# Final Draft ETSI ES 202 184 V1.1.1 (2004-09)

*ETSI Standard*

# MHEG-5 Broadcast Profile


Reference
DES/JTC-014

Keywords
broadcasting, data, digital, DVB, MHEG, MPEG, terrestrial, TV, 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 existance 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 ETSI Standard (ES) has been produced by the Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen do Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI) and is now submitted for the ETSI standards Membership Approval Procedure phase of the ETSI standards Two-step Approval Procedure.

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of braodcasting 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 organixations 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**

# 0 Introduction

The present document provides a Profile of the International Standard specification ISO/IEC 13522-5 that is adapted for use in enhanced digital television broadcasting. This allows for the broadcast of applications and their presentation in digital TV receivers using an interpreted language (MHEG-5) that is designed explicitly for television use and is robust and easy to use. The Profile also serves to extend some detailed elements of ISO/IEC 13522-5 in a manner that has been found to be useful in practical implementations.

The Profile provides a system for enhanced TV that enables client software in digital TV receivers to be of relatively low complexity. Future versions may include extensions to enable further functionality - in particular return path communications - to be incorporated in a compatible way.

# 1 Scope

The present document describes a complete system Final draft ETSI ES 202 184 V1.1.1 (2004-09) that provides for enhanced interactive TV in the context of a television service that uses the standards set out in the published ETSI specifications for digital TV. Applications for the technology include programme guides, information services, games and enhanced TV services with synchronized interactions and multiple content streams. The Profile identifies the minimum functionality that the receiver will need to support.

## 1.1 Localizing this specification

Unless otherwise stated, the functionality in the present document is mandatory.

In addition, there is functionality which requires additional consideration in a local implementation of the Profile. This information is put into a localized profile of the ETSI-MHEG Profile. The following issues must be addressed when determining what information is required in a localized profile:

- which packages are mandatory or optional;
- allocation of codes;
- handling duplicate services.

### 1.1.1 Packages

The present document also describes a number of "packages". A "package" is a collection of functionality that shall, if provided, be implemented as a whole. In some cases it may be necessary to implement one or more packages in order to satisfy mandatory functionality. In other cases implementation of a package may be optional.

For a particular implementation, a localized profile shall specify exactly which packages are mandatory or optional.

#### 1.1.1.1 Signalling packages

The following signalling packages are defined in the present document:

- AIT based signalling (see clause 9.2).
- PMT based signalling (see clause 9.3).

The Profile requires that one, other or both of these packages be implemented (see clause 9.1.4). A localized profile shall specify that either one must be implemented and the other is optional, or that both must be implemented.

#### 1.1.1.2 Debug package

Debug functionality is provided in a debug package (see clause 11.14). A localized profile shall state whether this package is mandatory or optional.

#### 1.1.1.3 Service Information package

Functionality for retrieving DVB Service Information (SI) is provided by the SI package (see clause 11.15). A localized profile shall state whether this package is mandatory or optional.

### 1.1.2 Allocation of codes

The present document provides for an "application_type_code" (see clause 9.2.10.1, and clause 9.3.2.1.3 ) and a "GetEngineSupport string" (see clause 11.4.1) for determining what level of conformance a receiver has with the Profile (or with a localized profile of this Profile). It may be necessary for a localized profile to state additional codes beyond those already provided by the present document if:

- the localized profile provides extra functionality beyond this Profile;
- receivers exist in a local network that are not yet fully conformant with this Profile and the localized profile of it.

### 1.1.3 Duplicate services

Occasionally a receiver may have to select from 2 or more duplicate services (see clause 16.3.3.3). The present document provides functionality that can be used to handle selecting duplicate services. If a local implementation wishes to use a different mechanism for handling duplicate services this must be specified in the localized profile for that implementation.

# 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, subsequent revisions do apply.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.


| | | |
|---|---|---|
| [1] | ETSI EN 300 468 | ETSI EN 300 468 v1.5.1 (2003-05): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems". |
| [2] | ETSI EN 301 192 | ETSI EN 301 192 v1.2.1: "Digital Video Broadcasting (DVB); DVB specification for data broadcasting". |
| [3] | ETSI TR 101 154 | ETSI TR 101 154 v1.4.1: "Digital broadcasting systems for television; Implementation guidelines for the use of MPEG-2 systems, Video and Audio in satellite, cable and terrestrial broadcasting applications". |
| [4] | ETSI TR 101 211 | ETSI TR 101 211 v1.5.1 (2003-01): "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)". |
| [5] | BS EN 50221 | BS EN 50221: "Common interface specification for conditional access and other digital video broadcasting decoder applications". Also published by CENELEC. |
| [6] | ITU-T X.680 | ITU-T Recommendation X.680: "Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation". Also published as ISO/IEC International Standard 8824-1. |
| [7] | ITU-T X.690 | ITU-T Recommendation X.690: "Information technology – ASN.1 Encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)". Also published as ISO/IEC International Standard 8825-1. |
| [8] | ISO/IEC 8859-1 | ISO/IEC 8859-1:1998: "Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet no.1". |
| [9] | ISO/IEC 13818-1 | ISO/IEC 13818-1:1996(E): "Information technology - Generic coding of moving pictures and associated audio information: Systems". |
| [10] | ISO/IEC 13818-2 | ISO/IEC 13818-2: "Information technology - Generic coding of moving pictures and associated audio information - Part 2: Video". |
| [11] | ISO/IEC 13818-3 | ISO/IEC 13818-3: "Information technology - Generic coding of moving pictures and associated audio information - Part 3: Audio". |
| [12] | ISO/IEC 13818-6 | ISO/IEC 13818-6: "Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for (DSM-CC)". |
| [13] | ISO/IEC 10646 | ISO/IEC 10646: 2000: "Information technology - Universal Multiple-Octet Coded Character Set (UCS): Architecture and basic multilingual plane". |
| [14] | ISO/IEC 13522-5 | ISO/IEC 13522-5: "Information technology - Coding of multimedia and hypermedia information - Part 5: Support for base-level interactive applications". |
| [15] | ETSI TR 101 202 | ETSI TR 101 202 v1.1.1: "Digital Video Broadcasting (DVB); Implementation guidelines for data broadcasting". |
| [16] | DAVIC 1.4.1 Part 09 | DAVIC 1.4.1 Specification Part 09: "Information representation". |

| | | |
|---|---|---|
| [17] | PNG | PNG: "Portable Network Graphics specification - version1.0, 01 October 1996". Available at: http://www.w3.org/TR/REC-png.html |
| [18] | ETSI TS 101 699 | ETSI TS 101 699 v1.1.1: "Digital Video Broadcasting (DVB); Extensions to the common interface specification". |
| [19] | ETSI TR 101 812 | ETSI TR 101 812 v1.3.1: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) specification 1.0.3". |
| [20] | ETSI-MHEG Codes | Allocation of codes for use with the ETSI-MHEG MHEG-5 specification. The latest version can be downloaded from: http://www.dtg.org.uk/dtgtl/mheg |
| [21] | ISO/IEC 13522-5: 1997/Cor.1:1999(E) | ISO/IEC 13522-5:1997/Cor.1:1999(E): "MHEG-5. Information technology - Coding of multimedia and hypermedia information: Support for base-level interactive applications. Technical corrigendum 1". |
| [22] | ITU-R BT.470-4 | ITU-R Recommendation BT.470-4: "Conventional television systems". The version in force is BT.470-6. |
| [23] | ITU-R BT.470-2 | ITU-R Recommendation BT.470-2: "Conventional television systems". The version in force is BT.470-6. |
| [24] | ITU-R BT.601 | ITU-R Recommendation BT.601: " Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios". |
| [25] | RFC 1951 | Request For Comments 1951: "DEFLATE Compressed Data Format Specification version 1.3". This document is available from: http://www.rfc.net/rfc1951.html. |

# 3 Definitions and abbreviations

## 3.1 Definitions

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

**access unit:** a coded representation of a presentation unit

NOTE: In the case of audio, an access unit is the coded representation of an audio frame. In the case of video, an access unit includes all the coded data for a picture, and any stuffing that follows it, up to but not including the start of the next access unit. If a picture is not preceded by a group_start_code or a sequence_header_code, the access unit begins with the picture start code. If a picture is preceded by a group_start_code and/or a sequence_header_code, the access unit begins with the first byte of the first of these start codes. If it is the last picture preceding a sequence_end_code in the bitstream all bytes between the last byte of the coded picture and the sequence_end_code (including the sequence_end_code) belong to the access unit.

**bouquet association table:** Defined in ETSI EN 300 468 [1]

**bouquet_id:** Defined in ETSI EN 300 468 [1]

**broadcaster:** an organization which assembles a sequence of events or programmes to be delivered to the viewer based upon a schedule

**conditional access:** a system to control subscriber access to services, programmes and events, e.g. Videoguard or Eurocrypt

**delivery system:** the physical medium by which one or more multiplexes are transmitted, e.g. satellite transponder, wide-band coaxial cable or fibre optics

**elementary stream:** (see ISO/IEC 13818-1 [9]) a generic term for one of the coded video, coded audio or other coded bit streams in PES packets

NOTE: One elementary stream is carried in a sequence of PES packets with one and only one stream_id.

**event:** a grouping of elementary broadcast data streams with a defined start and end time belonging to a common service, e.g. first half of a football match, news flash or first part of an entertainment show

**event information table:** defined in ETSI EN 300 468 [1]

**MPEG-2:** refers to the standard ISO/IEC 13818: systems coding is defined in part 1 (ISO/IEC 13818-1 [9]), video coding is defined in part 2 (ISO/IEC 13818-2 [10]), audio coding is defined in part 3 (ISO/IEC 13818-3 [11])

**Multimedia and Hypermedia Experts Group:** see ISO/IEC 13522-5 [14]

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

**network:** a collection of MPEG-2 transport stream multiplexes transmitted on a single delivery system, e.g. all digital channels on a specific cable system

**network_id:** defined in ETSI EN 300 468 [1]

**network information table:** defined in ETSI EN 300 468 [1]

**on screen display:** graphical information, locally generated by a piece of equipment, providing information to the user of that equipment

**original_network_id:** unique identifier of a network: defined in ETSI EN 300 468 [1]

**packet identifier:** a unique integer value used to associate elementary streams of a program in a single or multi-program

**packetized elementary stream:** see ISO/IEC 13818-1 [9]

**PES packet:** (see ISO/IEC 13818-1 [9]) the data structure used to carry elementary stream data: it consists of a PES packet header followed by PES packet payload

**PES packet header:** (see ISO/IEC 13818-1 [9]) the leading fields in a PES packet up to and not including the PES_packet_data_byte fields, where the stream is not a padding stream

NOTE: In the case of a padding stream, the PES packet header is similarly defined as the leading fields in a PES packet up to and not including padding_byte fields.

**presentation unit:** (see ISO/IEC 13818-1 [9]) a decoded audio access unit or a decoded picture

**program:** a collection of program elements. Program elements may be elementary streams. Program elements need not have any defined time base; those that do, have a common time base and are intended for synchronized presentation

**program association table:** see ISO/IEC 13818-1 [9]

**program_number:** identifier that identifies an MPEG program

NOTE: An MPEG program is equivalent to a DVB service. The program number is identical to a DVB service id.

**programme:** a concatenation of one or more events under the control of a broadcaster, e.g. news show or entertainment show

**reserved:** when used in the clause defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions

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

**section:** a syntactic structure used for mapping all service information defined in the present document into ISO/IEC 13818-1 [9] transport stream packets

**service:** a sequence of programmes under the control of a broadcaster which can be broadcast as part of a schedule

**service description table:** defined in ETSI EN 300 468 [1]

**service_id:** unique identifier of a service within a transport stream: defined in ETSI EN 300 468 [1]

**service information:** digital data describing the delivery system, content and scheduling/timing of broadcast data streams, etc.

NOTE: It includes MPEG-2 PSI together with independently defined extensions.

**service provider:** see broadcaster

**sub_table:** (from ETSI EN 300 468 [1]) a collection of sections with the same value of table_id and:

- for an NIT: the same table_id_extension (network_id) and version_number;
- for a BAT: the same table_id_extension (bouquet_id) and version_number;
- for an SDT: the same table_id_extension (transport_stream_id), the same original_network_id and version_number;
- for an EIT: the same table_id_extension (service_id), the same transport_stream_id, the same original_network_id and version_number

NOTE: The table_id_extension field is equivalent to the fourth and fifth byte of a section when the section_syntax_indicator is set to a value of "1".

**table:** a table is comprised of a number of sub_tables with the same value of table_id

**table_id:** an identifier of an MPEG table

NOTE: Within a particular context this identifies the data carried by that table.

**table_id_extension:** an identifier typically identifying a subdivision of the data identified by a table's table id

**transport stream:** a data structure defined in ISO/IEC 13818-1 [9]

**transport_stream_id:** unique identifier of a transport stream within an original network: defined in ETSI EN 300 468 [1]

## 3.2 Abbreviations

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

| | |
|---|---|
| BAT | Bouquet Association Table |
| BIOP | Broadcast Inter-Orb Protocol |
| CA | Conditional Access |
| CENELEC | European Committee for Electrotechnical Standardisation: Central Secretariat: Rue de Stassart 35, B - 1050 Brussels |
| CI | Common Interface |
| DSI | Download Server Initiate |
| DTG | Digital Television Group |
| DTT | Digital Terrestrial Television |
| EIT | Event Information Table |
| ES | Elementary Stream |
| EU | European Union |
| HTML | HyperText Markup Language |
| ID | Identifier |
| IOR | Interoperable Object Reference |
| MHEG | Multimedia and Hypermedia Experts Group |
| MPEG | Motion Picture Experts Group |
| NIT | Network Information Table |
| OSD | On Screen Display |
| PAT | Program Association Table |
| PES | Packetized Elementary Stream |
| PFR | Portable Font Resource |
| PID | Packet Identifier |
| PMT | Program Map Table |
| PNG | Portable Network Graphics |
| PSI | Program Specific Information |
| RAM | Random Access Memory |
| SDT | Service Description Table |
| SI | Service Information |
| TDN | The Digital Network |
| TS | Transport Stream |
| UK DTT | Digital Terrestrial Television within the UK |
| UTF-8 | Universal Character Set Transformation Format, 8-bit format |

# 4 Conventions

Void .

# 5 Basic architecture

The MHEG-5 system is designed specifically for delivering interactivity for TV. It is "object based" in its construction. The overall architecture is shown in figure 1. In this diagram, the return channel shown is optional and is not described in the present document.

