Multimedia Home Platform
MPEG	Moving Pictures Expert Group
MSB	Most Significant Bit
NIT	Network Information Table
OUI	Organization Unique Identifier
PAT	Program Association Table
PID	Packet IDentifier
PMT	Program Map Table
PSI	Program Specific Information
SI	Service Information
SSU	System Software Update
TS	Transport Stream
uimsbf	unsigned integer most significant bit first
UNT	Update Notification Table

4 Void

5 Profiles and types of system software update services

5.1 Signalling

The present document defines two profiles for software update services with respect to signalling of the service:

- **Simple profile** software update services: these are based on the description in clauses 7 and 8 using signalling in NIT/BAT and PMT, and do not require the Update Notification Table as per clause 9.

- **Update Notification Table enhanced profile** software update services: these are based on the description in clauses 7, 8 and 9. In this case the UNT carries scheduling, targeting or other selection criteria which cannot be carried in NIT/BAT or PMT.

Consequently also two profiles of receivers exist:

- Receivers supporting only the simple profile.

- Receivers supporting the UNT enhanced profile.

In order for compatibility to be guaranteed service operators and receivers need to comply to the following "backward compatibility" rules:

- Service operators shall at least support the simple profile.
- Receivers shall at least support the simple profile.

NOTE: Since the simple profile service is a subset of the UNT enhanced profile service this implies no significant complexity on either service operator or receiver.

5.2 Data carriage

The present document allows two different formats of *system software update* data carriage in the broadcast stream:

1) Proprietary format streams.
2) Standard update carousel (potentially shared between manufacturers)

In case of 1) it is the responsibility of the receiver manufacturers potentially sharing the update service to identify their organization's stream, or the stream can be uniquely identified as being specific to a receiver manufacturer through the data_broadcast_id_descriptor.

In case of 2) the standard carousel contains the identification of the receiver manufacturer.

6 Network (SI) signalling

The linkage descriptor with the linkage type of 0x09 (*system software update* service) conveys the location of the transport stream carrying a *system software update* service within a network or bouquet respectively. This descriptor **shall** be carried in the first loop of the NIT or in the first loop of a specifically identified BAT (called *system software update* BAT from here on).

The *system software update* BAT is identified by the *system software update* bouquet_id 0xFF00, and if the country_availability_descriptor is used, the country code applicable should be 902 (all countries). This allows a receiver to quickly identify it. If the *system software update* BAT is carried in the transport stream of a network it shall be the same as in any other transport stream of that network carrying the system software update BAT.

NOTE: The preferred positioning of this descriptor is in the NIT. On large networks which operate in an partitioned way (typical for satellite) it may be prohibitive to carry this descriptor in the NIT (e.g. due to size constraints of the NIT), in which case carriage in the *system software update* BAT is appropriate.

If OUIs (plus additional selector bytes) are listed in the linkage_descriptor the list of OUIs shall be complete in that it shall convey information about all software upgrades conveyed on the respective service. This allows a receiver to conclusively detect that it may *not* have to further explore a service. A specific OUI with value 0x00015A has been reserved by DVB. This OUI might be used for other purposes despite the System Software Update described in the present document. Within the scope of the present document it is used to signal that the data_broadcast_id_descriptor does not signal any specific OUI. In that case further selection information shall be carried either in the standard data carousel or the Update Notification Table as referenced in the descriptor. If the DVB OUI is used only this single OUI shall be contained in the loop of the data_broadcast_id descriptor. There can be multiple descriptors in the NIT or system software update BAT to allow multiple system software update services to be identified. It is specifically not the intention to remove this descriptor from the NIT or BAT in case of temporary absence of the service. For this purpose specific organization identification (OUI) **shall** not be removed from this descriptor in case of temporary absence of a *system software update* service for receivers of identified organization.

6.1 Linkage Descriptor for Systems Software Update

Table 1: Syntax for the private data bytes for linkage type 0x09

Syntax	No. of bits	Identifier
System_software_update_link_structure(){		
OUI_data_length	8	uimsbf
for (i=0; i<N; i++){		
OUI	24	bslbf
selector_length	8	uimsbf
for (j=0; j<N; j++){ selector_byte	8	uimsbf
}		
}		
for (i=0; i<N; i++){ private_data_byte	8	uimsbf
}		
}		

Semantics of the private data bytes for linkage type 0x09:

OUI_data_length: this field specifies the total length in bytes of the following OUI-loop.

OUI: this is a 24-bit field containing an IEEE OUI (as described in IEEE 802-1990 [5]) of the organization providing a system software update service on the transport-stream/service. DVB has defined OUI 0x00015A to signal that the stream is from any OUI.

selector_length: this 8-bit field specifies the total length in bytes of the following selector field.

selector_byte: this field provides information additional to the OUI that can be used by a receiver to locate and identify the *system software update* service, e.g. model type or ranges. The syntax and semantics of the selector field are defined by the organization owning the OUI.

private_data_byte: this is an 8-bit field, the value of which is privately defined.

6.1.1 SSU Scan Linkage Descriptor

This linkage descriptor defines a pointer to a transport stream carrying a *system software update* BAT or NIT with detailed signaling information about *system software update services*. The linkage type for this descriptor shall be 0x0A and may be inserted into a BAT or NIT.

It is different from a linkage descriptor of type 0x09 in the sense that this descriptor does not contain any OUI specific data. It may be exploited by the receiver to quickly acquire the multiplex carrying the *system software update* BAT or NIT without the need of scanning all multiplexes. The use of the linkage descriptor of type 0x0A is therefore complementary to the use of the linkage descriptor of type 0x09 in the NIT or *system software update* BAT.

The table_type field indicates whether the SSU Scan Linkage Descriptor points to a NIT or BAT on the target transport stream.

The use of this descriptor is optional.

Table 2: Syntax for the Linkage Descriptor of type 0x0A

Syntax	No. of bits	Identifier
linkage_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
linkage_type	8	uimsbf
if (linkage_type =0x0A){		
table_type	8	bslbf
}		
}		

Semantics for the linkage descriptor of type 0x0A:

transport_stream_id: this is a 16-bit field which identifies the TS containing the *system software update* BAT or NIT

original_network_id: this 16-bit field gives the label identifying the network_id of the originating delivery system of *system software update* BAT or NIT indicated.

service_id: this is a 16-bit field which is not relevant, and shall be set to 0x0000.

linkage_type: this is an 8-bit field specifying the type of linkage, and shall be set to 0x0A.

table_type: this is an 8-bit field containing a flag pointing either to the *system software update* BAT or NIT

Table 3: Table_type flag

Value	Description
0x00	not defined
0x01	NIT
0x02	BAT
0x03 – 0xFF	reserved for future use

7 PSI signalling

The PMT of the transport stream carrying *system software update* data **shall** contain the data_broadcast_id descriptor with the data broadcast id of 0x000A to indicate the elementary stream used for the *system software update* service.

The descriptor is considered essential for the location of a *system software update* service in all of the following cases:

- The descriptor provides an entry point to a proprietary stream.
- The descriptor provides the entry point to a standard two-layer data carousel without further reference from a table.
- The descriptor provides the reference to a Update Notification Table.

In these cases this descriptor **shall** be present on a "semi-static" basis; i.e. the identification of the *system software update* service operator shall not be removed from the PMT if there is presently no *system software update* service, but it is expected that there will be in the near future.

The descriptor may contain specific OUIs (plus selector bytes), in which case the list of OUIs (plus selector bytes) shall be complete.

A specific OUI with value 0x00015A has been reserved by DVB. This OUI might be used for other purposes despite the System Software Update described in the present document. Within the scope of the present document it is used to signal that the data_broadcast_id_descriptor does not signal any specific OUI. In that case further selection information shall be carried either in the standard data carousel or the Update Notification Table as referenced in the descriptor. If the DVB OUI is used only this single OUI shall be contained in the loop of the data_broadcast_id descriptor. There can be multiple descriptors in the NIT or *system software update* BAT to allow multiple *system software update* services to be identified. So it is specifically not the intention to remove this descriptor from the NIT or BAT in case of temporary absence of the service. For the same purpose specific organization identification **shall** not be removed from this descriptor in case of temporary absence of a *system software update* service for receivers of identified organization.

Where a separate standard update carousel is used for each OUI (plus applicable selector bytes), the data_broadcast_id_descriptor in the PMT **shall** contain the single OUI (plus selector bytes) for each component. This allows vendor unique identification of proprietary format streams and provides for additional convenience for the receiver in the process to identify the appropriate elementary stream in case there is only one applicable option.