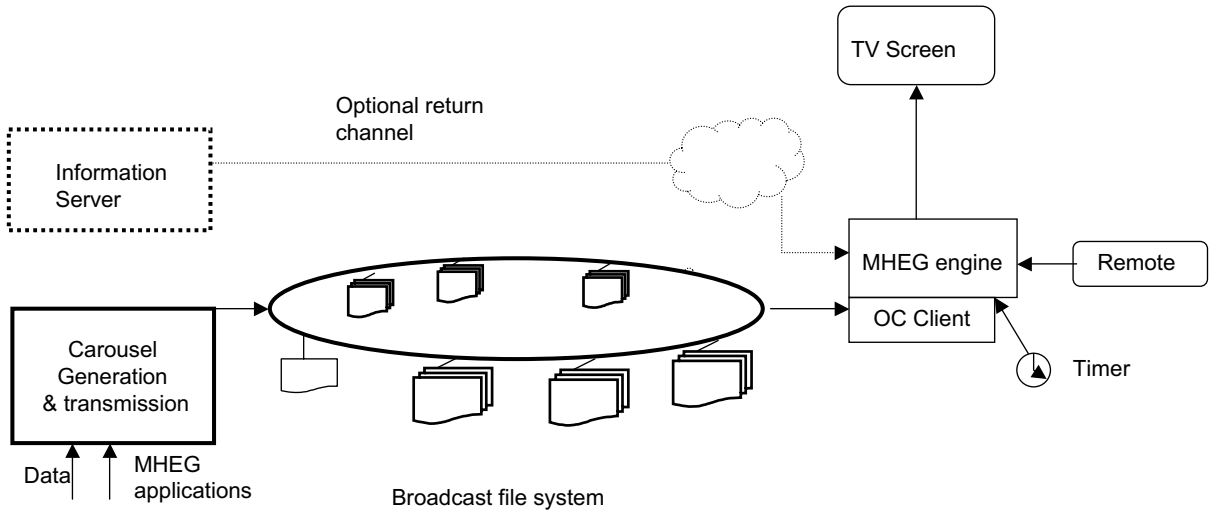



**Figure 1: Basic architecture**

MHEG-5 is conceived in terms of controlled placement of objects on the TV screen and is sometimes referred to as a "presentation engine". In fact it also enables logical constructs and in this Profile has been extended to provide a full solution to the principal needs of the broadcaster of interactive TV. MHEG-5 enables interaction between the user and the interactive TV application through the remote control.

The MHEG-5 programming language comprises objects for presentation, links that respond to events and Resident Programs. Presentation objects include video, audio, lists, text and graphics. Events respond to input from the remote control, a timer, a stream event message or the result of a logical condition specified by the programmer. Resident Programs are native functions, defined in the Profile, that extend the basic MHEG-5 and include sub-stringing and casting which enable the manipulation of text objects as structured data files.

An MHEG-5 application comprises programs of two types; application object and scene object. The application object holds global objects. The scene object(s) holds objects local to the scene. Within an MHEG-5 application there can be navigation between scenes.

MHEG-5 has a simple life-cycle, having only one application running at a time. An MHEG-5 application can launch other MHEG-5 applications but on doing so the original application is destroyed. In a broadcast system, an auto-launch application is started when a service is selected with which it is associated. The auto-launch application can then launch other applications and tune to other services. An application leaving the context of its service is terminated. Information can be passed between applications by making use of persistent store in the receiver.

An application is normally loaded from the DSM-CC object carousel or, optionally, from a return channel or from a detachable module. Data is loaded from the carousel, and this can be updated rapidly on screen as the content of a data object changes. Information is presented either as "included" content, where the text or graphic is embedded in the application, or as "referenced" content, which is acquired from the carousel as required.

In this Profile text is presented with a resident "Tiresias" screenfont, graphics objects are delivered as Portable Network Graphics and MPEG I-Frames or created as lineart. There is a standard 8 bit palette.

The diagram in figure 2 shows the MHEG-5 engine and its interaction with the other parts of the receiver.

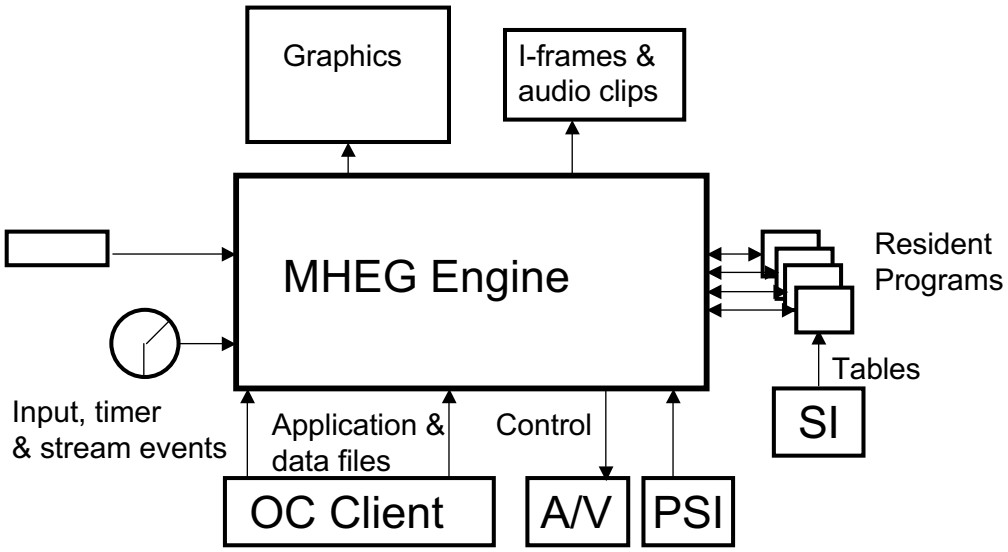



**Figure 2: MHEG-5 engine and its interaction with a receiver**

# 6 Transport protocols

Void .

# 7 Content formats

## 7.1 Static formats

### 7.1.1 Bitmap image formats

#### 7.1.1.1 PNG

See clause 12.7.

#### 7.1.1.2 MPEG-2 I-frames

See clause 12.8.

### 7.1.2 Monomedia formats for audio clips

See clause 11.5.3.

### 7.1.3 Monomedia formats for text

See clause 13.2.

## 7.2 Broadcast streaming formats

See table 18.

## 7.3 Resident fonts

See clause 13.3.2.

## 7.4 Colour representation

See clause 12.3.

# 8 Application model

This clause describes basic application lifecycle concepts, as such can be extracted from the receiver requirements/carousel clauses. It provides the "logical" lifecycle model. Two independent implementations of the model are specified in clause 9.

## 8.1 Application lifecycle

### 8.1.1 Launching and terminating MHEG-5 applications

Applications may be activated in a number of different ways:

- an auto-boot application starts after a broadcast service is selected (see clause 8.1.3);
- an application is introduced by a CI module (see clause 14.9.3.2);
- an application is launched or spawned by an already running application (see clause 8.2);
- an application restarts after an application that it spawned quits (see clause 8.2);
- an auto-boot application starts after all other applications have quit (see clause 8.1.3);
- an auto-boot application starts following a network level stop signal (see clause 9.3.7);
- an auto-boot application starts following a service level stop signal (see clause 9.3.6).

Applications may terminate for a number of different reasons:

- they execute a Quit action;
- they are killed by the receiver following a channel change (see clause 8.1.4);
- they are killed because a CI module generates a RequestStart or AppAbortRequest message (see clause 14.9.3.2);
- they are killed because of a network level stop signal (see clause 9.3.7);
- they are killed following a service level stop signal (see clause 9.3.6).

Two different file systems are potentially available to an application:

- the broadcast file system delivered by the DSM-CC object carousel (see clause 15);
- the file system provided by a CI module.

The broadcast file system, if present, is mounted as a consequence of selecting a broadcast service (see clause 9.3.7.1). The CI file system, if present, is mounted because of activity by a CI module (see clause 14.9). Either or both file systems may be mounted.

### 8.1.2 Preparing for launch

Before implementing Step 1 of the behaviour of the Launch action described in ISO/IEC 13522-5 [14] the receiver should make its best efforts to retrieve the application object to be launched into its memory, i.e. where possible, the engine should not start tearing down the current Scene/Application until a rapid build-up of the next application can be assured.

### 8.1.3 Auto boot broadcast application

Whenever a service that includes an auto-boot MHEG-5 application as a component is selected, or an AIT indicates an auto boot application is present, the MHEG-5 application shall be launched.

If an auto-boot MHEG-5 application component is added to a service (e.g. a new event starts that has an associated MHEG-5 application) the MHEG-5 application shall be launched.

How is this done? The receiver looks continuously for the availability of an MHEG-5 application as part of a service. Therefore:

- when a service is selected, the receiver shall follow the rules in clause 9.3 and clause 9.2 to identify an MHEG-5 application to launch;
- if an MHEG-5 application appears in a service that previously did not have an application, the application will be launched;
- if an MHEG-5 application Quits the receiver shall attempt to launch a new application. See clause 8.2, clause 9.3 and clause 9.2.

### 8.1.4 Auto kill application

**Whenever the receiver changes channel, any executing MHEG-5 application is killed, the application stack is cleared and all file systems are unmounted.**

Therefore if the user interacts with the receivers navigator functions to change channel the current application will be killed. If the new channel has an application this one will be launched. If the application invokes a channel change (via a resident program), and the channel change is successful, then current application will be killed then replaced by any application associated with the new channel.

NOTE: Invocation of different video and audio stream objects by an application is not considered a channel change. Therefore, an application can "preview" a service before selecting it.

### 8.1.5 Application context

The first part of a file reference is the "source" (see clause 16.3.2). This clause defines what this source is at any given time.

#### 8.1.5.1 Initial carousel

The initial carousel, if present, is the auto-mount file system (see clause 9.3.7.1) of the selected service. If there is no auto-mount file system then no initial carousel is defined.

#### 8.1.5.2 Current carousel

At first the Current carousel is the same as the Initial carousel. Application execution can cause the Current carousel to change by launching or spawning applications in a different carousel. This uses "LiteOptionsProfileBody" references between object carousels (see clause 15.2.5.2).

The Current carousel defines where file references with source of "DSM:" are found.

The Current carousel becomes the Initial carousel (if one is defined) in the following cases:

- if the current application quits and the application stack is empty;
- following a RequestStart with an non-empty initial object or AppAbortRequest message from a CI module.

#### 8.1.5.3 Current source

If the currently executing application object is delivered by a broadcast object carousel then the Current source is the Current carousel. So, file references with an empty source are found in the Current carousel. File references with a source of "CI:" can be used to access files in a CI file system if one is available.

If the currently executing application object is provided by a CI file system then the Current source is the CI file system. So, file references with an empty source are found in the CI file system. File references with a source of "DSM:" are found in the Current carousel - if one is exists.

### 8.1.6 Accessible file systems

#### 8.1.6.1 Broadcast applications

All auto-boot broadcast applications, and applications descended from them, have access to files in the Current carousel. To do this references where "source" resolves to "DSM:" are required.

A broadcast application will also have access to files provided by the CI module if a CI module has previously sent a RequestStart message and the session from the module is still open. To do this references where "source" resolves to "CI:" are required.

#### 8.1.6.2 CI introduced applications

All applications introduced by a CI module, and applications descended from them, have access to files provided by the CI module.

Applications from a CI module may also have access to broadcast files if the current service has an auto-mount file system (see clause 9.3.7.1). The broadcast file system available initially will be the Initial carousel of the service (a side effect of a RequestStart that introduces the new initial application from the CI module). However, application action can cause the Current carousel to change.

## 8.2 Application stacking

To support application stacking receivers shall implement an application identifier stack capable of holding references to five applications.

Since the spawned application may be in a different Service Domain (i.e. delivered by a different object carousel) which may be in a different service, each element in the stack shall be able to store not only the spawning application's GroupIdentifier but also information about how to locate the relevant Service Domain.

See:

- clause 11.11.5;
- clause 16.3.2;
- clause 15.2.5.2.

This is in accordance with the ISO/IEC 13522-5 [14]. Note particularly the corrigenda to ISO/IEC 13522-5 [14] with regard to the behaviour of Launch, Spawn and Quit.

If an application terminates following a Quit action and the stack is empty or the application identifier on the top of the stack is not valid, then the receiver shall:

1) Set all stream component decoders to decode the default components for the default service as determined by the receiver and set their presentation (including any video scaling and/or offset, and audio volume control) to the default state.

2) Launch (if present) the autoboot application for the default service following the rules in clause 9.3 or clause 9.2.

The application identifier stack shall be reset in the following circumstances:

- each time there is a service change;
- if a QUIT application control code is received corresponding to the current active application in the AIT;
- if invalid information is found on the stack;
- if an application is introduced by a CI module (using a RequestStart message) displacing an already executing application.

# 9 Signalling

## 9.1 Introduction to application lifecycle signalling

This clause covers the following topics:

- how the receiver identifies an application associated with the current service to use as an auto-boot application,
- the signalling that enables the broadcast to control the lifecycle of the current application.

Applications may be broadcast for a number of purposes and for varying lengths of time. For example, a digital text service may be required to be available 24 hours a day, seven days a week. An enhanced interactive service based on a particular broadcast "program" may only be required to be active for a period of hours or days. At the other end of the scale interactive adverts may only be available for a matter of seconds. As a result it can be seen that a broadcaster needs complete control of how, and when, interactive applications are launched, executed and terminated. This is what is meant by application lifecycle.

Application lifecycle signalling - that is the method by which applications are signalled to launch or terminate - can be categorized to three different types.

### 9.1.1 Application-level signalling

Application level lifecycle signalling is the method by which application authors can control the lifecycle of the application. This could be by authoring the application to quit at a particular time, by using stream events or other application signalling mechanisms. There is no reliance on the network or service configuration and thus it is not considered further in this clause.

### 9.1.2 Service-level signalling

Service level signalling is the way of passing application lifecycle information to the receiver in the broadcast stream that is independent of any currently running applications, i.e. no target code is required in any application. This includes details on which applications are available to launch, and when to stop a currently active application.

### 9.1.3 Network-level signalling

Network level application level signalling is the method by which network operators may control from which component applications are launched and when they are terminated. This enables network operators control over if, when and how applications are launched and terminated, regardless of the content passed to the network via broadcasters and/or carousel generators.

### 9.1.4 Choice of signalling method

The following clauses provide information on two signalling packages: a package based on Application Information Tables (see clause 9.2), and a package using a combination of the service PMT and information in the ServiceGateway object (see clause 9.3). Localized profiles of this Profile shall identify at least one (but potentially both) of these packages that receivers shall use.

Where both packages are to be used platforms should use the following rules to decide which method of signalling to use:

- When the broadcast service includes an AIT component, indicated by the presence of an Application Signalling Descriptor in the PMT, and an AIT sub-table is signalling an ETSI-MHEG application (see clause 9.2.3.2) the platform should use the AIT based signalling package. Otherwise the PMT based signalling package is to be used. The AIT is assumed to include both service and network signalling.
- Whichever package of signalling is chosen for the current service the platform shall continue to use this signalling for the remainder of the service lifetime. The choice of signalling method should be re-evaluated on entry to a new service.

## 9.2 AIT signalling package

### 9.2.1 Introduction

This signalling is conformant with clause 10 of TR 101 812 [19] unless stated otherwise in clause 9.2. The recommendations of clause 10.1.6 of TR 101 812 [19] have been followed.

### 9.2.2 Signalling for transport protocol

The mandatory transport protocol is the Object Carousel protocol as specified in clause 15. Transport via IP and Transport via interaction channel are optional. For Transport via Object Carousel, in either the "common" (first) descriptor loop or the "application" (inner) descriptors loop:

- Transport protocol descriptor, with the selector bytes containing the Object Carousel specific information as defined in table 26 of TR 101 812 [19] with the protocol_id value defined in table 3

### 9.2.3 Application Information Table

#### 9.2.3.1 Visibility of AIT

This clause replaces clause 10.4.4 of TR 101 812 [19].

If an MHEG-5 application tunes away from the current service with or without selecting a new service, it will stop running even if the application is signalled in the AIT of the new selected service. The receiver shall ignore the service_bound_flag (within the Application descriptor).

Note: In broadcasts conformant to the present document the service_bound_flag shall always be set to 1 to enable compliance with MHP receivers.

#### 9.2.3.2 ETSI-MHEG Application type

The value for the application-type field for the ETSI-MHEG type of application is defined in table 1.

**Table 1: ETSI-MHEG Application Type**

| application_type | Description |
|---|---|
| 0x0008 | ETSI-MHEG application |

Note: This value is the application_type defined by MHP and should not be confused with the application_type_code defined in the present document. See 10.4.6 of TR 101 812 [19].

### 9.2.4 Application identification

#### 9.2.4.1 Effects on Life cycle

This clause replaces clause 10.5.2 of TR 101 812 [19].

Only a single instance of a MHEG-5 application with a particular application identifier is allowed to execute at any time. So, if an MHEG-5 application is already running then another instance of the same application shall not be launched.

An MHEG-5 application is killed upon service selection or by Quit action.

### 9.2.5 Control of Application Lifecycle

This clause replaces clause 10.6 of TR 101 812 [19].

The dynamic control of the MHEG-5 application life cycle is signalled through the application_control_code for the application in the AIT. The values and their respective semantics are provided in table 2. If the receiver receives a code that it does not recognize the application shall continue in its current state.

**Table 2: ETSI-MHEG application control code values**

| Code | identifier | Semantics |
|---|---|---|
| 0x00 | | Reserved |
| 0x01 | AUTOSTART | The Application Entry Point of the MHEG-5 application is loaded and started. |
| 0x02 | | reserved for future use |
| 0x03 | QUIT | When the control code changes from AUTOSTART to QUIT the MHEG-5 application is terminated (i.e. a QuitApp EngineEvent shall be sent to the application. |
| 0x04 to 0xFF | | reserved for future use |

## 9.2.6 Starting an application signalled through AIT

When a broadcast service is selected the receiver shall launch, without explicit intervention by the user, the first ETSI-MHEG application signalled in the AIT and identified as AUTOSTART that it is capable of running. This is determined by application_type_code and ISO_639_language_code. (See clause 9.2.10.1.2 and clause 9.2.10.1.4).

The receiver shall monitor the AIT signalling for changes made by the broadcaster. These changes may include the termination of particular applications as well as the addition of new auto-start applications. Once a Carousel has been mounted and an application has been launched no other application shall be started until a QUIT associated with the launched application is received.

For a single "logical" application (identified by a unique application_identifier()), which is carried in the same Object Carousel, there can be 1 or more ETSI-MHEG application location descriptors. Each descriptor represents an "entry point" for the application, based on the following variables:

3) the application_type_code,

4) the boot_priority_hint,

5) the language code.

## 9.2.7 Generic descriptors

A terminal supporting AIT signalling as defined in the present document is only required to support the following descriptors:

- application_signalling_descriptor;
- application_descriptor;
- application_name_descriptor;
- application_icon_descriptor

All other descriptors defined in clause 10.7 of TR 101 812 [19] may be ignored.

#### 9.2.7.1 Application descriptor

The application descriptor is as defined in TR 101 812 [19]. As such there is exactly once instance of the application descriptor in every application descriptor loop. Thus there is a one-to-one mapping of a logical application (application_identifier()) to the Object Carousel carrying it. Note this does not preclude applications with different application_type_codes or language_codes existing in the same Object Carousel (see clause 9.2.10.1). If the same version of an application with different application_type_codes or language_codes are carried in different Object Carousels then they must be signalled as different logical applications.

### 9.2.8 Transport protocol descriptor

The Object Carousel as defined by the present document is compliant with MHP receivers which implement the MHP Object Carousel specification as defined in Annex B of TR 101 812 [19]. As such the value for protocol_id is the same as that defined in MHP (clause 10.8.1 of TR 101 812 [19]). Table 3 specifies the value of this entry.

**Table 3: Protocol_id**

| Protocol_id | Description |
|---|---|
| 0x0001 | Object Carousel protocol as specified in clause 15 |

#### 9.2.8.1 Selector bytes

Selector bytes shall conform to the syntax specified in clause 10.8.1.1 of TR 101 812 [19]. In the present document the remote_connection field in the selector bytes shall be set to 0.

#### 9.2.8.2 Pre-fetch signalling

Pre-fetch signalling is optional but if present shall follow clause 10.8.3 of TR 101 812 [19].

### 9.2.9 MHP application specific descriptors

Clauses 10.9 and 10.10 of TR 101 812 [19] are not required to be supported.

### 9.2.10 ETSI-MHEG specific descriptors

#### 9.2.10.1 ETSI-MHEG application location descriptor

The ETSI_MHEG_application_location_descriptor descriptor is mandatory and shall be used in the application loop of the AIT. It indicates the physical location of the application entry point in the transport media. There is at least one of these descriptors. However, if the carousel contains applications with a different application_type_code or language then there may be more of these descriptors.

The generic form of the ETSI_MHEG_application_location_descriptor is shown in table 4.

**Table 4: ETSI-MHEG application location descriptor**

| | Size (bytes) | Identifier | Value |
|---|---|---|---|
| `ETSI_MHEG_application_location_descriptor {` | | | |
| `descriptor_tag` | 1 | uimsbf | 0x66 |
| `descriptor_length` | 1 | uimsbf | |
| `application_type_code` | 2 | uimsbf | N1 |
| `boot_priority_hint` | 1 | uimsbf | |
| `ISO_639_language_code` | 3 | uimsbf | |
| `for (i=0; i<N; i++) {` | | | |
| `initial_path_byte` | 1 | uimsbf | |
| `}` | | | |
| `}` | | | |

#### 9.2.10.1.1 descriptor_tag

The value 0x66 in this 8-bit field identifies the descriptor.

#### 9.2.10.1.2 application_type_code

The value in this 16-bit field identifies that an autoboot application of this application type is carried by this data broadcast, as indicated by the application_descriptor transport_protocol label. The value is valid within the scope of the ETSI-MHEG application type as defined by the present document.

A receiver compliant with the present documentshall respond to the value for the "ETSI core" application_type_code (see ETSI-MHEG Codes [20]). A localized profile of the Profile may add further values for application_type_code.

#### 9.2.10.1.3 boot_priority_hint

This provides a mechanism for the receiver to decide which application type to boot when it supports more than one of those broadcast and no other rules dictate how to chose between them. The hint is provided by the broadcaster, with 0 the lowest preference. No two ETSI_MHEG_application_location descriptors with the same value of application_type_code and language_code shall have the same boot priority within a single application loop.

#### 9.2.10.1.4 ISO_639_language_code

Specifies the target language of the application.

To specify a default choice or a single, language independent choice, the ISO language code "und" shall be used.

#### 9.2.10.1.5 initial_path_byte

These bytes contain a string specifying the URL path component to the entry point document. This path is relative to the root defined in the physical_root_bytes field.

### 9.2.10.2 ServiceContextList

Whenever the AIT signalling is used the ServiceContextList in the ServiceGateway shall be ignored; the physical root and initial path shall be used to identify the initial object.

## 9.2.11 Service Information

An ETSI-MHEG receiver is not required to support clause 10.12 of TR 101 812 [19].

# 9.3 PMT and ServiceGateway signalling package

## 9.3.1 Introduction

This clause covers the identification and boot of an application by a receiver. The term "application" in this clause of the present document will describe the use of the broadcast stream. Therefore, the application defined in the present document can be described as:

"*ETSI Profile Version 1 of MHEG-5 delivered using the DVB object carousel revision 1*".

The mechanism specified is designed to rely only on information in the PMT and the Service Gateway, i.e. there is no need to access the SDT or EIT to be able to identify and boot the application.

The boot process consists of three steps:

1) Identification of an auto-boot application conforming to this Profile.

2) Acquisition of the ServiceGateway object.

3) Acquisition of the auto-boot object - in this Profile an MHEG-5 Application object. This may be explicitly via signalling in the ServiceContextList of the ServiceGateway object or implicitly by applying the default identification rules.

## 9.3.2 Identification of auto-boot application

Before commencing any boot process the receiver needs to know that there is an auto-boot application of a supported type being broadcast, and from which PID it shall auto-boot. This signalling is achieved by using the data_broadcast_id_descriptor which may be included in a second descriptor loop of a PMT and effectively identifies a "boot-PID", i.e. the elementary stream on which a DSI message can be found.

In this Profile the use of the data_broadcast_id_descriptor for signalling is mandatory when the PMT does not contain an application_signalling_descriptor.

### 9.3.2.1 data_broadcast_id_descriptor

This descriptor is DVB defined and may be included in the second descriptor loop of a PMT. Its exact use and meaning is dependent upon the value of the data_broadcast_id field.

When used with the data_broadcast_id described below, the descriptor lists the application types that can be booted from the elementary stream (component) with which it is associated. Also provided is an indication of the application type that may be booted by receivers that support more than one of the application types listed. Depending on the application type, additional application type specific data may also be provided.

There shall be at most one instance of the descriptor for each elementary stream where the value of the field data_broadcast_id is as specified in table 5. This single descriptor is capable of identifying multiple application types within that elementary stream if necessary.

The generic form of the data_broadcast_id_descriptor when used with the data_broadcast_id registered for this Profile is shown in table 5. Receivers shall be able to parse the descriptor regardless of the value(s) of application_type_code and shall be able to skip the application_specific_data_byte values if they do not recognize the application_type_code.

**Table 5: Data broadcast ID descriptor**

| | Size (bytes) | Value |
|---|---|---|
| data_broadcast_id_descriptor{ | | |
| descriptor_tag | 1 | 0x66 |
| descriptor_length | 1 | N1 |
| data_broadcast_id | 2 | 0x0106 |
| for( i=0; i<N1-2; i++){ | | |
| application_type_code | 2 | |
| boot_priority_hint | 1 | |
| application_specific_data_length | 1 | N2 |
| for( j=0; j<N2; j++ ){ | | |
| application_specific_data_byte | 1 | |
| } | | |
| } | | |
| } | | |

#### 9.3.2.1.1 descriptor_tag

The value 0x66 in this 8-bit field identifies the descriptor.

#### 9.3.2.1.2 data_broadcast_id

The value 0x0106 in this 16-bit field has been registered with DVB for the present document.

#### 9.3.2.1.3 application_type_code

The value in this 16-bit field identifies that an autoboot application of this application type is carried by this data broadcast. The value is valid within the scope of the data_broadcast_id registered by "The Digital Network" with DVB.

A receiver compliant with this Profile shall respond to the value for the "ETSI core" application_type_code (see ETSI-MHEG Codes [20]).

If there is more than one application_type_code signalled in the PMT then the receiver shall use the boot_priority_hint to determine which to select. The data_broadcast_id_descriptor containing the selected application_type_code identifies the "boot-PID".

#### 9.3.2.1.4 boot_priority_hint

This provides a mechanism for the receiver to decide which application type to boot when it supports more than one of those broadcast and no other rules dictate how to chose between them. The hint is provided by the broadcaster, with 0 the lowest preference. No two application types shall have the same boot priority.

#### 9.3.2.1.5 application_specific_data_length

The value in this 8-bit field specifies the number of bytes of application specific data that follow.

#### 9.3.2.1.6 application_specific_data_byte

In this Profile these bytes may contain zero or more of the following sub-descriptors.

### 9.3.2.2 Network boot info sub-descriptor

The network_boot_info sub-descriptor provides a method by which a network operator may signal a running application to perform a specified action. This signal is provided in the data_broadcast_id descriptor so as to become independent of the data broadcast itself.

See table 6.

**Table 6: Network boot info sub-descriptor**

| | Size (bytes) | Value |
|---|---|---|
| network_boot{ | | |
| tag | 1 | 0x01 (Local to ETSIProfile1) |
| length | 1 | (N+2) |
| NB_version | 1 | |
| NB_action | 1 | 0x00 - No action<br>0x01 - Initiate autoboot for DSM-CC mounted applications, generate "NetworkBootInfo" EngineEvent for CI mounted applications.<br>0x02 - Generate "NetworkBootInfo" EngineEvent |
| NB_info | N | Allocated by Broadcaster |
| } | | |

If the value of the NB_version changes then the receiver shall execute the appropriate action as indicated by the value of the NB_action field as indicated in clause 11.8 and clause 9.3.7.

### 9.3.2.3 Service boot info sub-descriptor

The service_boot_info sub-descriptor indicates the location of any service-level application lifecycle signalling

See table 7.

**Table 7: Service boot info sub-descriptor**

| | Size (bytes) | Value |
|---|---|---|
| service_boot{ | | |
| tag | 1 | 0x02 (Local to ETSIProfile1) |
| length | 1 | 2 |

**Table 7: Service boot info sub-descriptor**

| | Size (bytes) | Value |
|---|---|---|
| assocTag | 2 | + |
| } | | |

The value of assocTag indicates the association tag (and thus component) which the receiver must filter for the signalling sections described in clause 9.3.6.

Note that if any DsmccDescriptorList sections are provided as part of the currently active object carousel (e.g. to provide stream event information), service-level application lifecycle signalling sections must be broadcast on the same component.

## 9.3.3 Acquisition of the ServiceGateway object

For a receiver supporting this Profile, the next step is to acquire the ServiceGateway object. This object is the root directory of the file system delivered by an object carousel and must be acquired before any other object can be downloaded. The presence of an object carousel is indicated by the Acquisition of the auto-boot object. This descriptor may be included in the second descriptor loop of a PMT corresponding to a PID on which the DSI message for an object carousel is broadcast, i.e. the boot-PID.

In this Profile the use of the Acquisition of the auto-boot object for signalling is mandatory. The consequence is that if a PMT second descriptor loop contains a data_broadcast_id_descriptor that provides signalling for this Profile, it shall also contain a Acquisition of the auto-boot object.

NOTE: A single PID shall only contain messages from a single object carousel and so only one Acquisition of the auto-boot object shall be present in any second descriptor loop. However, a single service may contain more than one object carousel. Consequently, the Acquisition of the auto-boot object may appear more than once in any single PMT.

### 9.3.3.1 carousel_id_descriptor

This descriptor is MPEG defined and in this Profile may be included in the second descriptor loop of a PMT.

See table 8.

**Table 8: Carousel identifier descriptor syntax**

| Syntax | Bits | Type | Value |
|---|---|---|---|
| carousel_identifier_descriptor { | | | |
| descriptor_tag | 8 | uimsbf | 0x13 |
| descriptor_length | 8 | uimsbf | N1 |
| carousel_id | 32 | uimsbf | |
| FormatID | 8 | uimsbf | |
| for( i=0; i<N1-5; i++ ){ | | | |
| private_data_byte | 8 | | |
| } | | | |
| } | | | |

#### 9.3.3.1.1 carousel_id

This 32-bit field identifies the object carousel with the corresponding carouselId.

#### 9.3.3.1.2 FormatID

This field is an 8-bit integer as defined in TR 101 202 [15]. It is not used in this Profile. Receivers compliant with this Profile shall ignore this field and any subsequent private data.

## 9.3.4 Acquisition of the auto-boot object

The location of the auto-boot object within the file system delivered by an object carousel may be explicit via signalling in the ServiceContextList of the ServiceGateway object or implicit by applying the default identification rules.

### 9.3.4.1 ServiceContextList

This defines the use of the MessageSubHeader::ServiceContextList for the BIOP::ServiceGatewayMessage. This structure is used to provide a potential location (in addition to the data_broadcast_id_descriptor in the PMT) for the inclusion of application specific data.

The ServiceContextList contains a loop over "ServiceID". In this Profile the 4-byte ServiceID is generated by the combination of data_broadcast_id and application_type_code. The inclusion of an entry in this list optional and is defined in the specification of each registered application type, i.e. an application type signalled in the PMT may not necessarily have a corresponding element in this list. There are no rules regarding the ordering of entries in this list.

If the ServiceContextList is absent, or does not list a supported application_type_code, then the implicit identification rules apply. See clause 9.3.4.2.