It should be noted that the data_broadcast_id_descriptor for a *system software update* service is defining a single elementary stream. A single program can encompass multiple elementary streams and thus multiple *system software update* streams (carousels), each of which shall be described by its own data_broadcast_id_descriptor. A system software update stream can also be carried as a component of another service, which may simplify network management.

7.1 Data Broadcast Id Descriptor selector byte definition for *System Software Update*

data_broadcast_id: this field shall be set to 0x000A to indicate a *system software update* service (see ETR 162 [3]).

selector_byte: the selector bytes shall convey the *system_software_update_info* structure which is defined as follows.

Table 4: Syntax for the *system_software_update_info* structure

Syntax	No. of bits	Identifier
system_software_update_info(){		
OUI_data_length	8	uimsfb
for (i=0; i<N; i++){		
OUI	24	bslbf
reserved	4	
update_type	4	
reserved	2	
update_versioning_flag	1	
update_version	5	
selector_length	8	uimsbf
for (j=0; j<N; j++){		
selector_byte	8	uimsbf
}		
}		
for (i=0; i<N; i++){		
private_data_byte	8	uimsbf
}		
}		

Semantics of the id_selector bytes for data_broadcast_id 0x000A:

OUI_data_length: this field specifies the total length in bytes of the following OUI-loop.

OUI: this is a 24-bit field containing an IEEE OUI (as described in IEEE 802-1990 [5]) of the organization providing a system software update service on the transport-stream/service. DVB has defined OUI 0x00015A to signal that the stream is from any OUI.

update_type: this is a four-bit field defining the type of the system software update service as indicated in table 5.

Table 5: Update_type table

Value	Description
0x0	proprietary update solution
0x1	standard update carousel (i.e. without notification table) via broadcast
0x2	system software update with notification table (UNT) via broadcast
0x3	system software update using return channel with UNT
0x4-0xF	reserved for future use

update_versioning_flag: if it is 0 no relevant versioning information is carried in the version field. If it is 1 the version field **shall** reflect changes in the system software update service component.

update_version: the version **shall** be incremented on each change of the update. If the update_versioning_flag is set to 1 and the update_type is set to 0x2 or 0x3 (UNT) then the update_version field shall be the same as the version_number in the UNT section header.

selector_length: this 8-bit field specifies the total length in bytes of the following selector field.

selector_byte: this is an 8-bit field. The sequence of selector_byte fields specifies the selector field. This field provides information additional to the OUI that can be used by a receiver to locate and identify the *system software update* service, e.g. model type or ranges. The syntax and semantics of the selector field are defined by the organization identified by the OUI.

8 Standard Data Carousel layout for *System Software Update* services

8.1 Structure of the Standard Update Carousel

The proposed protocol is based on the DSM-CC data carousel specification (ISO/IEC 13818-6 [1]) and the specification of DVB data carousels (EN 301 192 [2]).

Multiple *system software updates* of multiple manufacturers are transmitted as groups in a two-layered Data Carousel. The DownloadServerInitiate message (DSI) is used as the entry point in the carousel and is shared by multiple manufactures. One manufacturer can have multiple updates, each update in a separate group. It is assumed that all groups and modules can be transmitted on a shared elementary stream.

The DownloadServerInitiate message describes the downloads (groups) with the GroupInfoByte (gi) field. The GroupInfoByte field consists also of a loop of descriptors that may contain miscellaneous information. The compatibilityDescriptor of the DSI message is located in the GroupInfoIndication field and allows the identification of the manufacturer (using the IEEE OUI).

Only the DSI is shared by multiple manufacturers, all data in a group will typically belong to one manufacturer. Figure 1 illustrates the proposed protocol. In the figure, manufacturer A has one active update and one non-active (i.e. scheduled/announced) update (empty group). Manufacturer B has one active update.

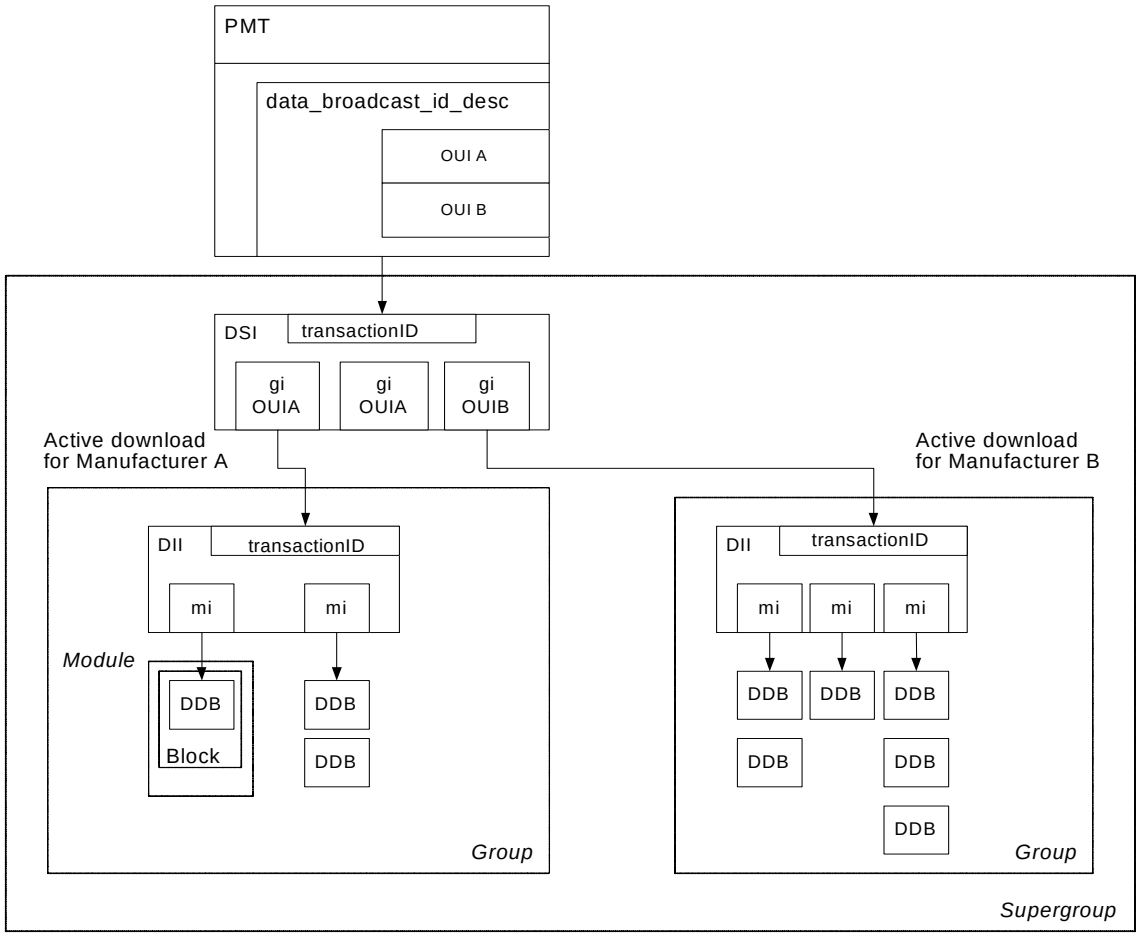

Figure 1: Multiple updates in a two-layered Data Carousel all sharing the same elementary stream

A data_broadcast_id_descriptor in the PMT is used to signal the presence of one or more *system software updates*, of one or more manufacturers.

8.1.1 DownloadServerInitiate message (DSI)

The DownloadServerInitiate message (DSI) is used as the entry point in the carousel and can be shared by multiple manufactures. One manufacturer can have multiple updates, each update in a separate group.

It is assumed that all groups and modules can be transmitted on a single elementary stream.

The DownloadServerInitiate (DSI) message carries the compatibilityDescriptor in the GroupCompatibility field of the GroupInfoIndication structure to allow the identification of the manufacturers group (download) using the IEEE OUI. The GroupInfoByte (gi) field of the GroupInfoIndication structure can consist of a loop of descriptors that contain miscellaneous information for each group. In case UNT information is essential in the interpretation of the carousel, clause 9.6.2.2 indicates how to use the compatibility descriptor to avoid access by receivers uninformed of the UNT.