The generic form of the ServiceContextList adapted to carry application specific data is shown in table 9. Receivers shall be able to parse the structure regardless of the value(s) of application_type_code and shall be able to skip the application_specific_data_byte values if they do not recognize the application_type_code.

**Table 9: Service context list**

| | Size (bytes) | Value |
|---|---|---|
| ServiceContextList{ | | |
| serviceContextList_count | 1 | N1 |
| for( i=0; i<N1; j++){ | | |
| ServiceID{ | | |
| data_broadcast_id | 2 | |
| application_type_code | 2 | |
| } | | |
| application_specific_data_length | 2 | N2 |
| for( j=0; j<N2; j++ ){ | | |
| application_specific_data_byte | 1 | |
| } | | |
| } | | |
| } | | |

#### 9.3.4.1.1 serviceContextList_count

The value in this 8-bit field is the number of different application types that have data in this structure.

#### 9.3.4.1.2 ServiceID

The value in this 32-bit field is the concatenation of data_broadcast_id and application_type_code as shown in table 9.

A receiver compliant with this Profile shall recognize the ServiceID that is a concatenation of the data_broadcast_id 0x0106 and the selected application_type_code (see clause 9.3.2.1).

#### 9.3.4.1.3 application_specific_data_length

The value in this 16-bit field specifies the number of bytes of application specific data that follow.

#### 9.3.4.1.4 application_specific_data_byte

In this Profile these bytes may contain a set of language specific initial objects (as defined in table 10 ).

If the receiver does not provide a mechanism for selection based on language preference for any of the initial objects specified then the implicit identification rules apply. See clause 9.3.4.2. The receiver shall as a minimum support the language code "und".

See table 10.

**Table 10: Application boot service context list**

| | Size (bytes) | Value |
|---|---|---|
| application_specific_data{ | | |
| number_languages | 1 | N3 ≥ 1 |
| for( k=0; k<N3; k++){ | | |
| ISO_639_language_code | 3 | |
| initial_object_length | 1 | N4 ≥ 2 |
| for( l=0; l<N4; l++){ | | |
| initial_object_byte | 1 | |
| } | | |
| } | | |
| } | | |

#### 9.3.4.1.5 number_languages

Specifies the number of different languages for the application type.

#### 9.3.4.1.6 ISO_639_language_code

Specifies the target language of the application.

To specify a default choice or a single, language independent choice, the ISO language code "und" shall be used.

#### 9.3.4.1.7 initial_object_length

The value in this 8-bit field specifies the length of the null terminated initial object string. The null termination is included in the length.

The initial object string shall not be empty. So, its length shall be ≥ 2.

#### 9.3.4.1.8 initial_object_byte

In this Profile these 8-bit values form a null terminated string specifying the name of the initial object. This optionally may include a path from the root of the ServiceGateway with directories delimited by the character "/" (0x2F).

There shall be only one initial object specified for each combination of application_type_code and ISO_639_language_code.

### 9.3.4.2 Locating the initial object

#### 9.3.4.2.1 Explicit Initial Object Identified

If an appropriate **initial_object** field can be identified from the ServiceContextList of the ServiceGateway then the receiver shall first use this to locate the initial Application object. See clause 9.3.4.1.

If the **initial_object** field identifies a File object, this shall be the initial object.

If the **initial_object** field identifies a Directory object, the receiver shall use the default names "a" and "startup", in that order, to locate the initial Application object in this Directory object.

#### 9.3.4.2.2 No Explicit Initial Object Identified

If no appropriate **initial_object** field can be identified in the ServiceContextList message or the initial Application object identified cannot be acquired from the object carousel, then the receiver shall use the following default names to locate the initial Application object in the ServiceGateway object (in the following order):

- "DSM://a"
- "DSM://startup"

For example, for the **initial_object** bytes "my_app", the receiver shall work down the following list (in the order shown):

"DSM://my_app"

"DSM://my_app/a"

"DSM://my_app/startup"

"DSM://a"

"DSM://startup"

### 9.3.5 Example of steps required for auto-boot

1) "Identification of auto-boot application":

   a) Tune to the transport stream of the service (MPEG Program).

   b) Acquire the PAT.

   c) Acquire the PMT of the service using the information in the PAT.

   d) Search through the second descriptor loops in the PMT, i.e. the descriptor loop for each elementary stream, for an instance of the data_broadcast_id_descriptor containing an appropriate application_type_code value. This identifies the boot-PID.

   **If there is no match, there is no auto-boot application for this service.**

2) "Acquisition of the ServiceGateway object":

   a) Search through the second descriptor loop corresponding to the boot-PID for an instance of the Acquisition of the auto-boot object.

   b) Acquire the DSI from this elementary stream.

   c) Acquire the DII message (DII1) that identifies the module (Module1) containing the ServiceGateway object, using information in the DSI message (IOR1).

   d) Acquire Module1 using information in DII1.

   e) Extract the ServiceGateway object from Module1 using information in IOR1.

3) "Acquisition of the auto-boot object":

   a) Get the name of the initial File object (in this case containing an MHEG-5 Application object) using the ServiceContextList information in the ServiceGateway message and/or default values. The location of this File object is relative to the ServiceGateway object.

   b) If the name points to subdirectories follow the whole path, filtering the appropriate Directory objects (essentially repeating steps  to  in a more generic fashion as required).

   c) Acquire the DII message (DII2) that identifies the module (Module2) containing the initial File object, using information in the ServiceGateway object (IOR2).

   d) Acquire Module2 using information in DII2.

   e) Extract the initial File object from Module2 using information in IOR2.

### 9.3.6 Service-level application lifecycle signalling

Application lifecycle signalling at the service level is a way to signal actions to the receiver via the broadcast stream that are completely independent of any currently running applications.

These actions are signalled using DsmccDescriptorList sections (i.e. DSMCC_sections with table_id of 0x3D) with particular defined table_id_extensions. The location of these sections is specified by the assocTag field value contained in the service_boot_info sub-descriptor (see clause 9.3.2.1).

The table_id_extensions shown in table 11 are supported in this Profile. Note that there is no requirement for a receiver to parse the contents of the section; the table_id_extension on its own is enough to signal which action the receiver should perform.

**Table 11: Reserved extensions for DsmccDescriptorList sections**

| table_id_extension | Action |
|---|---|
| 0xFFFF | All applications that originated from the mounted DSM-CC carousel shall be terminated (if the application is currently running) or removed from the application stack. The boot process shall then be restarted. No CI mounted applications are terminated. |

The action is performed only once for a particular table_id_extension and table version: the receiver shall not perform the action again until the table version changes.

### 9.3.7 Network-level application lifecycle signalling

To provide network level signalling all platforms must monitor changes to the current service PMT. Support for the data_broadcast_id descriptor to identify auto-boot applications is not required when the service includes AIT signalling.

#### 9.3.7.1 Auto mount broadcast file system

If a PMT of a service has a data_broadcast_id descriptor for a component then this indicates that the component carries a DSI of an object carousel to be mounted and optionally searched for an auto-boot application.

If no auto-boot application is found then the object carousel remains mounted. A broadcast auto-boot application may be introduced subsequently and/or an application may be introduced by a CI module using a RequestStart message (see clause 14.9.3.2).

#### 9.3.7.2 network_boot_info sub-descriptor

All applications that originated from the mounted DSM-CC carousel shall be terminated (if the application is currently running), or removed from the application stack, when a network_boot_info sub-descriptor (contained in a data_broadcast_id descriptor) either appears or changes (as indicated by the NB_version), and the NB_action is set to 0x01 (see clause 9.3.2.1). The boot process shall then be restarted. This shall occur regardless of the signalling provided by any current AIT.

A CI mounted application shall receive a NetworkBootInfo EngineEvent if a network_boot_info sub-descriptor either appears or changes and the NB_action is set to 0x01 (see clause 9.3.2.1).

An application can access the value of the NB_info field at any time via the resident program GetBootInfo (see clause 9.3.2.1 and clause 11.10.9.3).

#### 9.3.7.3 data_broadcast_id descriptor

If a data_broadcast_id descriptor is removed from a service then all applications that originated from the mounted DSM-CC carousel shall be terminated (if the application is currently running) or removed from the application stack. The boot process shall then be restarted.

#### 9.3.7.4 carousel_id descriptor

If the carousel id, indicated in the carousel_id descriptor, changes on the mounted carousel then all applications that originated from the mounted DSM-CC carousel shall be terminated (if the application is currently running) or removed from the application stack. The boot process shall then be restarted.

If the carousel id changes on a non-mounted carousel (i.e. a carousel which does not contain the auto-boot application as indicated by the data_broadcast_id descriptor) then no action is taken.

#### 9.3.7.5 Carousels moving components

If a mounted carousel, indicated by a valid carousel_id descriptor and data_broadcast_id descriptor, moves between components or pids no action is to be taken unless the carousel id has been changed. In this case all applications that originated from the mounted DSM-CC carousel shall be terminated (if the application is currently running) or removed from the application stack. The boot process shall then be restarted. This allows receivers to ignore which components or pids carousels are provided on, and to only act upon changes in the carousel id in the currently mounted carousel.

# 10 Security

Applications conforming to this Profile do not require authentication to run. Some small risk is present in allowing applications freedom to tune to services other than the one from which they are launched. However, the application lifecycle determines that applications leaving their original service shall be stopped, so the risk arising from tuning behaviour is minimal.

In the present document persistent storage management is described in clause 14.6 and clause 14.7. Any risk that might lead to the need to securely manage the control of this storage resource is therefore mitigated by these factors.

Extensions to this Profile that include 'return channel' interactivity will include provision for security mechanisms.

# 11 MHEG-5 engine profile

This clause provides the detailed Profile of the MHEG-5 engine required in compliant digital TV receivers. This Profile is an "application domain" in the terms set out in Annex D of ISO/IEC 13522-5 [14].

This "application domain" is referred to as "*ETSIEngineProfile1*".

## 11.1 Basic specification

The engine shall be compliant with the ISO/IEC 13522-5 [14] (MHEG-5) specification for a Hypermedia presentation engine using the "application domain" definition provided by the present document. The following are also considered to be in force:

- the corrigenda to ISO/IEC 13522-5 [14] published as ISO/IEC 13522-5:1997/Cor.1:1999(E) [21];
- the additional language specifications described in clause 11.12.

See also:

- the clarifications and restrictions described in clause 11.13; and
- the cache model described in clause 15.4.

## 11.2 Object interchange format

The ASN.1 notation (see ITU-T Recommendation X.680 [6]) defined in Annex A of ISO/IEC 13522-5 [14] shall be used as the application interchange format. The encoding of the MHEG-5 objects from this ASN.1 syntax shall make use of the *Distinguished Encoding Rules* (DER) defined in ITU-T Recommendation X.690 [7].

In line with the recommended error handling behaviour in the MHEG-5 specification:

- the engine shall ignore unrecognized objects;
- the engine shall ignore unrecognized fields;
- the engine shall ignore unrecognized elementary actions;
- the engine may ignore Ingredients that include an unrecognized ContentHook;
- the engine shall ignore the ObjectInformation exchanged attributes where encoded.

## 11.3 Set of classes

Table 12 identifies the classes that a receiver implementing the *ETSIEngineProfile1* shall implement.

**Table 12: Classes supported by this engine profile (Sheet 1 of 2)**

| Class | Abstract/concrete (NOTE 1) | Required (Y/N) (NOTE 2) | Notes |
|---|---|---|---|
| Root | A | Y | |
| Group | A | Y | |
| Application | C | Y | |
| Scene | C | Y | |
| Ingredient | A | Y | |
| Link | C | Y | |
| Program | A | Y | |
| ResidentProgram | C | Y | See clause 11.10. |
| RemoteProgram | C | N | |
| InterchangedProgram | C | N | |

**Table 12: Classes supported by this engine profile (Sheet 2 of 2)**

| Class | Abstract/concrete (NOTE 1) | Required(Y/N) (NOTE 2) | Notes |
|---|---|---|---|
| Palette | C | N | |
| Font | C | N | See clause 13. |
| CursorShape | C | N | |
| Variable | A | Y | See clause 11.11. |
| BooleanVariable | C | Y | |
| IntegerVariable | C | Y | |
| OctetStringVariable | C | Y | |
| ObjectRefVariable | C | Y | |
| ContentRefVariable | C | Y | |
| Presentable | A | Y | |
| TokenManager | A | Y | |
| TokenGroup | C | Y | |
| ListGroup | C | Y | |
| Visible | A | Y | |
| Bitmap | C | Y | See clause 11.5.2 and clause 12.8. |
| LineArt | C | N | See clause 12.5. See footnote (NOTE 3). |
| Rectangle | C | Y | |
| DynamicLineArt | C | Y | See clause 12.5. |
| Text | C | Y | See clause 13.6. |
| Stream | C | Y | |
| Audio | C | Y | See clause 11.4.1.4. and clause 11.5.3. |
| Video | C | Y | |
| RTGraphics | C | N | |
| Interactible | A | Y | |
| Slider | C | Y | See clause 13.9. |
| EntryField | C | Y | See clause 13.7. |
| HyperText | C | Y | See clause 13.8. |
| Button | A | N | |
| HotSpot | C | N | |
| PushButton | C | N | |
| SwitchButton | C | N | |
| Action | C | Y | |

NOTE 1: Abstract classes are not "interchanged" (i.e. not directly used) in *ETSIEngineProfile1* MHEG-5 applications.

NOTE 2: "Y" = yes, i.e. receivers implementing *ETSIEngineProfile1* shall support these classes. "N" = no, i.e. receivers implementing *ETSIEngineProfile1* are not required to support these classes. Also, classes marked "N" may not be completely defined in the present document.

NOTE 3: The LineArt class shall not be instantiated, i.e. no MHEG-5 applications conforming to this Profile shall include LineArt objects. No content encoding is defined for this class in this Profile. However, engine implementations may include the LineArt class effectively as an abstract class supporting instantiable classes based on LineArt such as Rectangle and DynamicLineArt.

## 11.4 Set of features

All receivers shall implement all of the effects of MHEG-5 actions and the internal behaviours of MHEG-5 classes required under clause 11.3. Table 13 identifies the *ETSIEngineProfile1* requirement with regard to features defined as "optional" within ISO/IEC 13522-5 [14].

**Table 13: Requirements for MHEG-5 "optional" features**

| Feature | Required (Y/N) (NOTE) | Notes |
|---|---|---|
| Caching | Y | See clause 15.4. |
| Ancillary connections | N | See clause 11.4.2. |
| Cloning | Y | The receiver shall support cloning at least of the following classes: Text, Bitmap and Rectangle. |
| Free-moving cursor | N | See clause 11.4.2. |
| Bitmap scaling | N | Limited scaling is required for I-frame bitmaps, but not for PNGs.See clause 11.5.2.1. |
| Video scaling | Y | See clause 14.4.4. |
| Stacking of applications | Y | See clause 14.8. |
| Trick mode | N | Not applicable for any currently defined stream-items' content type. See clause 11.4.2 and clause 14.4.3. |
| NOTE : "Y" = yes, i.e. receivers implementing *ETSIEngineProfile1* shall support these features. | | |

### 11.4.1 GetEngineSupport "feature" strings

Table 14 identifies the GetEngineSupport feature strings that receivers implementing *ETSIEngineProfile1* shall support.