In order to allow for multiple updates to be generated independent from each other and transmitted on the same carousel, specific assignment rules for some particular fields are defined below (see annex B).

transactionId: the two least significant bytes of a DSI's transactionId shall be in the range 0x0000 to 0x0001. The least significant bit of actual transaction ID changes every time there is a change to the underlying carousel structure (i.e. a group is added, changed or removed) as specified in EN 301 192 [2].

The two most significant bytes (bits 31 to 16) contain a number which identifies the carousel version and can be used to detect version changes.

serverId: this field shall be set to 20 bytes with the value of 0xFF.

compatibilityDescriptor(): this structure shall only contain the compatibilityDescriptorLength field of the CompatibilityDescriptor() as defined in DSM-CC (see ISO/IEC 13818-6 [1]). It shall be set to the value of 0x0000.

The privateDataByte fields shall contain the GroupInfoIndication structure as defined below:

privateDataLength: this field defines the length in bytes of the following GroupInfoIndication structure.

privateDataByte: these fields shall convey the GroupInfoIndication structure as defined in table 6.

Table 6: GroupInfoIndication structure

Syntax	Num. of Bytes	Remarks
GroupInfoIndication() {		
NumberOfGroups	2	number of updates (maximum 150)
for (i = 0; i < NumberOfGroups; i++) {		
GroupId	4	
GroupSize	4	
GroupCompatibility		
GroupInfoLength	2	
for (i=0; i<N; i++) {		
GroupInfoByte	1	
}		
PrivateDataLength	2	
for(i=0;i< privateDataLength;I++) {		
PrivateDataByte	1	
}		
}		
}		

Table 7: CompatibilityDescriptor

Syntax	Num. of Bytes	Remarks
CompatibilityDescriptor() {		
CompatibilityDescriptorLength	2	
DescriptorCount	2	
for (i = 0; i < descriptorCount; i++) {		
descriptorType	1	
descriptorLength	1	
specifierType	1	0x01 (IEEE OUI)
SpecifierData	3	IEEE OUI as described in IEEE 802-1990
model	2	is equal to 0 if the model is transmitted in a manufacturer private location
version	2	is equal to 0 if the version is transmitted in a manufacturer private location
subDescriptorCount	1	
for (j = 0; j < subDescriptorCount; ++		
subDescriptor()		
}		
}		
}		

Table 8: DescriptorType

descriptorType	Description
0x00	Pad descriptor
0x01	System Hardware descriptor.
0x02	System Software descriptor.
0x03- 0x3F	ISO/IEC 13818-6 [1] reserved.
0x40- 0xFF	Private use.

Semantics of the GroupInfoIndication structure:

numberOfGroups: this is a 16-bit field that indicates the number of groups described in the loop following this field.

Applying the procedure described in clause 8.1, with the LSB of the counter position of the groupInfo within the groupInfo loop (the download number) is copied to the MSB of the moduleId, the maximum number of downloads is limited to 255, which is more than sufficient for a MPEG2 transport format

groupId: this is a 32-bit field which shall be equal to the transactionId of the DownloadInfoIndication message that describes the group.

Applying the procedure described in clause 8.1, the id part is the same as the counter position of the groupInfo within the groupInfo loop (the download number). The download number is in the range 1 to NumberOfGroups. The range starts at 1 to meet the requirement that at least one bit in the least significant bits 1 to 15 of the DII transactionId has to be 1.

groupSize: this is a 32-bit field that shall indicate the cumulative size in bytes of all the modules in the group.

groupCompatibility: the GroupCompatibility structure is equal to the CompatibilityDescriptor structure of DSM-CC. The CompatibilityDescriptor should contain a system hardware descriptor containing the OUI that is equal to the OUI present in the system_software_update_info structure of the data_broadcast_id_descriptor in the PMT. If multiple updates of the same manufacturer are present, the model and version fields in the system hardware descriptor and the system software descriptor can be used by the receiver to select the correct stream. Only descriptors of descriptorType System Hardware descriptor and System Software descriptor are used.

groupInfoLength: this is a 16-bit field indicating the length in bytes of the descriptor loop to follow.

groupInfoByte: not defined in the present document.

privateDataLength: this field defines the length in bytes of the following privateDataByte fields.

privateDataByte: these fields are not used.

8.1.2 DownloadInfoIndication message (DII)

The DII message provides information about all the modules that are part of the download scenario. The DII message shall follow the syntax as specified in ISO/IEC 13818-6 [1], table 7-6.

In order to allow for multiple updates to be generated independent from each other and transmitted on the same carousel, specific assignment rules for some particular fields are defined below (see annex B).

transactionId: for DownloadInfoIndication messages the id part of the transactionId shall be in the range 0x0002-0xFFFF to differentiate it from a DownloadServerInitiate messages.

The transactionId is equal to the groupId (group number) in the corresponding groupInfo structure in the DSI.

downloadId: is equal to the transactionId.

Semantics of the moduleInfo structure:

moduleId: field is an identifier for the module that is described here further

Applying the procedure described in clause 8.1:

- Bits 15 to 8: has the same value as the LSB of groupId in the corresponding groupInfo structure in the DSI referencing this particular download.
- Bits 7 to 0: is the moduleId of a particular download, supporting 256 modules.

The maximum number of modules in this case is limited to 256, which can be considered as sufficient for system software update.

moduleVersion: field is the version of the described module.

Applying the procedure described in clause 8.1, this value is also reflected in the LSB of the transaction id in the corresponding groupInfo structure in the DSI referencing this particular download.

8.1.3 DownloadDataBlock message (DDB)

The DDB message is used to convey module payloads. The message syntax shall be as specified in ISO/IEC 13818-6 [1], table 7-7. The DSM-CC section syntax shall follow that defined in ISO/IEC 13818-6 [1], table 9-2 and clause 9.2.2.1.

moduleId: is equal to the moduleId of the module to which this block belongs.

moduleVersion: is equal to the moduleVersion in the DII moduleInfo structure of the module to which this block belongs.

blockNumber: identifies the position of the block within the module. Block number 0 shall be the first block of a module.

8.2 Standard Data Carousel Descriptors

8.2.1 SSU Module Type Descriptor

The SSU_type_descriptor contains the type of the SSU module.

Table 9: Syntax of SSU_type_descriptor

Syntax	No. of bytes	Remarks
Type_descriptor(){		
Descriptor_tag	1	0x0A
Descriptor_length	1	
SSU_module_type	1	
}		

Semantics of the type_descriptor:

descriptor_tag: this 8-bit field identifies the descriptor. For the SSU type descriptor it is set to 0x0A.

descriptor_length: this 8-bit field specifies the number of bytes of the descriptor immediately following this field.

SSU_module_type: this is an 8-bit field, types are describes as following:

Table 10: SSU_module_type

Value:	Module type:
0x00	executable module type
0x01	memory mapped code module type
0x02	data module type
0x03 to 0xFF	reserved for future use

8.3 Time availability guidelines for simple SSU services.

The availability of a download in the simple profile configuration should be no less than two hours. This allows receivers to monitor arrival of a new download with an interval of approximately 1 hour. In specific situations (mutually agreed between operator and receiver maker) other rules may be applicable.

9 Update Notification Table

9.1 Description

The Update Notification Table (UNT) is used with the data_broadcast_id_descriptor (0x000A) where the update_type is set to the value 0x2 or 0x3.

The UNT is broadcast in SI table format; the format is laid out in EN 300 468 [4], clause 5, except that the section length limit is 4 096 bytes.

The UNT is divided into sub-tables indexed by an action_type and an Organization Unique Identifier, administered by the IEEE (IEEE OUI or simply OUI).

NOTE: The OUI selected to form part of the sub-table index is flexible. The OUI may be selected from one of the compatibilityDescriptor OUI's or any other OUI agreed upon. E.g. a vertical network operator offering a standardized IRD to its subscribers, may decide to organize the UNT by its OUI, rather than the contributing manufacturer's OUI, in which case the compatibility descriptors are used to differentiate between devices offered by a specific manufacturer.

9.2 PSI, SI and related UNT signalling

The PMT references the UNT by including the data_broadcast_id_descriptor (data_broadcast_id = 0x000A) in the ES_info loop, where the update_type in the system_software_update_info is set to 0x2 or 0x3. The system_software_update_info OUI, may be either set to the DVB reserved IEEE OUI of 0x00015A to indicate that selection is only possible by analysing the UNT (referenced by this stream in this PMT entry) or the OUI must contain a valid IEEE OUI corresponding to a UNT sub-table index.