**Table 14: *ETSIEngineProfile1* GetEngineSupport behaviour (Sheet 1 of 2)**

| String | | Constraint |
|---|---|---|
| **Standard** | **Short** | |
| AncillaryConnections | ACo | Shall return "false". See clause 11.4.2. |
| ApplicationStacking | ASt | Shall return "true". See clause 14.8. |
| Cloning | Clo | Shall return "true". |
| FreeMovingCursor | FMC | Shall return "false". See clause 11.4.2. |
| MultipleAudioStreams(N) | MAS(N) | Shall return "true" for N≤1. See clause 11.4.1.4. |
| MultipleVideoStreams(N) | MVS(N) | Shall return "true" for N≤1. |
| OverlappingVisibles(N) | OvV(N) | Shall return "true" for all values of N. See clause 14.4.2 and clause 12.4. |
| Scaling | Sca | Shall return "false". See clause 14.4.4. |
| SceneAspectRatio(W,H) | SAR(W,H) | Shall return "true" for (W, H) is (4,3) or (16,9). |
| SceneCoordinateSystem(X,Y) | SCS(X,Y) | Shall return "true" for (X,Y) is (720,576). See clause 12.1. |
| TrickModes | TrM | Shall return "false". See clause 11.4.2. |
| VideoScaling(CHook,X,Y) (NOTE 1) | VSc(CHook,X,Y) (NOTE 1) | Shall return "true" for the combinations of CHook, X and Y tabulated below that are supported by the implementation. See clause 14.4.4. |

| CHook | X | Y | Status |
|---|---|---|---|
| 10 | 1440 | 1152 | Optional |
| | 720 | 576 | Required |
| | 360 | 288 | |

**Table 14: *ETSIEngineProfile1* GetEngineSupport behaviour (Sheet 2 of 2)**

<table>
<tr><th colspan="2">String</th><th>Constraint</th></tr>
<tr><th>Standard</th><th>Short</th><th></th></tr>
<tr><td>BitmapScaling(CHook,X,Y)<br>(NOTE 2)</td><td>BSc(CHook,X,Y)<br>(NOTE 2)</td><td>Shall return "true" for the combinations of CHook, X and Y tabulated below that are supported by the implementation. See clause 14.4.4.
<table>
<tr><th>CHook</th><th>X</th><th>Y</th><th>Status</th></tr>
<tr><td rowspan="3">2</td><td>1440</td><td>1152</td><td>Optional</td></tr>
<tr><td>720</td><td>576</td><td rowspan="2">Required</td></tr>
<tr><td>360</td><td>288</td></tr>
</table></td></tr>
<tr><td>VideoDecodeOffset(CHook,Level)<br>(NOTE 1)</td><td>VDO(CHook,Level)<br>(NOTE 1)</td><td>Shall return "true" for the combinations of CHook and Level tabulated below that are supported by the implementation.<br><br>See table 15.</td></tr>
<tr><td>BitmapDecodeOffset(CHook,Level)<br>(NOTE 2)</td><td>BDO(CHook,Level)<br>(NOTE 2)</td><td>Shall return "true" for the combinations of CHook and Level tabulated below that are supported by the implementation.<br><br>See table 16.</td></tr>
<tr><td>UniversalEngineProfile(N)</td><td>UEP(N)</td><td>Shall return "true" when N corresponds to a Profile supported by this receiver. See ETSI-MHEG Codes [20].<br><br>Shall also return "true" when N is a manufacturer specific string identifying an element of the receiver software or hardware. Receivers may respond to a number of such strings, each representing, for example, a module of code. See clause 11.4.1.3.<br><br>See also clause 17.19.1.</td></tr>
<tr><td>DebugPackageSupport (NOTE 3)</td><td>DPS</td><td>Shall return "true" if the receiver implements the Debug package. See clause 11.14.</td></tr>
<tr><td>SIPackageSupport (NOTE 3)</td><td>SPS</td><td>Shall return "true" if the receiver implements the Service Information package. See clause 11.15.</td></tr>
<tr><td colspan="3">NOTE 1: This CHook is that for a Stream. See table 18.<br>NOTE 2: This CHook is that for a Bitmap. See table 18.<br>NOTE 3: An application using the Debug or Service Information packages shall check for the availability of the package using these GetEngineSupport strings.</td></tr>
</table>

Unrecognized requests will always result in GetEngineSupport returning false. This implies that future MHEG-5 engine implementations supporting more than these minimum features will recognize additional strings. These strings will have to be agreed by DTG Testing Ltd so that consistency is maintained.

**Receivers shall truthfully reflect their capability when interrogated with GetEngineSupport.**

### 11.4.1.1 VideoDecodeOffset

Table 15 shows the parameters for VideoDecodeOffset.

**Table 15: VideoDecodeOffset parameters**

| CHook | Level | Description | Note |
|---|---|---|---|
| 10 | 0 | VideoDecodeOffset attribute is constrained so that the decoded and scaled video is fully on screen, i.e. the resulting top left and bottom right corner coordinates are both in the range (0, 0) to (720, 576). | Required |
| | 1 | VideoDecodeOffset attribute can be such that the decoded and scaled video can be partially off screen.<br><br>For quarter-screen video, at least half of the decoded image height and width must remain within the display raster.<br><br>For full-screen video, half of the decoded image height and one quarter of the width must remain within the display raster. | Optional |

See the examples under clause 11.12.10.2:

- "Example 1" on page 82 requires level 0.
- "Example 2" on page 83 requires level 1.
- "Example 3" on page 84 requires level 1 and 1440 x 1152 scaling support.

### 11.4.1.2 BitmapDecodeOffset

Table 16 shows the parameters for BitmapDecodeOffset.

**Table 16: BitmapDecodeOffset parameters**

| CHook | Level | Description | Note |
|---|---|---|---|
| 2 | 0 | BitmapDecodeOffset attribute is constrained so that the decoded and scaled bitmap is fully on screen, i.e. the resulting top left and bottom right corner coordinates are both in the range (0, 0) to (720, 576). | Required |
| | 1 | BitmapDecodeOffset attribute can be such that the decoded and scaled bitmap can be partially off screen.<br><br>For quarter-screen bitmap, at least half of the decoded image height and width must remain within the display raster.<br><br>For full-screen bitmap, half of the decoded image height and one quarter of the width must remain within the display raster. | Optional |

See clause 11.12.7 and the examples under clause 11.12.10.2:

- "Example 1" on page 82 requires level 0.
- "Example 2" on page 83 requires level 1.
- "Example 3" on page 84 requires level 1 and 1440 x 1152 scaling support.

#### 11.4.1.3 Engine identification strings

All receivers shall return true to at least:

- one string that uniquely identifies the receiver via its make, model and version number. This shall be of the form "mmmcccvvv", where "mmm" uniquely identifies the manufacturer, "ccc" is a unique model code and "vvv" is the principal version of the complete receiver software build;
- one string that identifies the MHEG engine provider and version number. This shall be of the form "MHGmmmvvv", where "mmm" uniquely identifies the manufacturer of the MHEG engine and "vvv" is the version number of the currently embedded build;
- one string that identifies the DSMCC stack provider and version number. This shall be of the form "DSMmmmvvv", where "mmm" uniquely identifies the manufacturer of the DSMCC stack and "vvv" is the version number of the currently embedded build.

The subfields "mmm", "ccc" and "vvv" shall be encoded as three octets with values in the range 0x21 to 0x7E inclusive.

So for example, a particular receiver might recognize the following strings:

UEP(FEG001103)

UEP(MHGFEG056)

UEP(DSMFEG017)

Whilst another receiver might recognize the following:

UEP(AST003213)

UEP(MHGBUP122)

UEP(DSMCDU008)

Other strings may also return true.

#### 11.4.1.4 Audio stream decoders

Two sources of audio are considered in the context of the *ETSIEngineProfile1* MHEG-5 engine:

1) MPEG audio ISO/IEC 13818-3 [11] data "live" from a stream as it is broadcast (e.g. the sound track for a TV programme).
2) MPEG audio ISO/IEC 13818-3 [11] data from memory (see clause 11.5.3).

*ETSIEngineProfile1* receivers shall provide one MPEG audio ISO/IEC 13818-3 [11] decoder which can decode data either from a stream **or** from an object in memory.

The meaning of MultipleAudioStreams(N) and MAS(N) for N > 1 has not yet been defined.

### 11.4.2 Not required features

Features identified as not required in this Profile **shall not** be implemented by receivers conforming to this Profile. This constraint is to ensure that, when implemented, these features are correctly specified for regions implementing this Profile.

## 11.5 Content data encoding

Table 17 identifies the minimum set of coding of attributes that shall be supported by the engine.

**Table 17: Content table**

| Attribute | Permissible values |
|---|---|
| FontAttributes | See clause 13.4.1. |
| FontName | See clause 13.3.3. |
| AbsoluteColour | See clause 12.3.3. |
| CharacterSet | See table 63. |
| TransitionEffect | Not required feature. |

Table 18 identifies the minimum set of data types that shall be supported by the engine for each type of content. It also identifies the content hook values for each data type.

**Table 18: Encoding table**

| Type of content | Specification (data types) | Hook values |
|---|---|---|
| Font | (None specified) see clause 13.3.1. | |
| Palette | (None specified) | |
| Bitmap | Reserved | 1 |
| | MPEG-2 Intra frame ISO/IEC 13818-2 [10]. See clause 12.8. | 2 |
| | Reserved | 3 |
| | PNG [17] bitmap. See clause 12.7. | 4 |
| Text | See clause 13.6. | 10 |
| EntryField | See clause 13.2.<br>See clause 13.7. | 10 |
| HyperText | See clause 13.8. | 10 |
| Stream | A broadcast MPEG program. | 10 |
| | A "file" containing a single elementary stream that can be decoded from memory. Data formats are described in clause 11.5.3.<br>In this Profile Audio is the only stream component that can be decoded from memory. | 11 |
| LineArt | (None specified) | |
| CursorShape | (None specified) | |
| InterchangedProgram | (None specified) | |

### 11.5.1 Use of negative hook values

Negative hook values are reserved for use by receiver manufacturers to signal manufacturer specific encoding formats. These shall never be signalled by a broadcaster and may only be used by the manufacturer for local applications.

### 11.5.2 Bitmap objects

#### 11.5.2.1 Scaling

Scaling (the ScaleBitmap action) shall be supported for bitmap objects with MPEG I-frame content. See clause 12.8. Scaling shall not be supported for Bitmap objects using PNG [17] bitmaps.

#### 11.5.2.2 Tiling

Support for tiling is only required for PNG [17] bitmaps.

#### 11.5.2.3 Transparency

Transparency can be encoded within the PNG [17] bitmaps. Support for object level transparency (i.e. the Transparency internal attribute of the Bitmap class) is not required for any bitmap type. See clause 12.4.1.

### 11.5.3 Stream "memory" formats

In this Profile the only StreamComponent that accepts a Storage specification of memory is Audio. Video can only be played from Stream.

#### 11.5.3.1 Audio

Each "file" of audio content is an OctetString carrying audio elementary stream data. Each "file" delivers an integer number of audio access units and the first byte of each file is the first byte of an audio access unit.

## 11.6 User input

### 11.6.1 Remote control functions

Figure 3 illustrates the functions of the receiver remote control. This is provided to help explain the "function groups". The physical appearance (including possible labelling) of the remote control and the methods of interaction delivering the required functions may be quite different from the one illustrated.

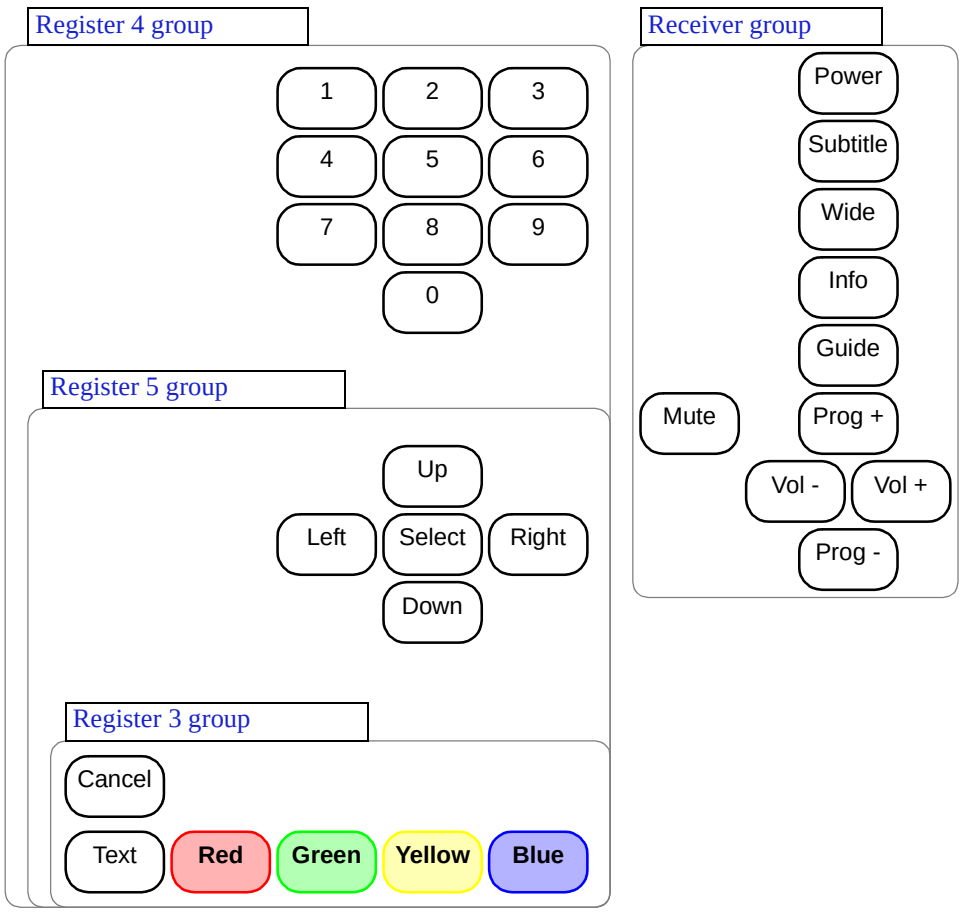



**Figure 3: Remote control function groups**

#### 11.6.1.1 Receiver group

The input provided by this group **is never available to a running MHEG-5 application**.

This group provides the most frequently used receiver control functions. They have the same effect whether the viewer is watching a TV service, an MHEG-5 service or using other receiver screens. For example, the "Program up" and "Program down" always have the effect of changing channel.

All other receiver specific functions (e.g. to invoke the receiver set-up screens) are also within this group.

#### 11.6.1.2 Register 3 group

The input provided by this group is **always available to a running MHEG-5 application**.

In some receiver implementations this group may also be used for interacting with receiver applications such as the "Info" and "Guide" screens when there is either no running MHEG-5 application or the current MHEG-5 application has been "paused" (see clause 14.8). This group shall not be used for receiver "channel surfing".

#### 11.6.1.3 Register 4 group

The input provided by this group **is sometimes available to a running MHEG-5 application**.

MHEG-5 applications have exclusive use **only after** the viewer has *entered* the broadcast application. This is to avoid confusion with "channel surfing" for which these groups may also be used.

#### 11.6.1.4 Register 5 group

As Register 4 group.

### 11.6.2 UserInput registers

Receivers shall implement input event registers 3, 4 and 5 defined in table 19. The DAVIC defined registers 1 and 2 are shown for information only.