Once a candidate UNT has been selected, a search of the table for a sub-table corresponding to the platform's IEEE OUI will proceed. If a sub-table is found, a sequential search through each sub-table section's outermost loop, comparing the compatibilityDescriptor (see clause 9.4.2 or ISO/IEC 13818-6 [1], clause 6) is performed.

For each compatibilityDescriptor match, the target descriptors (if any) must also be compared. The target descriptor loop targets a specific platform device via one or more of the target descriptors defined in the present document. In the case of sequential parsing of the sub-table sections, a match on the compatibility descriptor and the appropriate target descriptors ends the search, and no further searching need be performed. This allows e.g. a beta-release software update to be selected by targeted receivers in favour of a regular software release which may be available at the same time for a larger group of receivers by ordering the UNT entry for the targeted download before that of the regular download. An empty target descriptor loop defines an untargeted SSU (this SSU applies to all platforms identified by the compatibilityDescriptor, unless previously specifically targeted). A target descriptor loop containing unrecognized descriptors also denotes a targeted download, and as such, does not target a device not recognizing the descriptors unless otherwise specifically targeted by other descriptors.

A successful search will yield, depending upon the action(s) to be performed, a reference to the appropriate data carousel via the SSU_location_descriptor's association_tag. This association_tag is used in conjunction with the deferred association_tag_descriptor() in the program descriptor loop of the PMT, or the stream_identifier_descriptor() in the ES_info loop of the PMT's service component stream.

If the update is scheduled, but not yet available, a memorization of start time and location can be performed.

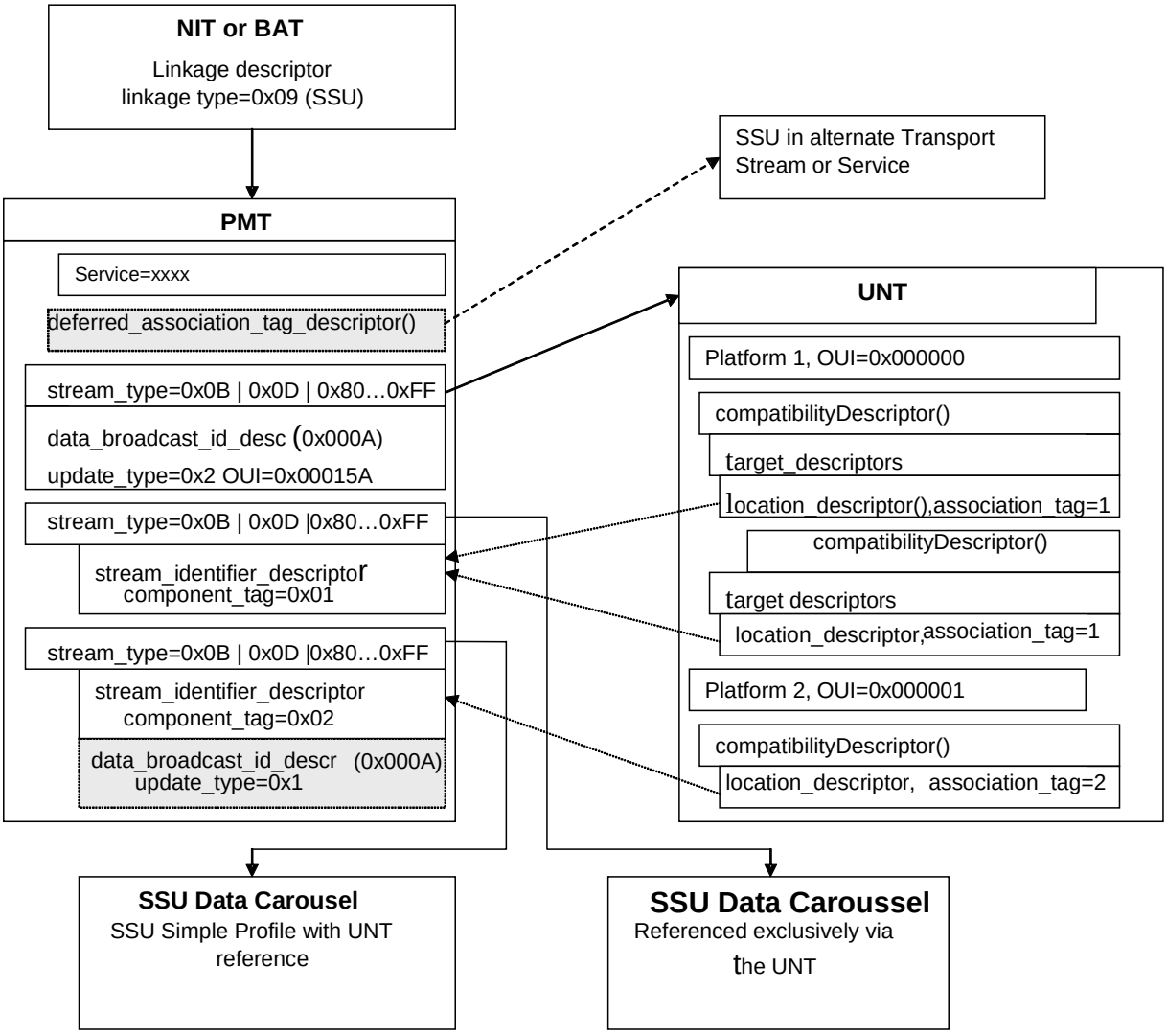

Figure 2: Example for SSU services with UNT reference

NOTE: The DVB generic OUI of 0x00015A shown in the diagram above serves only as a generic example. Other OUIs or lists of OUIs can be used for discrimination.

9.3 Description of the Update Notification Table

The UNT describes the availability and location of SSUs under its jurisdiction. There may be one or many UNT's covering all SSUs for a network. The UNT is referenced by the data_broadcast_id_descriptor (data_broadcast_id = 0x000A), in the ES_info loop of the PMT, where the update_type in the system_software_update_info, is set to 0x2 or 0x3 (indicating a UNT reference). The system_software_update_info OUI may be either set to the DVB reserved IEEE OUI of 0x00015A to indicate that selection is only possible by analysing this UNT or the OUI must contain a valid IEEE OUI corresponding with a UNT's sub-table index OUI.

To assist receiver devices with limited section filter capabilities in locating an appropriate UNT sub-table, the UNT's sub-table OUI is hashed using a simple XOR function and included as part of the table_id_extension.

The definition of the UNT has been limited to SSUs, however during the specification process the need for other notification types was foreseen. In an effort to facilitate these, yet to be defined, notification types, an action_type field forms part of the sub-table index. In addition the processing_order field is provided to assist in selecting the most appropriate action_type.

The UNT is divided into sub-tables using standard DVB table syntax. As such, there may be one or more sections forming the sub-table. A sub-table contains a grouping of SSUs available under the sub-table's OUI and action_type. Note that all sub-tables with the same OUI but possibly different action_type values form a logical group since they together convey all information pertaining to devices covered by the respective OUI holder.

A sub-table section is further divided into 5 hierarchical loops, the 1st loop, the common_descriptor_loop() contains a list of descriptors which, unless overridden in the operational_descriptor_loop(), apply to all SSUs in this sub-table section.

The 2nd loop, the N1 loop, provides a mechanism by which many receiver devices may be addressed by a single sub-table section. The 1st element of the N1 loop is the compatibilityDescriptor(). This is not really a descriptor in the DVB sense, it is, in fact, a structure, the name compatibilityDescriptor has been adopted for historical reasons. Each entry of the N1 loop is identified by the compatibilityDescriptor(), all further elements of this loop relate to this compatibilityDescriptor(), the loop is encoded to allow the remaining elements of the N1 loop iteration to be quickly skipped using the platform_loop_length, which immediately follows the compatibilityDescriptor().

The N2 or platform loop associates an operational_descriptor_loop() with target_descriptor_loop(), allowing multiple targeted or untargeted SSUs to be associated with a platform.

The target_descriptor_loop() contains zero or more descriptors which are used exclusively for targeting. If the loop contains at least one descriptor, the receiver device must be explicitly targeted by at least one descriptor, otherwise this SSU must not be considered.

The final loop, the operational descriptor loop mostly contains descriptors relating to the update processes. Descriptors in this loop normally, but not always, override equivalent descriptors in the common_descriptor_loop(). This loop may be empty, implying no additional descriptors are necessary (to those specified in the common_descriptor_loop()).

9.4 Semantics of the UNT

Syntax

Table 11: Syntax of the update_notification_section

Name	Size (bits)	Unit	Default value
Update_Notification_Table() {			
table_id	8	uimsbf	0x4B
section_syntax_indicator	1	bslbf	1b
Reserved_for_future_use	1	bslbf	1b
Reserved	2	bslbf	11b
section_length	12	uimsbf	Max value=0xFFD
action_type	8	uimsbf	0x01
OUI_hash	8	uimsbf	
Reserved	2	bslbf	11b
version_number	5	uimsbf	
current_next_indicator	1	bslbf	1b
section_number	8	uimsbf	
last_section_number	8	uimsbf	
OUI	24	uimsbf	
processing_order	8	uimsbf	
common_descriptor_loop()			
for (I=0, I<N1, I++) {			
compatibilityDescriptor()			
platform_loop_length	16	uimsbf	
for (i=0, i<N2, I++) {			
target_descriptor_loop()			
operational_descriptor_loop()			
}			
}			
}			
CRC_32	32	rpchof	
}			

9.4.1 Fields description

table_id: uniquely defined for Update Notification Table (0x4B)

action_type: identifies the action to be performed. Coded according to table 12.

Table 12: action_type coding

action_type	Action Specification
0x00	reserved
0x01	System Software Update
0x02 – 0x7F	reserved for future use
0x80 – 0xFF	user defined

OUI_hash: the OUI_hash is formed by XORing all three bytes of the OUI together to form a single byte value (OUI_hash = OUI[23..16]^OUI[15..8]^OUI[7..0]).

section_number: this 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id, action_type and OUI (OUI_hash).

OUI: this field is the IEEE OUI selected to form the sub-table index.

processing_order: indicates the sequence in which to perform actions. If the software download requires more than one action this field can be used to indicate the order to perform these actions. Coded according to table 13.

Table 13: processing_order coding

processing_order	Specification
0x00	first action
0x01 – 0xFE	subsequent actions (ascending)
0xFF	no ordering implied

common_descriptor_loop(): this descriptor loop is intended for descriptors which apply to all platform/target devices listed in this section of the sub-table, unless overridden by descriptors in the operational loop. The common_descriptor_loop is not intended to support platform identification or device targeting descriptors. See table 19 for a list of possible descriptors.

Table 14: Syntax of the common_descriptor_loop()

Name	Size (bits)	Unit/Type
common_descriptor_loop () {		
reserved	4	bslbf
common_descriptor_loop_length	12	uimsbf
for (I = 0; i < N1; i++)		
descriptor()		
}		

NOTE: The application of descriptors in the operational_descriptor_loop() take precedence over (or override) this descriptor loop. See clause 9.4.5 for further details.

9.4.2 compatibilityDescriptor

The compatibilityDescriptor provides the mechanism for discrimination between target platforms. This descriptor may contain hardware and software compatibilityDescriptors or any other compatibilityDescriptor permitted in the SSU data carousel groupInfo. If an SSU data carousel is referenced, the compatibilityDescriptor content must be equal to the equivalent compatibilityDescriptor in the data carousel.

Multiple occurrences of either descriptor in sequence shall have logical OR semantics, and (a sequence of) system hardware descriptor(s) (i.e. H1, H2, H3) followed by (a sequence of) system software descriptor(s) (i.e. S1, S2) shall have logical AND semantics. So the compatibility in the example is defined as (H1 OR H2 OR H3) AND (S1 OR S2).

NOTE 1: The compatibilityDescriptor is not intended to discriminate between individual target devices.

Syntax

NOTE 2: Refer to ISO/IEC 13818-6 [1].

Table 15: Syntax of the compatibilityDescriptor() structure

Syntax	Number of Bytes	Remarks
compatibilityDescriptor() {		
compatibilityDescriptorLength	2	
descriptorCount	2	
for (i = 0; I < descriptorCount; i++) {		
descriptorType	1	
descriptorLength	1	
specifierType	1	0x01 (IEEE OUI)
specifierData	3	IEEE OUI as described in IEEE 802-1990
model	2	is equal to 0 if the model is transmitted in a manufacturer private location
version	2	is equal to 0 if the version is transmitted in a manufacturer private location
subDescriptorCount	1	
for (j = 0; j < subDescriptorCount; j++)		
subDescriptor()		
}		
}		
}		

descriptorType

Table 16: descriptorType coding

descriptorType	Description
0x00	Pad descriptor
0x01	System Hardware descriptor.
0x02	System Software descriptor.
0x03 to 0x3F	ISO/IEC 13818-6 [1] reserved.
0x40 to 0x7F	DVB reserved for future use
0x80 to 0xFF	User defined

NOTE 3: The compatibilityDescriptor is a descriptor loop and not a descriptor in the SI sense with a tag and length.

A receiver device not recognizing a descriptorType must assume this compatibility descriptor is not compatible with this receiver device.

9.4.3 platform_loop_length

Length of the N2 loop (target_descriptor_loop() + operational_descriptor_loop()).

9.4.4 target_descriptor_loop()

The target_descriptor_loop discriminates between individual devices. This descriptor loop may contain target MAC address, smartcard or private, etc, descriptors. This descriptor loop forms a list of all target devices to be addressed and the operation loop applied. If this descriptor loop is empty, the operation loop applies to all devices for this platform (as identified by the compatibilityDescriptor()). When targeting is in use, i.e. target descriptors are present in the target descriptor loop, a receiver device not specifically targeted must not perform an update.

A receiver device not recognizing a target descriptor (new or unknown target descriptor) must assume this target descriptor does not target this receiver device.

Table 17: Syntax of the target_descriptor_loop()

Name	Size (bits)	Unit/Type
target_descriptor_loop () {		
reserved	4	bslbf
target_descriptor_loop_length	12	uimsbf
for (i = 0; I < N1; i++)		
target_descriptor()		
}		

NOTE: When the target descriptor loop is in use, the compatibilityDescriptor assigned in the groupInfo structure of the SSU data carousel must be replaced with the standard DVB compatibility descriptor OUI 0x00015A, to avoid untargeted updates. See also clause 9.6.

9.4.5 operational_descriptor_loop()

The operational_descriptor_loop contains action, informational, and operational descriptors, which apply only to those target devices that meet the requirements of the compatibilityDescriptor and the target descriptor loop. Descriptors in this loop take precedence over (or override) descriptors in the common_descriptor_loop unless, specifically stated otherwise in the descriptor's functional description.

Table 18: Syntax of the operational_descriptor_loop()

Name	Size (bits)	Unit/Type
operational_descriptor_loop () {		
reserved	4	bslbf
operational_descriptor_loop_length	12	uimsbf
for (i = 0; I < N1; i++)		
operational_descriptor()		
}		

9.5 SSU UNT descriptors

9.5.1 Descriptor identification and location

Table 19: SSU UNT descriptors

Descriptor	Tag Value	Allowed in Loop		
		Common	Target	Operational
reserved	0x00			
scheduling_descriptor	0x01	*		*
update_descriptor	0x02	*		*
ssu_location_descriptor	0x03	*		*
message_descriptor	0x04	*		*
ssu_event_name_descriptor	0x05	*		*
target_smartcard_descriptor	0x06		*	
target_MAC_address_ descriptor	0x07		*	
target_serial_number_descriptor	0x08		*	
target_IP_address_descriptor	0x09		*	
target_IPv6_address_descriptor	0x0A		*	
ssu_subgroup_association_descriptor	0x0B			*
reserved for future SSU use	0x0C – 0x3F			
in the scope of DVB SI	0x40 – 0x7F			
telephone_descriptor	0x57	*		*
private_data_specifier_descriptor	0x5F	*	*	*
user private	0x80 – 0xFE			
reserved	0xFF			

Descriptors from the DVB SI range (0x40 to 0x7F) shall have their standard semantics as defined in EN 300 468 [4]. Equally MPEG descriptors in the range 0x00 to 0x3F can not be used in the UNT.

9.5.2 Descriptor coding

9.5.2.1 target_smartcard_descriptor

Location: **Table:** UNT

Loop of table: target

This descriptor targets a specific platform receiver device based upon its smartcard identifier. The smart card identifier is conveyed in the private data bytes.

Table 20: Syntax of the target_smartcard_descriptor

Name	Size (bits)	Unit/Type	Default value
target_smartcard_descriptor () {			
descriptor_tag	8	uimsbf	0x06
descriptor_length	8	uimsbf	
super_CA_system_id	32	uimsbf	
for (I=0;I<N;i++) {			
private_data_byte	8	uimsbf	
}			
}			

super_CA_system_id: DVB CA identifier as per TS 101 197-1 [8] (Simulcrypt).

smart cards numbers are conveyed in the private_data field.