**Table 19: InputEvent Registers**

| UserInput EventData | Function name | Register number | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 1 | Up | ✓ | ✓ | | ✓ | ✓ |
| 2 | Down | ✓ | ✓ | | ✓ | ✓ |
| 3 | Left | ✓ | ✓ | | ✓ | ✓ |
| 4 | Right | ✓ | ✓ | | ✓ | ✓ |
| 5-14 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 respectively | ✓ | | | ✓ | |
| 15 | Select | ✓ | ✓ | | ✓ | ✓ |
| 16 | Cancel | ✓ | ✓ | ✓ | ✓ | ✓ |
| 17 | Help | ✓ | ✓ | | | |
| 18-99 | Reserved by DAVIC | | | | | |
| 100 | Red | | | ✓ | ✓ | ✓ |
| 101 | Green | | | ✓ | ✓ | ✓ |
| 102 | Yellow | | | ✓ | ✓ | ✓ |
| 103 | Blue | | | ✓ | ✓ | ✓ |
| 104 | Text | | | ✓ | ✓ | ✓ |
| 105-256 | Reserved for future specification | | | | | |
| ≥257 | Vendor specific | | | | | |

NOTE: Applications may switch between InputEvent registers dynamically as they progress. Selection of an InputEvent register shall not have side effects on other aspects of the receiver behaviour such as video presentation.

### 11.6.3 Implementation of this interaction model

The receiver is responsible for implementing the set of input functions defined for the input event registers (see clause 11.6). How applications use the different InputEvent registers and the resultant user experience is addressed through authoring guidelines (see clause 17.1.1).

## 11.7 Semantic constraints on MHEG-5 applications

As implied by the capabilities of the engine in clause 11.4.1.

## 11.8 EngineEvents

Table 20 lists the minimum set of engine events that the engine is required to support.

**Table 20: Required engine events (Sheet 1 of 2)**

| EventData | | Notes |
|---|---|---|
| Name | Value | |
| | < 0 | Manufacturer specific. |
| Reserved | 0 | Reserved. |
| | 1 | |
| GroupIDRefError | 2 | This event shall be raised if an application attempts to access an application or scene object that cannot be provided by the file system. |
| ContentRefError | 3 | This event shall be raised if an application attempts to access referenced content that cannot be provided by the file system. |
| TextKeyFunction | 4 | Generated when the user activates the "Text" function and there is an active scene object. This event is raised independently of the InputEvent register selected at the current moment or whether any interactible has the InteractionStatus of True.<br><br>If "Text" function causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first.<br><br>See clause 11.6.1. |
| Reserved | 5 | Reserved. |
| VideoPrefChanged | 6 | Generated when the user preferences for video aspect ratio handling change in a way that would alter the return values for the VideoToGraphics resident program, see clause 11.10.10.1.<br><br>Note that this event is **not** generated when the broadcast video changes aspect ratio, nor when a change affects only **display** format conversion. |
| PauseResume | 7 | Generated after the MHEG-5 engine process resumes processing after a period where it is descheduled. See clause 14.8. |
| Reserved | 8 | Reserved. |
| NetworkBootInfo | 9 | Generated in response to changes in the relevant data_broadcast_id_ descriptor when:<br>• the network_boot_sub-descriptor either appears or changes (as indicated by the NB_version), and the NB_action is set to 0x02;<br>• the network_boot_info sub-descriptor either appears or changes (as indicated by the NB_version), the NB_action is set to 0x01, and the current application is being executed from the CI file system.<br><br>The application can access the value of the NB_info field at any time via the resident program GetBootInfo.<br><br>Note that this event will never be received unless a data_broadcast_id_ descriptor with appropriate network_boot_info sub-descriptor is used in the signalling method. See clause 9.1.4. |

**Table 20: Required engine events (Sheet 2 of 2)**

| EventData | | Notes |
|---|---|---|
| **Name** | **Value** | |
| Reserved | 10 to 15 | Reserved. |
| CancelKeyFunction | 16 | Generated when the user activates the "cancel" function from the appropriately mapped key and there is an active Scene object.This event is raised independently of the InputEvent register selected at the current moment or whether any interactible has the InteractionStatus of True.<br><br>If a key press causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first.<br><br>See clause 11.6.1. |
| Reserved | 17 to 99 | Reserved. |
| RedKeyFunction | 100 | Generated when the user activates the appropriate colour key and there is an active scene object. This event is raised independently of the InputEvent register selected at the current moment or whether any interactible has the InteractionStatus of True.<br><br>If a key press causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first.<br><br>See clause 11.6.1. |
| GreenKeyFunction | 101 | |
| YellowKeyFunction | 102 | |
| BlueKeyFunction | 103 | |
| Reserved | All other values | Reserved. |

### 11.8.1 Object retrieval errors

The engine events GroupIDRefError and ContentRefError shall be raised if an Action directly attempts to access an object or some content that cannot be provided by the file system.

The above does not define the behaviour of file access errors that are the consequence of file accesses from resident programs. In this Profile the only ResidentPrograms that attempt to access files are CheckContentRef and CheckGroupIDRef. These do not raise any engine events (the return parameters from the program include an indication of file availability).

## 11.9 Protocol mapping and external interaction

Table 21 shows the protocol mapping and external internal interaction.

**Table 21: Protocol mapping**

| MHEG-5 entity | Mapping needed | Semantics |
|---|---|---|
| OpenConnection, CloseConnection | Mapping to connection management | Not required in *ETSIEngineProfile1*.<br><br>See clause 11.4.2. |
| RemoteProgram objects | Mapping to RemoteProgram call protocol in the application domain | |
| Application name space in case a TransitionTo action uses the ConnectionTag parameter | Mapping to the name space of the application domain | |
| Persistent storage name space | Mapping to the name space of the persistent storage | See clause 14.6. |

**Table 21: Protocol mapping**

| MHEG-5 entity | Mapping needed | Semantics |
|---|---|---|
| Stream actions | Mapping to the stream interface of the application domain | See clause 16.3 |
| Stream events | Mapping to stream states and stream events in the application domain | |

# 11.10 Resident Programs

Table 22 lists the resident programs that a *ETSIEngineProfile1* receiver shall implement.

**Table 22: *ETSIEngineProfile1* mandatory resident programs**

| Resident program | | Invocation | | | Reference |
|---|---|---|---|---|---|
| | | Typical use | | Never Fork | |
| Description | Name | Call | Fork | | |
| GetCurrentDate | GCD | ✓ | | | clause 11.10.4.2. |
| FormatDate | FDa | ✓ | | | clause 11.10.4.3. |
| GetDayOfWeek | GDW | ✓ | | | clause 11.10.4.4. |
| Random | Rnd | ✓ | | | clause 11.10.5.1. |
| CastToContentRef (NOTE 1) | CTC | ✓ | | | clause 11.10.6.1. |
| CastToObjectRef (NOTE 1) | CTO | ✓ | | | clause 11.10.6.2. |
| GetStringLength | GSL | ✓ | | | clause 11.10.7.2. |
| GetSubString | GSS | ✓ | | | clause 11.10.7.3. |
| SearchSubString | SSS | ✓ | | | clause 11.10.7.4. |
| SearchAndExtractSubString | SES | ✓ | | | clause 11.10.7.5. |
| SI_GetServiceIndex | GSI | ✓ | | | clause 11.10.8.1. |
| SI_TuneIndex | TIn | ✓ | | ✓ (NOTE 2) | clause 11.10.8.2. |
| SI_TuneIndexInfo | TII | ✓ | | | clause 11.10.8.4. |
| SI_GetBasicSI | BSI | ✓ | | | clause 11.10.8.3. |
| GetBootInfo | GBI | ✓ | | | clause 11.10.9.3. |
| CheckContentRef | CCR | | ✓ | | clause 11.10.9.1. |
| CheckGroupIDRef | CGR | | ✓ | | clause 11.10.9.2. |
| VideoToGraphics | VTG | ✓ | | | clause 11.10.10.1. |
| SetWidescreenAlignment | SWA | ✓ | | | clause 11.10.10.2. |
| GetDisplayAspectRatio | GDA | ✓ | | | clause 11.10.10.3. |
| CI_SendMessage | CIS | ✓ | | | clause 11.10.11.1. |
| SetSubtitleMode | SSM | ✓ | | | clause 11.10.10.4. |
| WhoAmI | WAI | ✓ | | | clause 11.10.12.1. |

NOTE 1: See "Type conversion" in clause 13.1.1.
NOTE 2: Applications shall not invoke this resident program using Fork.

## 11.10.1 Typical use

Table 22 provides an indication of the method by which the resident program is typically invoked (Call or Fork). This indication is informative. **The function of the resident program shall be identical whether invoked by Call or Fork.**

See also clause 17.14.

## 11.10.2 Program names

The three character "Name" indicated in table 22 shall be used to invoke the ResidentProgram.

## 11.10.3 Encoding of resident program names

The names of resident programs are conveyed in OctetStrings. The character encoding for names of resident programs uses ISO/IEC 8859-1. All of the characters used in table 22 lie in the code range 0x00 to 0x7E.

### 11.10.3.1 Case sensitive names

The names of resident programs are case sensitive.

## 11.10.4 Date and time functions

### 11.10.4.1 Day, date and time functions

Some of these functions return textual information. In all cases this shall use the UTF-8 representation of character codes selected from the character repertoire in table 78.

### 11.10.4.2 GetCurrentDate

Retrieves the current local date and time.

#### 11.10.4.2.1 Synopsis

GCD( date, time )

#### 11.10.4.2.2 Arguments

Arguments for GetCurrentDate are shown in table 23.

**Table 23: GetCurrentDate parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | date | The modified Julian date. Encoded as the number of days since midnight on November 17, 1858. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | time | The current time encoded as the number of seconds since midnight. |

### 11.10.4.3 FormatDate

Format a string representing a date according to a specifiable format.

#### 11.10.4.3.1 Synopsis

FDa( dateFormat, date, time, datestring )

#### 11.10.4.3.2 Arguments

Arguments for FormatDate are shown in table 24.

**Table 24: FormatDate parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericOctetString | dateFormat | A string specifying the format for presenting the date and time in the output string. See below. |
| input | GenericInteger | date | The Modified Julian date. Encoded as the number of days since Midnight on November 17, 1858. |
| input | GenericInteger | time | The time encoded as the number of seconds since midnight. |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | dateString | The resultant formatted date and time. |

#### 11.10.4.3.3 Description

The dateString is formed by taking the input dateFormat string and expanding the fields prefixed with "%" as described:

```
‘%Y’ Year, 4 digits
‘%y’ Year, last 2 digits
‘%X’ Month, with padding zero (01-12)
‘%x’ Month, without padding zero (1-12)
‘%D’ Day, with padding zero (01-31)
‘%d’ Day, without padding zero (1-31)
‘%H’ Hour, with padding zero (00-23)
‘%h’ Hour, without padding zero (0-23)
‘%I’ Hour, with padding zero (01-12)
‘%i’ Hour, without padding zero (1-12)
‘%M’ Minutes, with padding zero (00-59)
‘%m’ Minutes, without padding zero (0-59)
‘%S’ Seconds, with padding zero (00-59)
‘%s’ Seconds, without padding zero (0-59)
‘%A’ AM/PM indication
‘%a’ am/pm indication
‘%%’ single"%" character
```

For example, on June 4, 1995, at 16:56, with dateFormat "%Y-%x-%d %I:%M %a" the result in dateString is "1995-6-4 4:56 pm".

### 11.10.4.4 GetDayOfWeek

Returns the day of the week.

#### 11.10.4.4.1 Synopsis

GDW( date, dayOfWeek )

#### 11.10.4.4.2 Arguments

Arguments for GetDayOfWeek are shown in table 25.

**Table 25: GetDayOfWeek parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | date | The Modified Julian date. Encoded as the number of days since Midnight on November 17, 1858. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | dayOfWeek | An integer representing the day of the week. 0 represents Sunday, 1 Monday etc. |

## 11.10.5 Random number function

### 11.10.5.1 Random

Returns a random number.

#### 11.10.5.1.1 Synopsis

Rnd( num, random )

#### 11.10.5.1.2 Arguments

Arguments for Random are shown in table 26.

**Table 26: Random parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | num | Specifies the upper limit to the range of random numbers returned. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | random | A random number in the range 1 to num inclusive. |

## 11.10.6 Type conversion functions

### 11.10.6.1 CastToContentRef

Casts an OctetString to a ContentReference.

#### 11.10.6.1.1 Synopsis

CTC( string, contentRef )

#### 11.10.6.1.2 Arguments

Arguments for CastToContentRef are shown in table 27.

**Table 27: CastToContentRef parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericOctetString | string | |
| output | GenericContentReference (Shall provide an IndirectReference to an ContentRefVariable) | contentRef | |

#### 11.10.6.1.3 Description

Casts the OctetString variable string to the ContentReference variable contentRef.

### 11.10.6.2 CastToObjectRef

Casts an OctetString and Object Identifier to an Object Reference.

#### 11.10.6.2.1 Synopsis

CTO( string, objectId, objectRef )

#### 11.10.6.2.2 Arguments

Arguments for CastToObjectRef are shown in table 28.

**Table 28: CastToObjectRef parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericOctetString | string | String to be cast to group identifier. |
| input | GenericInteger | objectId | Integer to be cast to object number. |
| output | GenericObjectReference (Shall provide an IndirectReference to an ObjectRefVariable) | objectRef | |

#### 11.10.6.2.3 Description

Casts the combination of the OctetString string and the Integer objectId to the ObjectReference variable objectRef.

This ResidentProgram can only yield the long form of object reference (where the group identifier is explicit).

## 11.10.7 String manipulation functions

These resident programs are for the manipulation of OctetStrings, and octet values in the range 0x00 to 0xFF are valid and particular values have no special meaning. In particular the value zero does not indicate a string termination. Furthermore, neither mark-up codes nor multi-byte UTF-8 characters are given special treatment; they are just considered as octets within the string.

### 11.10.7.1 Range of string index values

The first octet of any string is referenced by an index of 1.

For the input indices beginExtract, endExtract and startIndex the following bounds checking shall be observed:

- if the index is less than 1 it shall be treated as 1;
- if the index is greater than the string length then it shall be treated as the string length.

### 11.10.7.2 GetStringLength

Returns the number of octets within the string.

#### 11.10.7.2.1 Synopsis

GSL( string, length )

#### 11.10.7.2.2 Arguments

Arguments for GetStringLength are shown in table 29.

**Table 29: GetStringLength parameters**

| In/out / in-out | Type | Name | Example | Comment |
|---|---|---|---|---|
| input | GenericOctetString | string | "Foo##Bar" | |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | length | 8 | |

#### 11.10.7.2.3 Description

Returns in the output *length* the number of octets within the input *string*.

### 11.10.7.3 GetSubString

Extracts a sub-string from a string

#### 11.10.7.3.1 Synopsis

GSS( string, beginExtract, endExtract, stringResult )

#### 11.10.7.3.2 Arguments

Arguments for GetSubString are shown in table 30.

**Table 30: GetSubString parameters**

| In/out / in-out | Type | Name | Example | Comment |
|---|---|---|---|---|
| input | GenericOctetString | string | "Foo##Bar" | |
| input | GenericInteger | beginExtract | 2 | Index of the first octet to be extracted. |
| input | GenericInteger | endExtract | 4 | Index of the last octet of the string to be extracted. |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | stringResult | "oo#" | |

#### 11.10.7.3.3 Description

Extracts the part of *string* from the octet specified by *beginExtract* up to, and including, the octet specified by *endExtract*.

### 11.10.7.4 SearchSubString

Searches for a sub-string within a string

#### 11.10.7.4.1 Synopsis

SSS( string, startIndex, searchString, stringPosition )

#### 11.10.7.4.2 Arguments

Arguments for SearchSubString are shown in table 31.