9.5.2.2 target_MAC_address_descriptor

Location: **Table**: UNT

Loop of table: target

This descriptor is used to target a single, or a group of receiver devices for a specific platform, the MAC_addr_mask field is used to define which bits of the MAC address are targeted for comparison, a one bit indicates that this bit shall be compared against the bit in the equivalent position of the MAC_addr_match field. Repeated MAC_addr_match fields are applied as a logical OR (a list of addressed receivers).

Table 21: Syntax of the target_MAC_address_descriptor

Name	Size (bits)	Unit/Type	Default value
target_MAC_address_descriptor () {			
descriptor_tag	8	uimsbf	0x07
descriptor_length	8	uimsbf	
MAC_addr_mask	48	uimsbf	
for (i=0;I<N;i++) {			
MAC_addr_match	48	uimsbf	
}			
}			

9.5.2.3 target_IP_address_descriptor

Location: **Table:** UNT

Loop of table: target

This descriptor is used to target a single, or a group of receiver devices for a specific platform, the IP_addr_mask field is used to define which bits of the IP address are targeted for comparison, a one bit indicates that this bit shall be compared against the bit in the equivalent position of the IP_addr_match field. Repeated IP_addr_match fields are applied as a logical OR (a list of addressed receivers).

Table 22: Syntax of the target_IP_address_descriptor

Name	Size (bits)	Unit/Type	Default value
target_IP_address_descriptor () {			
descriptor_tag	8	uimsbf	0x09
descriptor_length	8	uimsbf	
IP_addr_mask	32	uimsbf	
for (I=0;i<N;I++) {			
IP_addr_match	32	uimsbf	
}			
}			

9.5.2.4 target_IPv6_address_descriptor

Location: **Table:** UNT

Loop of table: target

This descriptor is used to target a single, or a group of receiver devices for a specific platform, the IPv6_addr_mask field is used to define which bits of the IPv6 address are targeted for comparison, a one bit indicates that this bit shall be compared against the bit in the equivalent position of the IPv6_addr_match field. Repeated IPv6_addr_match fields are applied as a logical OR (a list of addressed receivers).

Table 23: Syntax of the target_IPv6_address_descriptor

Name	Size (bits)	Unit/Type	Default value
target_IPv6_address_descriptor () {			
descriptor_tag	8	uimsbf	0x0A
descriptor_length	8	uimsbf	
IPv6_addr_mask	128	uimsbf	
for (I=0;i<N;I++) {			
IPv6_addr_match	128	uimsbf	
}			
}			

9.5.2.5 target_serial_number_descriptor

Location: **Table:** UNT

Loop of table: target

This descriptor targets an action for a specific platform receiver device.

Table 24: Syntax of the serial_number_descriptor

Name	Size (bits)	Unit/Type	Default value
target_serial_number_descriptor () {			
descriptor_tag	8	uimsbf	0x08
descriptor_length	8	uimsbf	
for (I=0;i<N;I++) {			
serial_data_byte	8	uimsbf	
}			
}			

serial_data_byte: this information is intended to target devices based on some manufacturing id. No further definition on the semantics of this information is implied.

9.5.2.6 update_descriptor

Location: **Table:** UNT

Loop of table: common, operational

This descriptor guides the SSU process.

Table 25: Syntax of the update_descriptor

Name	Size (bits)	Unit/Type	Default value
update_descriptor () {			
descriptor_tag	8	uimsbf	0x02
descriptor_length	8	uimsbf	
update_flag	2	bslbf	see table 26 below
update_method	4	bslbf	see table 27 below
update_priority	2	bslbf	11b
for (i=0;I<N;I++) {			
private_data_byte	8	uimsbf	
}			
}			

NOTE: The update_flag and update_method fields in the context of the present document are advisory.

update_flag: This field indicates whether an update should be performed automatically.

Table 26: update_flag coding

Value	Comment
00b	The update has to be activated manually.
01b	The update may be performed automatically.
10b	reserved for future use
11b	reserved for future use

update_method: This field advises the receiver behaviour for the update process.

Table 27: update_method coding

Value	Comment
0	immediate update: performed whatever the IRD state.
1	IRD available: the update is available in the stream; it will be taken into account when it does not interfere with the normal user operation.
2	next restart: the update is available in the stream; it will be taken into account at the next IRD restart.
3…7	reserved for future use
8..14	private use
15	reserved

Example uses of the update_method and update_flag is given in table 28.

Table 28: update_method and update_flag usage example

update_flag update_method	0: manual	1: automatic
0: immediate update	A message asking for the update of the IRD shall be displayed and the IRD shall wait for the user agreement.	The update shall be performed whatever the state of the IRD (force update).
1: IRD available	A message shall inform the user about the availability of an update only if it does not disturb the user (front panel etc…) but current display should not be disturbed by a message.	The update shall be performed only if the IRD is available and the update will not disturb the user.
2: next restart	At the next restart, a message shall ask the user for his agreement to perform the IRD update.	The update will automatically be performed at the next restart.

update_priority: 4 values, meaning the priority associated to the SSU, with 0 being the highest priority, and 3 being the lowest priority.

9.5.2.7 SSU_location_descriptor

Location: **Table:** UNT

Loop of table: common, operational

This descriptor locates the SSU data carousel.

Table 29: Syntax of the SSU_location_descriptor

Name	Size (bits)	Unit/Type	Default value
SSU_location_descriptor () {			
descriptor_tag	8	uimsbf	0x03
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	
if (data broadcast id == 0x000A) {			
association_tag	16	uimsbf	
}			
for (i=0;I<N;I++) {			
private_data_byte	8	uimsbf	
}			
}			

data_broadcast_id: specifies the transport mechanism as defined in ETR 162 [3]. If the value 0x000A is used, this refers to a standard SSU two layer data carousel.

association_tag: this 16-bit field gives the association between the respective update and the streams of the data service which may be done by either using the deferred_association_tag descriptor defined in ISO/IEC 13818-6 [1] or the stream_identifier_descriptor in EN 300 468 [4]. In the latter case, it is assumed that the component_tag field of the stream_identifier descriptor is the least significant byte of the referenced association_tag value.

9.5.2.8 SSU_subgroup_association_descriptor

Location: **Table:** UNT

Loop of table: operational

This descriptor associates an update group within the SSU carousel with the given targeting criteria. This is intended to complement the information in the SSU_location_descriptor.

Table 30: Syntax of the SSU_subgroup_association_descriptor

Name	Size (bits)	Unit/Type	Default value
SSU_subgroup_association_descriptor () {			
descriptor_tag	8	uimsbf	0x0B
descriptor_length	8	uimsbf	5
subgroup_tag	40	uimsbf	
}			

subgroup_tag: the least significant 16 bits of this field shall contain the same value as the subgroup_association_descriptor in the GroupInfoBytes in the GroupInfoIndication structure of the DSI message. This is a unique value under the defining authority of the holder of the OUI conveyed in the field's most significant 24 bits. Note that no relationship between this OUI and any other OUI in the system is implied.

9.5.2.9 scheduling_descriptor

Location: **Table:** UNT

Loop of table: common, operational

This descriptor defines scheduling information. It can appear more than once in the same descriptor loop. If it does it indicates that the update will be available during all the indicated times. Note that schedules listed in the operational descriptor loop replace (i.e. have precedence over) schedules in the common descriptor loop (not additive).

Table 31: Syntax of the scheduling descriptor

Name	Size (bits)	Unit/Type	Default value
scheduling_descriptor () {			
descriptor_tag	8	uimsbf	0x01
descriptor_length	8	uimsbf	
start_date_time	40	uimsbf	
end_date_time	40	uimsbf	
final_availability	1	bslbf	
Periodicity_flag	1	bslbf	0: not periodic. 1: periodic.
period_unit	2	bslbf	see table 32
duration_unit	2	bslbf	see table 32
estimated_cycle_time_unit	2	bslbf	see table 32
Period	8	uimsbf	
Duration	8	uimsbf	
estimated_cycle_time	8	uimsbf	
for (I=0;i<P;I++) {			
private_data_byte	8	uimsbf	
}			
}			

start_date_time and end_date_time: these 40 bit fields indicate the scheduled start and end date and time of the SSU campaign, the actual availability of an SSU may be further refined by the periodicity and related fields. The date and times are encoded in the same format as specified by DVB in the UTC_time of the TDT and TOT (EN 300 468 [4], clause 5.2).

final_availability: this informative field indicates the final schedule for this update. If set to 1 it can be expected that after completion of this schedule the update will be unavailable for the foreseeable future. If set to the default value zero, there *may* be future schedules conveying this update.

periodicity_flag: the SSU may only be available periodically during an update campaign. Setting this 1 bit field to 1 indicates that this schedule corresponds to an SSU which is available periodically between the start and end date and times specified.

period_unit, duration_unit and estimated_cycle_time_unit indicate the units to be used when interpreting the period, duration and estimated_cycle_time fields. The table 32 defines the unit multiplier to be used.

Table 32: Time units coding

Value	Unit multiplier
00b	Second
01b	Minute
10b	Hour
11b	Day

period: this 8 bit field defines the repetition period of the SSU availability expressed in units as defined in the field period_units. The first period always starts at the scheduled start_date_time and repeats periodically until the scheduled end_date_time.

duration: this 8 bit field defines the duration the SSU is available at the start of each period. The duration is expressed in the units defined in duration_units.

estimated_cycle_time: estimated time in estimated_cycle_time_units units of the repetition time of the data required for doing the software update. A value of zero means this field has no meaning.

9.5.2.10 telephone_descriptor (Informative)

Location: **Table:** UNT

Loop of table: common, operational

NOTE: Refer to EN 300 468 [4].

The telephone_descriptor may be used to indicate a telephone number which may be used in conjunction with a modem (PSTN or cable) to use narrow band interactive channels. The actual session/data transfer protocol is not defined by this descriptor: i.e. it is defined privately.

Table 33: Syntax of the telephone_descriptor

Name	Size (bits)	Unit/Type	Default value
telephone_descriptor() {			
descriptor_tag	8	uimsbf	0x57
descriptor_length	8	uimsbf	
reserved_future_use	2	bslbf	
foreign_availability	1	bslbf	
connection_type	5	uimsbf	
reserved_future_use	1	bslbf	
country_prefix_length	2	uimsbf	
international_area_code_char	3	uimsbf	
operator_code_length	2	uimsbf	
reserved_future_use	1	bslbf	
national_area_code_length	3	uimsbf	
core_number_length	4	uimsbf	
for (i=0;i<N;i++){			
Country_prefix_char	8	uimsbf	
}			
for (i=0;i<N;i++){			
international_area_code_char	8	uimsbf	
}			
for (i=0;i<N;i++){			
operator_code_char	8	uimsbf	
}			
for (i=0;i<N;i++){			
national_area_code_char	8	uimsbf	
}			
for (i=0;i<N;i++){			
core_number_char	8	uimsbf	
}			
}			

Semantics for the telephone_descriptor:

foreign_availability: this is a 1-bit flag. When set to "1" it indicates that the number described can be called from outside of the country specified by the country_prefix. When set to "0" it indicates that the number can only be called from inside the country specified by the country_prefix.

connection_type: this is a 5-bit field which indicates connection types. One example of the use of the connection type is to inform the IRD that when, if an interaction is initiated, if the connection is not made within 1 minute, then the connection attempt should be aborted.

country_prefix_length: this 2-bit field specifies the number of 8-bit alphanumeric characters in the country prefix.

international_area_code_length: this 3-bit field specifies the number of 8-bit alphanumeric characters in the international area code.

operator_code_length: this 2-bit field specifies the number of 8-bit alphanumeric characters in the operator code.

national_area_code_length: this 3-bit field specifies the number of 8-bit alphanumeric characters in the national area code.

core_number_length: this 4-bit field specifies the number of 8-bit alphanumeric characters in the core number.

country_prefix_char: this 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [6] gives one alphanumeric character of the country prefix.

international_area_code_char: this 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [6] gives one alphanumeric character of the international area code.

operator_code_char: this 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [6] gives one alphanumeric character of the operator code.

national_area_code_char: this 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [6] gives one alphanumeric character of the national area code.

core_number_char: this 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [6] gives one alphanumeric character of the core number.

9.5.2.11 SSU_event_name_descriptor

Location: **Table:** UNT

Loop of table: common, operational

This descriptor may be used to inform the user of this system software update (SSU) as an event. It is similar in structure and meaning to the short_event_descriptor.

Table 34: Syntax of the SSU_event_name_descriptor

Name	Size (bits)	Unit/Type	Default value
SSU_event_name_descriptor () {			
descriptor_tag	8	uimsbf	0x05
descriptor_length	8	uimsbf	
ISO_639_language_code	24	bslbf	
name_length	8	uimsbf	
for (i=0; i<N; i++) {			
name_char	8	uimsbf	
}			
text_length	8	uimsbf	
for (i=0; i<N; i++) {			
text_char	8	uimsbf	
}			
}			

ISO_639_language_code: this 24-bit field contains the ISO 639-2 [7] three character language code of the language of the following text field. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is encoded into 8 bits according to ISO/IEC 8859-1 [6] and inserted in order into the 24-bit field.

EXAMPLE: French has 3-character code "fre", which is coded as: '0110 0110 0111 0010 0110 0101'.

name_length: this 8 bit field contains the length of the following name string.

name_char: this string of characters provides the SSU event name.

text_length: this 8 bit field indicate the length of the following text_char field.

text_char: this is an 8-bit field. A string of "text_char" fields specifies the text message as described above. Text information is coded using the character sets and methods described in EN 300 468 [4], annex A.

9.5.2.12 message_descriptor

Location: **Table:** UNT

Loop of table: common, operational

This descriptor carries a message in multiple languages. The message may be used to inform the user about the purpose of this System Software Update (SSU) in order to receive the users consent to perform the actual update.

Any occurrence of this descriptor in the operational loop overrides any descriptor of this type in the common loop (even when languages mismatch).

Table 35: Syntax of the message_descriptor

Name	Size (bits)	Unit/Type	Default value
message_descriptor () {			
descriptor_tag	8	uimsbf	0x04
descriptor_length	8	uimsbf	
descriptor_number	4	uimsbf	
last_descriptor_number	4	uimsbf	
ISO_639_language_code	24	bslbf	
for (i=0; i<N; i++) {			
text_char	8	uimsbf	
}			
}			

descriptor_number: this 4-bit field gives the number of the descriptor. It is used to associate information with the same lSO_639_language_code, which cannot be fitted into a single descriptor. The descriptor_number of the first message_descriptor of an associated set of message_descriptors shall be "0x00". The descriptor_number shall be incremented by 1 with each additional extended message_descriptor with the same ISO_639_language_code in the same loop.

last_descriptor_number: this 4-bit field specifies the number of the last message_descriptor (that is, the descriptor with the highest value of the descriptor_number) of the associated set of descriptors of which this descriptor is part.

ISO_639_language_code: this 24-bit field contains the ISO 639-2 [7] three character language code of the language of the following text field. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is encoded into 8 bits according to ISO/IEC 8859-1 [6] and inserted in order into the 24-bit field.

EXAMPLE: French has 3-character code "fre", which is coded as: '0110 0110 0111 0010 0110 0101'.

text_char: this is an 8-bit field. A string of "text_char" fields specifies the text message as described above. Text information is coded using the character sets and methods described in EN 300 468 [4], annex A.

9.5.2.13 private_data_specifier_descriptor (Informative)

Location: **Table:** UNT

Loop of table: all

NOTE: Refer to EN 300 468 [4].

This descriptor is used to identify the specifier of any private descriptors or private fields within descriptors.

Table 36: Syntax of the private_data_specifier_descriptor

Name	Size (bits)	Unit/Type	Default value
private_data_specifier_descriptor () {			
descriptor_tag	8	uimsbf	0x5F
descriptor_length	8	uimsbf	
private_data_specifier	32	uimsbf	
}			

9.6 SSU Data Carousel descriptors

This clause describes additional descriptors for use with the SSU Enhanced Profile in the messages of the SSU Data Carousel.

9.6.1 Descriptor identification and location

For information on identification and location of SSU descriptors in the SSU Data Carousel, please refer to EN 301 192 [2].

9.6.2 Descriptor coding

9.6.2.1 subgroup_association_descriptor

Location: DSI, GroupInfoBytes

This descriptor associates an update group within the SSU carousel with targeting criteria in the UNT.