**Table 31: SearchSubString parameters**

| In/out / in-out | Type | Name | Example | Comment |
|---|---|---|---|---|
| input | GenericOctetString | string | "Foo##Bar" | |
| input | GenericInteger | startIndex | 1 | Index of the first octet to be considered in the search. |
| input | GenericOctetString | searchString | "##" | The target string. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | stringPosition | 4 | The index of the target within *string* or -1 if not found. |

#### 11.10.7.4.3 Description

Searches for a sub-string within a string from a specified starting index. The string is specified by *string*. The octet to start the search from is specified by *startIndex* and the target sub-string is specified by *searchString*. *stringPosition* returns the index within the string of the first octet of the target sub-string, -1 is returned if the string is not found or if *string* is empty. If *searchString* is empty, the search succeeds at *startIndex*. If both *string* and *searchString* are empty, -1 is returned.

### 11.10.7.5 SearchAndExtractSubString

Searches and extracts a sub-string within a string.

#### 11.10.7.5.1 Synopsis

SES( string, startIndex, searchString, stringResult, stringPosition )

#### 11.10.7.5.2 Arguments

Arguments for SearchAndExtractSubString are shown in table 32.

**Table 32: SearchAndExtractSubString parameters**

| In/out / in-out | Type | Name | Example | Comment |
|---|---|---|---|---|
| input | GenericOctetString | string | "Foo##Bar" | |
| input | GenericInteger | startIndex | 1 | Index of the first octet to be considered in the search. |
| input | GenericOctetString | searchString | "##" | The target string. |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | stringResult | "Foo" | The string between the startIndex and the target. An empty string is returned if the target is not found. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | stringPosition | 6 | The index just after the target or -1 if not found. |

#### 11.10.7.5.3 Description

Searches and extracts a sub-string within a string from a specified starting index. The string is specified by *string*. The octet to start from is specified by *startIndex* and the target sub-string is specified by *searchString*.

*stringPosition* returns the index of the octet **immediately after** the *searchString* within the string or -1 if the string is not found or if *string* is empty. If *searchString* is empty, the search succeeds at *startIndex*. If both *string* and *searchString* are empty, -1 is returned. The first index of the string is 1.

*stringPosition* may, if the sub-string is the last element within *string*, have an index beyond its end. *stringResult* returns the string from *startIndex* up to, but not including, *searchString*. For example, if "Bar" is sought in "Foo##Bar" then the returned values are "Foo##" and 9.

## 11.10.8 Service selection

### 11.10.8.1 SI_GetServiceIndex

Returns an index providing an engine specific reference to a service.

#### 11.10.8.1.1 Synopsis

GSI( serviceReference, serviceIndex )

#### 11.10.8.1.2 Arguments

Arguments for SI_GetServiceIndex are shown in table 33.

**Table 33: SI_GetServiceIndex parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericOctetString | serviceReference | serviceReference defines a Stream using one of the locator formats defined in clause 16. |
| output | GenericInteger<br>(Shall provide an IndirectReference to an IntegerVariable) | serviceIndex | This variable returns an integer greater than or equal to zero referencing a service which can be used as the input to SI_TuneIndex.<br><br>-1 is returned if the service is not available. |

#### 11.10.8.1.3 Description

The resident program returns the index of the Service in the Service list described by ServiceReference. ServiceReference is the string used to define a Service in a URL format. This resident program shall accept both explicit (see clause 16.3.3) and inheritance (see table 119) formats.

Service availability is determined by the description of currently running services provided by the SI in the TS currently selected at the time the resident program is called. Services with a running status of "running" or "undefined" shall be considered available. However, this doesn't guarantee that the service is running!

### 11.10.8.2 SI_TuneIndex

Tunes to the given service. **This shall not be invoked with Fork.**

#### 11.10.8.2.1 Synopsis

TIn( serviceIndex )

#### 11.10.8.2.2 Arguments

Arguments for SI_TuneIndex are shown in table 34.

**Table 34: SI_TuneIndex parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | serviceIndex | This integer describes the service to which the receiver shall attempt to tune. |

#### 11.10.8.2.3 Description

The receiver attempts to tune to the specified service. Calls to this resident program may or may not return before the tune has completed in an implementation dependent way. The behaviour is undefined if an invalid serviceIndex is used.

### 11.10.8.3 SI_GetBasicSI

Returns basic SI information about the service identified in the input serviceReference.

#### 11.10.8.3.1 Synopsis

BSI( serviceIndex, networkId, origNetworkId, transportStreamId, serviceId )

#### 11.10.8.3.2 Arguments

Arguments for SI_GetBasicSI are shown in table 35.

**Table 35: SI_GetBasicSI parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | serviceIndex | This integer is a receiver specific identifier for the service about which basic SI is required (see clause 11.10.8.1). |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | networkId | Returns the appropriate DVB SI value. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | origNetworkId | |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | transportStreamId | |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | serviceId | |

#### 11.10.8.3.3 Description

The resident program returns a series of integers representing basic service information (SI) about a service. The service is identified by means of a receiver specific "ServiceIndex". This integer can be determined by means of the SI_GetServiceIndex resident program (see clause 11.10.8.1).The output values are undefined if an invalid serviceIndex is used.

### 11.10.8.4 SI_TuneIndexInfo

Provides a means to define how a receiver shall execute a subsequent application initiated service tune.

#### 11.10.8.4.1 Synopsis

TII( tuneinfo )

#### 11.10.8.4.2 Arguments

Arguments for SI_TuneIndexInfo are shown in table 36.

**Table 36: SI_TuneIndexInfo parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | tuneinfo | The value of the least significant bit shall be interpreted as follows:<br><br>0 = perform any subsequent application initiated service tune normally<br><br>1 = perform any subsequent application initiated service tune quietly<br><br>All other bits are reserved and shall be ignored. |

#### 11.10.8.4.3 Description

The resident program can be used to define how the receiver shall execute any subsequent application initiated service tune. It may be called at any time prior to the call of the SI_TuneIndex resident program. The resident program effectively writes a system variable accessible by the receiver. The scope of this variable is that of the calling application. The default receiver behaviour, until explicitly changed by the application, shall be to tune to a service normally (see below). See clause 11.10.14.

If the lsb of tuneinfo is 0, then the receiver shall execute any subsequent application initiated service tune "normally". This means that the viewer experience shall be exactly as if they had selected the new service themselves using any of the receiver's inherent service navigation mechanisms.

If the lsb of tuneinfo is 1, then the receiver shall execute any subsequent application initiated service tune "quietly". This means that the receiver shall suppress the presentation of all information, e.g. channel number, service name, now/next etc., usually presented during a service tune. The effect should be such that it is not possible to tell the difference between a quiet tune to another service and previewing the components of the same service using an application. In addition the "viewer service context" shall not be changed by a quiet tune, i.e. it shall remain that of the last normal tune whether initiated by an application or by the viewer. This means that a number of quiet tunes may be cascaded without changing the viewer service context. Whatever the actual service currently tuned to, the viewer service context shall be used for all relevant receiver interaction, including (but not restricted to):

- presentation of any front panel channel number;
- presentation of any now/next information;
- the point from which any relative navigation, such as Ch+/-, shall be performed.

The value of tuneinfo shall not affect the receiver's execution of a viewer initiated service tune.

## 11.10.9 Checking references

This set of resident programs can be used by applications to determine if objects are available before embarking on a course of action that requires the objects.

The tests serve two functions, first they confirm that the "file" implied by the reference is available in the file system, secondly the test confirms that, where practical, the file has been brought into the receiver's memory.

For the purposes of these Resident Programs the minimum condition under which a "file" may be considered as available is when the IOR of the corresponding File object has been extracted from the relevant parent Directory (or ServiceGateway) object.

An authoring example is provided under clause 17.9, which illustrates the expected use of these resident programs.

#### 11.10.9.1 CheckContentRef

Allows an application to check if an item of content is available.

##### 11.10.9.1.1 Synopsis

CCR( ref-to-check, ref-valid-var, ref-checked-var )

##### 11.10.9.1.2 Arguments

Arguments for CheckContentRef are shown in table 37.

**Table 37: CheckContentRef parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericContentReference | ref-to-check | This input parameter specifies the target object whose availability is to be checked. |
| output | GenericBoolean (Shall provide an IndirectReference to an BooleanVariable) | ref-valid-var | This output parameter signals whether the target object is "available". |
| output | GenericContentReference (Shall provide an IndirectReference to a ContentRefVariable) | ref-checked-var | This output parameter delivers the content reference input to the resident program when it was invoked. |

##### 11.10.9.1.3 Description

The intended use of CheckContentRef is "non blocking", i.e. it will normally be invoked using the Fork action. This allows the scene to continue operating while the check is performed.

The ref-valid-var result is true if the referenced object exists in the file system, and if receiver resources permit, has been retrieved from the file system to receiver memory. The engine is **not** required to start decoding the content.

The application author is responsible for ensuring the type of the file. For example, if the referenced object is a scene or application object then the resident program will still return true if the file is found to be available.

#### 11.10.9.2 CheckGroupIDRef

Allows an application to check if an application or scene object is available.

##### 11.10.9.2.1 Synopsis

CGR( ref-to-check, ref-valid-var, ref-checked-var )

##### 11.10.9.2.2 Arguments

Arguments for CheckGroupIDRef are shown in table 38.

**Table 38: CheckGroupIDRef parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericObjectReference (The object number shall be 0, i.e. the object shall be an application or a scene) | ref-to-check | This input parameter specifies the target object whose availability is to be checked. |
| output | GenericBoolean (Shall provide an IndirectReference to a BooleanVariable) | ref-valid-var | This output parameter signals whether the target object is "available". |
| output | GenericObjectReference (Shall provide an IndirectReference to an ObjectRefVariable) | ref-checked-var | This output parameter delivers the object reference input to the resident program when it was invoked. |

#### 11.10.9.2.3 Description

The intended use of CheckGroupIDRef is "non blocking", i.e. it will normally be invoked using the Fork action. This allows the scene to continue operating while the check is performed.

The ref-valid-var result is true if the referenced object exists in the file system, and if receiver resources permit, has been retrieved from the file system to receiver memory. The engine is **not** required to start decoding the content.

The application author is responsible for ensuring the type of the file. For example, if the referenced object is a PNG graphics file the resident program will still return true if the file is found to be available.

### 11.10.9.3 GetBootInfo

Allows an application to access the value of the NB_info field at any time.

#### 11.10.9.3.1 Synopsis

GBI( infoResult, bootInfo )

#### 11.10.9.3.2 Arguments

Arguments for GetBootInfo are shown in table 39.

**Table 39: GetBootInfo parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| Output | GenericBoolean (Shall provide an IndirectReference to a Boolean Variable) | infoResult | Set to "true" if a network_boot_info sub-descriptor has been received and found; otherwise "false" is returned. |
| Output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | bootInfo | Return value is NB_Info field if found. An empty string is returned if infoResult is "false". |

#### 11.10.9.3.3 Description

The resident program returns a GenericBoolean in the infoResult output variable indicating whether or not a network_boot_info sub-descriptor has been found and received; if it has, the bootInfo output variable shall be set to the value of the NB_info field of the network_boot_info sub-descriptor (see clause 9.3.2.1).

## 11.10.10 Presentation information

### 11.10.10.1 VideoToGraphics

Transforms a point in the logical 720 x 576 video co-ordinate space to the corresponding point in the co-ordinate system of the current Scene.

#### 11.10.10.1.1 Synopsis

VTG( videoX, videoY, graphicsX, graphicsY )

#### 11.10.10.1.2 Arguments

Arguments for VideoToGraphics are shown in table 40.

**Table 40: VideoToGraphics parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | videoX | Specifies the input point for the transformation. The point (in the MPEG coded frame) is specified in a logical 720 x 576 co-ordinate space (see also clause 14.4.4.2). |
| input | GenericInteger | videoY | |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | graphicsX | Returns a point where an object could be positioned so as to appear on top of the specified input point in the video frame, taking into account any Decoder Format Conversion currently being applied (see also clause 14.5). |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | graphicsY | |

NOTE: If no video is being presented, the return values of this resident program are undefined.

#### 11.10.10.1.3 Informative note

An implementation might perform this calculation in matrix form as follows:

$$\begin{bmatrix} graphicsX \\ graphicsY \\ 1 \end{bmatrix} = \begin{bmatrix} scale & 0 & (posX + offsetX) \\ 0 & scale & (posY + offsetY) \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} DecFC \end{bmatrix} \times \begin{bmatrix} videoX \\ videoY \\ 1 \end{bmatrix}$$

where:

- *videoX* and *videoY* are the input co-ordinates in the 720 x 576 video co-ordinate system;
- *DecFC* is a matrix describing the current Decoder Format Conversion;
- *scale* is the current Video scale factor (0.5, 1.0 or 2.0);
- *posX* and *posY* are the current video position from the Position attribute of the MHEG-5 Video object;
- *offsetX* and *offsetY* are the current values of the VideoDecodeOffset attribute of the MHEG-5 Video object;
- *graphicsX* and *graphicsY* are the output co-ordinates in the 720 x 576 Scene co-ordinate system.

A few common DecFCs might be:

$$DecFC_{none} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad DecFC_{CCO} = \begin{bmatrix} 16/12 & 0 & -120 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$DecFC_{pillarbox} = \begin{bmatrix} 12/16 & 0 & 90 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad DecFC_{letterbox} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 12/16 & 72 \\ 0 & 0 & 1 \end{bmatrix}$$

$$DecFC_{14by9lettterbox} = \begin{bmatrix} 16/14 & 0 & -720/14 \\ 0 & 12/14 & 576/14 \\ 0 & 0 & 1 \end{bmatrix}$$

Of course, the resident program must in practice use the true transformation in use rather than any theoretical one if the two differ.

Examples are given in table 41.

**Table 41: VideoToGraphics parameters**

| Scenario | Input points | Output points |
|---|---|---|
| Broadcast is a 16:9 coded frame of size 720 x 576.<br>The display is 16:9 and no DecFC is being applied. | (40,60)<br>(-500,-500) | (40,60)<br>(-500,-500) |
| Broadcast is a 16:9 coded frame of size 720 x 576.<br>The display is 4:3 and the DecFC is centre-cut-out. | (0,0)<br>(360,100) | (-120,0)<br>(360,100) |
| Broadcast is a 16:9 coded frame of size 352 x 288.<br>The display is 4:3 and the video is being shown in a 16:9 letterbox. | (0,0)<br>(720,576) | (0,72)<br>(720,504) |
| Broadcast is a 4:3 coded frame of size 720 x 576.<br>The display is 4:3 and the video has been scaled to 360 x 288 and positioned at (200,200). | (0,0)<br>(720,576) | (200,200)<br>(560,488) |

### 11.10.10.2 SetWidescreenAlignment

Sets the current mode for aligned presentation of graphics and 16:9 video for 4:3 displays.

#### 11.10.10.2.1 Synopsis

SWA( mode )

#### 11.10.10.2.2 Arguments

Arguments for SetWidescreenAlignment are shown in table 42.

**Table 42: SetWidescreenAlignment parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | mode | Specifies a new WidescreenAlignment Mode:<br>1 = Centre-Cut-Out<br>2 = Letterbox |

#### 11.10.10.2.3 Description

This resident program sets the Widescreen Alignment Mode for forced alignment of 16:9 video and 4:3 graphics (typically to support 4:3 displays). The value is only relevant when the active Scene has an explicit 4:3 AspectRatio attribute defined and the video is broadcast with a 16:9 coded frame. See clause 14.5.