Table 37: Syntax of the subgroup_association_descriptor

Name	Size (bits)	Unit/Type	Default value
subgroup_association_descriptor () {			
descriptor_tag	8	uimsbf	0x0A
descriptor_length	8	uimsbf	5
subgroup_tag	40	uimsbf	
}			

subgroup_tag: the least significant 16 bits of this field shall contain the same value as the SSU_subgroup_association_descriptor in the UNT. This is a unique value under the defining authority of the holder of the OUI conveyed in the field's most significant 24 bits. Note that no relationship between this OUI and any other OUI in the system is implied.

9.6.2.2 Compatibility descriptor

When the UNT is essential to the download, the hardware compatibility descriptor used in the groupInfo structure of the SSU data carousel must be replaced. The replacement hardware compatibility descriptor's OUI shall be set to the DVB SSU reserved value of 0x00015a, the model and version fields are reserved and shall each contain the value 0xffff. The original hardware compatibility descriptor including any sub-descriptors (if any) shall be copied to the sub-descriptor loop of this replacement hardware compatibility descriptor.

An SSU enhanced profile receiver searching the groupInfo compatibility finding the DVB SSU OUI (0x00015a) will find the original hardware compatibility descriptor in the sub-descriptor loop, but must not act upon any compatibility match without first consulting the appropriate UNT.

At no time must the DVB SSU compatibility descriptor be used in the groupInfo compatibility descriptor loop other than as described above.

9.7 Interworking requirements for operators

In order to allow any receiver to operate in the context of a network claiming to support the present document, the operator shall at least implement the following facilities:

- The operator shall be able to accept the UNT data supplied by the receiver manufacturer with a maximum section size of 4 096 bytes.
- The operator shall be able to modify all necessary fields of the UNT data supplied by the manufacturer so as to compose a valid (potential multi receiver manufacturer) UNT.
- The operator shall support the standard SSU Data Carousel as a format for carrying the actual SSU service. In addition other formats may be supported.
- The operator shall be able to support the scheduling_descriptor. The time an actual software update is pre-announced by a schedule should allow receivers to be able to take notice (and if necessary communicate with the user) under typical use conditions with high reliability. A suggested pre-announcement time could be one week. Note that a schedule may be composed by multiple schedule_descriptors. Note that the operator's support of the scheduling descriptor may be implemented by allowing the receiver manufacturer to supply the scheduling_descriptor as part of the UNT data (obviously after agreement with the operator on the actual schedule).

- Minimal UNT repetition rates are defined as:
  - 10 s on cable and satellite networks.
  - 60 s on terrestrial networks.
- Repetition Rates for DSI and DII Messages: The DSI and each DII shall be repeated at least every 5 s.

9.8 Interworking requirements for receivers

In order to allow a receiver to operate in the network of an operator supporting the present document, the receiver shall at least implement the following:

- The SSU simple profile as defined in clause 5.1 shall be supported.
- All options to locate the appropriate UNT through NIT/BAT and PMT scanning.
- All mandatory requirements in the present document applicable to scanning/interpreting the UNT.
- Receivers should check the action_type and match to the actions they support. They should ignore actions_types not supported.
- Receivers shall support processing_order value 0xFF. Support for other processing order values is optional.
- Support for clause 9.4.2.
- The receiver shall be able to analyse the following descriptors and react appropriately in accordance with the specification:
  - scheduling_descriptor.
  - SSU_location_descriptor.
  - SSU_subgroup_association_descriptor.
  - private_data_specifier_descriptor.
- Processing of descriptors in the target loop shall be in accordance with clause 9.4.4.

Any table sections and carousel groups in the scope of an OUI mismatching the receivers OUI shall be ignored by the receiver. So IRDs shall be robust against any (perceived) non-compliance of such transmitted data not intended for their use.

Annex A (informative): Locating the appropriate System Software Update service

The NIT/BAT, PMT, and the standard update carousel or Update Notification Table define a hierarchy of selection mechanisms so as to allow a receiver to locate and identify the appropriate system software update service.

First a receiver should attempt to locate the appropriate transport stream in a network by examining the NIT and (if available) the system software update BAT. The NIT or BAT may contain OUI unspecific linkage descriptors or multiple instances of descriptors identifying the receivers OUI. Further exploration of each of the remaining candidate services will be required.

The PMT of each system software update service may contain specific OUIs, in which case the receiver can conclude if the service is relevant, i.e. system software update data for it is currently and/or will in the near future be broadcasted. It is possible that the relevant signalling in the PMT did not identify any specific OUI(s) relating to a system software update service and refers to a standard update carousel. If this is the case then the group compatibility descriptor contained in the DSI **shall** contain OUIs for all organizations for which this system software update service is relevant. It may be the case that at this point in time system software update data for some (or even all) OUIs listed is not actually available. However, the rule is still valid as it allows a receiver to determine if this system software update service is relevant. This allows a receiver to constrain scanning for available system software updates to a single service after initial scan.

NOTE 1: In any case there may be multiple locations for the receivers system software update if the selector bytes do not provide additional identification for the receiver to distinguish between multiple system software update services addressing its OUI. Also the location of a system software update service for a receiver may move or may be cancelled. The receiver should be robust against this.

NOTE 2: Only in case a standard update carusel is identified with a single OUI a receiver may be able to identify its system software update service component as exclusive (i.e. no sharing with other manufacturers).

Annex B (informative): Recommendations for transferring *System Software Update* service data from receiver manufacturer to network operator

In order to allow for multiple downloads to be transmitted on the same elementary stream, specific assignment needs to be applied for particular fields, proposed in the following clauses. Main purpose is to avoid overlap between Groups (=downloads) without any need for pre-agreement, i.e. one-layer carousels can be created independently and assembled into a two-layer carousel at some later point. Just to be clear, this is additional profiling proposed for DVB System Software Update and not interpretation of the existing DVB Data Carousel specification. When the transport format in use is MPEG2, the DSI is encapsulated in a single MPEG2 section. Thus the number of Groups is limited by the size of this and allows approximately 150 Groups to be described depending upon the level of detail for the description of each Group.

Using 8 bits of the moduleId field (MSB) to map to 8-bits in the identification sub-field (LSB) of the transactionId would then allow 255 groups, which fits nicely with the ~150 maximum described above. This leaves 8-bits of the moduleId (LSB) for discriminating between Modules related to each Group - as linked by the other 8-bits of the moduleId as described above.

The moduleVersion could also be used to link changes in the download by mapping to 8-bits in the version sub-field of the transactionId (LSB).

The file delivered from the manufacturer to the broadcaster should preferably be in transport stream format (i.e. a sequence of 188 bytes long MPEG-2 TS packets). The manufacturer takes care of generating the file with a full continuity counter run through or the broadcaster has to ensure proper remultiplexing in this respect, which means some further arrangement is necessary between manufacturer and broadcaster. All other fields in the transport packet can be considered as uncritical and can be easily replaced by remultiplexing. Normative repetition rates of the DSI and DIIs are defined in clause 9.7.

In case of a UNT based download service the transfer format suggested for all data from the receiver manufacturer to the operator should be a single Transport Stream formatted file, containing PAT, PMT, UNT(s) and carousels (or proprietary streams). Repetition rates of PMT, PAT and UNT may be adapted by the operator as part of the composition process of the stream that is broadcast, and data may be combined with that of other downloads. In case of a scheduled download service the different (partial) carousels in the transfer file should be carried on different PIDs.

Annex C (normative): Use of the UNT descriptors

This clause describes the semantics for the descriptors used in the UNT.

C.1 compatibilityDescriptor

In each UNT sub-table there shall be at least one compatibilityDescriptor.

C.2 Target loop

The target loop may be empty. If it is empty, all devices under the OUI and compatibilityDescriptor information are addressed.

C.3 Common loop and operational loop

Each update indicated by one compatibilityDescriptor needs to be locatable via location resolution information (LRI). In the case of the SSU, this LRI can be given in:

- SSU_location_descriptor, or
- telephone_descriptor.

To ensure this, there may or may not be LRI in the common loop. In the case that LRI is present in the common loop, it may be overridden by further LRI in the operational loop, possibly under the scope of targeting information. In the case that no LRI is given in the common loop, every iteration instance of the operational loop shall contain unambiguous LRI (i.e. exactly one of the aforementioned descriptors).

Annex D (informative): Bibliography

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

"Implementation guidelines for use of telecommunications interfaces in the Digital Broadcasting systems (DVB Project Office)".

History

Document history		
V1.1.1	December 2001	Publication as TS 102 006-1 (Withdrawn)
V1.2.1	October 2002	Publication