The default mode is "1" (Centre-Cut-Out). See clause 11.10.14.

### 11.10.10.3 GetDisplayAspectRatio

Returns the aspect ratio of the display.

#### 11.10.10.3.1 Synopsis

GDA( aspectratio )

#### 11.10.10.3.2 Arguments

Arguments for GetDisplayAspectRatio are shown in table 43.

**Table 43: GetDisplayAspectRatio parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| output | GenericInteger<br>(Shall provide an IndirectReference to an IntegerVariable) | aspectratio | Indicates the display:<br>1 = 4:3<br>2 = 16:9 |

#### 11.10.10.4 SetSubtitleMode

Enables or disables subtitle presentation simultaneous to MHEG-5 presentation.

##### 11.10.10.4.1 Synopsis

SSM( on )

##### 11.10.10.4.2 Arguments

Arguments for SetSubtitleMode are shown in table 44.

**Table 44: SetSubtitleMode parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericBoolean | on | Enables or disables subtitle presentation on receivers that support simultaneous display of subtitles and MHEG-5 applications.<br><br>True = enable. |

##### 11.10.10.4.3 Description

If a receiver is able to simultaneously display subtitles and MHEG-5 applications then this resident program provides a means for the application author to explicitly disable or enable subtitle presentation. See clause 11.10.14 and clause 14.3.3.

On receivers which do not display subtitles and MHEG-5 applications simultaneously this resident program shall have no effect.

The default mode is enabled.

### 11.10.11 Conditional access

#### 11.10.11.1 CI_SendMessage

Sends a message via an open DVB CI Application MMI session related to the current application. **This shall not be invoked with Fork.**

##### 11.10.11.1.1 Synopsis

CIS( Message, Response )

##### 11.10.11.1.2 Arguments

Arguments for CI_SendMessage are shown in table 45.

**Table 45: CI_SendMessage parameters**

| In/out / in-out | Type | Name |
|---|---|---|
| Input | GenericOctetString | Message |
| Output | GenericOctetString<br>(Shall provide an IndirectReference to an OctetStringVariable) | Response |

##### 11.10.11.1.3 Description

Sends the OctetString bytes in Message to the open DVB CI Application MMI session related to the current application. The bytes are sent using the FileRequest message (see clause 14.9.3.4). The FileRequest RequestType will be set to "data". On receiving the FileAcknowledge message the resident program returns the DataByte field of the FileAcknowledge message in the OctetString Response (see clause 14.9.3.5).

If no DVB CI Application MMI session exists then the resident program shall have no effect.

## 11.10.12 Developer utilities

### 11.10.12.1 WhoAmI

Returns all engine identification strings recognized by the engine.

#### 11.10.12.1.1 Synopsis

WAI( ident )

#### 11.10.12.1.2 Arguments

Arguments for WhoAmI are shown in table 46.

**Table 46: WhoAmI parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | ident | This string contains a list of all identification strings recognized by the receiver, separated by spaces. |

#### 11.10.12.1.3 Description

This resident program returns a list of the engine identification strings recognized by the receiver and which would generate a "true" response in the UniversalEngineProfile(N) GetEngineSupport request (see clause 11.4.1). For example, a receiver might return the string "FEG001103 MHGFEG056 DSMFEG017".

## 11.10.13 Data exchange with ResidentPrograms

This clause is intended to clarify the behaviour of information passed between MHEG-5 applications and resident programs.

### 11.10.13.1 Memory spaces

There are two distinct memory spaces to consider:

- MHEG-5 application memory; and
- procedural code memory.

### 11.10.13.2 On invocation

When a ResidentProgram is invoked (with Call or Fork) the behaviour is as if a snap shot of the input and in-out parameters were passed from the application memory space to the memory space of the procedural code.

### 11.10.13.3 CallSucceeded/ForkSucceeded Values

In clause 14.4 of the ISO/IEC 13522-5 [14] it states that the value of the BooleanVariable CallSucceeded / ForkSucceeded is set to false if the Program "finishes abnormally".

CallSucceeded and ForkSucceeded shall return false only if the ResidentProgram could not be called, for example if the parameters were of the incorrect type or the Program did not exist. In all other cases, the call is deemed to have succeeded even if, for example, the input parameters are formatted with incorrect values such as null strings or negative integers.

### 11.10.13.4 During execution

While the procedural code executes there is no connection between its memory space and that of the MHEG-5 application.

In principle the MHEG-5 application could modify variables passed by reference to the procedural code with no effect on the procedural code's version. However, this is probably not a useful thing to do.

#### 11.10.13.5 On completion

If the procedural code completes normally then its results (any in-out, output parameters and the succeeded parameter) are transferred back to the MHEG-5 application memory. The timing here is significant, from the MHEG-5 application's point of view all of the results of the ResidentProgram are delivered atomically between the processing of asynchronous events.

NOTE: The processing of an asynchronous event includes the processing of all consequent synchronous events. The behaviour is as if an Action object with a series of SetVariable actions is performed.

### 11.10.14 Duration of effect of ResidentPrograms

Certain ResidentPrograms affect the state of the receiver. In all cases the scope of such state changes is that of the running MHEG-5 Application. Default state shall be restored whenever an Application terminates or a new Application starts (see clause 8.1.1).

## 11.11 Limitations on standard data-types

### 11.11.1 BooleanVariable

The BooleanVariable size is undefined as the implementation of its representation is not significant provided that all possible Boolean values can be represented. However, when modelling storage (e.g. when written to persistent storage) Booleans occupy 1 byte. See table 47.

**Table 47: BooleanVariable values**

| Type of value | Size | Min. value | Max. value |
|---|---|---|---|
| true or false | Unspecified | n/a | n/a |

### 11.11.2 IntegerVariable

See table 48.

**Table 48: IntegerVariable values**

| Type of value | Size | Min. value | Max. value |
|---|---|---|---|
| signed integer | 32 bits | -2 147 483 648 | 2 147 483 647 |

### 11.11.3 OctetString

The OctetString data type is restricted to be not longer than 2 147 483 647 octets long, thus the only practical restriction is that of available memory (see clause 14.7).

A null OctetString value shall be encoded as "".

### 11.11.4 ObjectNumber

The ObjectNumber data type is restricted as shown in table 49.

**Table 49: ObjectNumber values**

| Type of value | Size | Min. value | Max. value |
|---|---|---|---|
| signed integer | 32 bits | 0 (NOTE) | 2 147 483 647 |
| NOTE : 0 for Group objects and 1 for Ingredient objects. | | | |

### 11.11.5 GroupIdentifier and ContentReference

GroupIdentifiers shall not be more than 64 bytes long. This limit applies to the fully resolved form of the identifier (for example "DSM://weather/today/cloud") rather than the abbreviated form of the identifier (such as "/cloud") that can exist in the exchanged representation of the MHEG-5 application.

See clause 16.3.2.

## 11.12 Extensions to the MHEG-5 language specification

### 11.12.1 Preamble

This Profile specifies a number of changes to the MHEG-5 language that must be supported by receivers. The changes have been made to improve the performance of the language, whilst maintaining compatibility with the original specification.

Where there is a definition of ASN.1 notation, this shall be used in conjunction with Annex A of the language specification.

### 11.12.2 Changes to the Group class

Timer functionality has been moved to the Group class from the Scene class.

#### 11.12.2.1 Changes to "Own internal attributes"

The following internal attribute is added:

| | |
|---|---|
| Timers | (As defined in the Scene class of ISO/IEC 13522-5 [14].) All references in the text to Scene shall be replaced by Group. |

#### 11.12.2.2 Changes to "Events"

The following event is added:

| | |
|---|---|
| TimerFired | This event is generated when a timer has fired.<br>• Associated data: TimerIdentifier - Integer. |

#### 11.12.2.3 Changes to "Effect of MHEG-5 actions"

The following action is added:

| | |
|---|---|
| SetTimer(TimerId, TimerValue, AbsoluteTime) | The SetTimer action (as defined in the Scene class of ISO/IEC 13522-5 [14]) is added. All references in the text to Scene shall be replaced by Group. |

### 11.12.3 Changes to the Application class

#### 11.12.3.1 Changes to "Own exchanged attributes"

The following changes are made:

| | |
|---|---|
| BackgroundColour | The BackgroundColour attribute is renamed to OriginalBackgroundColour. |
| TextColour | The TextColour attribute is renamed to OriginalTextColour. |
| FontAttributes | The FontAttributes attribute is renamed to OriginalFontAttributes. |

NOTE: The above changes have no impact on the textual notation and the ASN.1 tags.

## 11.12.4 Changes to the Scene class

To support the concept of a Scene with no aspect ratio, the AspectRatio attribute is made optional.

Timer functionality has been moved to the Group class.

### 11.12.4.1 Changes to "Own exchanged attributes"

The following exchanged attribute is changed:

AspectRatio — Original aspect ratio of the Scene. This attribute is expressed by a width / height ratio.

- Optional rational number.
- If no AspectRatio is specified, the Scene has no aspect ratio.

**ASN.1 form:**

```
A.4 Scene Class

SceneClass ::= SET
{
   COMPONENTS OF GroupClass,
   input-event-register [51] INTEGER,
   scene-coordinate-system [52] SceneCoordinateSystem,
   aspect-ratio [53] AspectRatio OPTIONAL,
   moving-cursor [54] BOOLEAN DEFAULT FALSE,
   next-scenes [55] SEQUENCE SIZE (1..MAX) OF NextScene OPTIONAL
}
```

### 11.12.4.2 Changes to "Own internal attributes"

The following internal attribute is removed:

Timers — The internal attribute Timers is removed from the Scene class.

### 11.12.4.3 Changes to "Events"

The TimerFired event is removed.

### 11.12.4.4 Changes to "Effect of MHEG-5 actions"

The following action is removed:

SetTimer(TimerId, TimerValue, AbsoluteTime) — The SetTimer action is removed from the Scene class.

The following action is added:

SetInputRegister(NewInputRegister) Change the InputEventRegister attribute of the target Scene object. Changing the register will affect how subsequent key presses are handled by the Scene. Note that key events generated before the ElementaryAction are unchanged.

Provision of use:

- The target object shall be an available Scene object.

**ASN.1 form:**

```
set-input-register [239] SetInputRegister

SetInputRegister ::= SEQUENCE
{
      target GenericObjectReference,
      new-input-register GenericInteger
}
```

**Text form:**

```
":SetInputReg" "(" Target GenericInteger ")" .
```

## 11.12.5 Changes to the TokenGroup class

### 11.12.5.1 Changes to "Effect of MHEG-5 actions"

The following action is changed:

CallActionSlot(Index) The second provision of use is changed to read:

"Index shall be set in the range [1, number of ActionSlots associated with the item that currently has the token]."

## 11.12.6 Changes to the ListGroup class

### 11.12.6.1 Changes to "Own exchanged attributes"

The ListGroup class exchanged attribute Positions is renamed to OriginalPositions. The ASN.1 value and text representation remains the same.

### 11.12.6.2 Changes to "Own internal attributes"

The ListGroup class is extended with a new internal attribute Positions. The default value for this is that of attribute OriginalPositions.

#### 11.12.6.3 Changes to "Effect of MHEG-5 actions"

The following action is added:

SetCellPosition(CellIndex, NewXPosition, NewYPosition)

Change the display position of a ListGroup object display cell. A cell is identified by its (one based) index. The Positions attribute for this cell is changed and the object is redrawn. If the CellIndex specifies an index smaller than or equal to 1 then the position of the first cell is changed. If the CellIndex specifies an index greater than or equal to the number of cells then the position of the last cell is changed.

Provision of use:

- The target object shall be an available ListGroup object.

**ASN.1 form:**

```
set-cell-position [238] SetCellPosition

SetCellPosition ::= SEQUENCE
{
      target GenericObjectReference,
      index GenericInteger,
      new-x-position GenericInteger,
      new-y-position GenericInteger
}
```

**Text form:**

```
":SetCellPosition" "(" Target Index XPosition YPosition ")" .
```

### 11.12.7 Changes to the Bitmap class

#### 11.12.7.1 Changes to "Own internal attributes"

The following internal attribute is added:

BitmapDecodeOffset

Position of the top left corner of the decoded and scaled bitmap with respect to the top left corner of the Bitmap object.

- Pair of Integers (XOffset, YOffset)
- Initial value: (0,0).

### 11.12.7.2 Changes to "Effect of MHEG-5 actions"

The following actions are added:

SetBitmapDecodeOffset( NewXOffset, NewYOffset)

Change the location of the decoded and scaled bitmap with respect to the target Bitmap object. The offset parameters may be negative.

Execute the following sequence of actions:

1) Set the BitmapDecodeOffset attribute according to NewXOffset and NewYOffset.

2) If the Bitmap object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the BoxSize and Position attributes and according to its position in the DisplayStack of the active Application object.

Provisions of use:

- The Target object shall be an available Bitmap object.
- NewXOffset and NewYOffset shall correspond to an offset interpreted in the Scene coordinate system defined by the SceneCoordinateSystem attribute of the currently active Scene.

**ASN.1 form:**

```
set-bitmap-decode-offset [246] SetBitmapDecodeOffset
SetBitmapDecodeOffset ::= SEQUENCE
{
   target GenericObjectReference,
   new-x-offset GenericInteger,
   new-y-offset GenericInteger
}
```

**Text form:**

```
":SetBitmapDecodeOffset" "(" Target NewXOffset NewYOffset ")" .
NewXOffset::= GenericInteger .
NewYOffset::= GenericInteger .
```

**Examples:**

See examples under SetVideoDecodeOffset in clause 11.12.10.

GetBitmapDecodeOffset( XOffsetVar, YOffsetVar)

Return the location of the decoded and scaled bitmap with respect to the target Bitmap object. The offset values may be negative.

Set the Variables referenced by XOffsetVar and YOffsetVar to the value of the X and Y decode offset of the target Bitmap object respectively.

Provisions of use:

- The Target object shall be an available Bitmap object.
- XOffsetVar and YOffsetVar shall refer to active IntegerVariable objects.

**ASN.1 form:**

```
get-bitmap-decode-offset [247] GetBitmapDecodeOffset
GetBitmapDecodeOffset ::= SEQUENCE
{
   target GenericObjectReference,
   x-offset-var ObjectReference,
   y-offset-var ObjectReference
}
```

**Text form:**

```
":GetBitmapDecodeOffset" "(" Target XOffsetVar YOffsetVar ")" .
XOffsetVar::= ObjectReference .
YOffsetVar::= ObjectReference .
```

## 11.12.8 Changes to the Text class

### 11.12.8.1 Changes to "Own exchanged attributes"

The following changes are made:

| | |
|---|---|
| FontAttributes | The FontAttributes attribute is renamed to OriginalFontAttributes. |
| | Change the first sentence of the description to: |
| | "This attribute is used to set initial specific Font attributes such as style, character size, text colour and background colour". |
| | Change all references to FontAttributes in the description to OriginalFontAttributes. |
| TextColour | The TextColour attribute is renamed to OriginalTextColour. |
| | Change the first sentence of the description to: |
| | "Indicate which colour should initially be used…" |
| BackgroundColour | The BackgroundColour attribute is renamed to OriginalBackgroundColour. |
| | Change the first sentence of the description to: |
| | "Indicate which colour should initially be used…" |

NOTE: The above changes have no impact on the textual notation and the ASN.1 tags.

### 11.12.8.2 Changes to "Own internal attributes"

The following internal attributes are added:

TextColour

Colour to use for the text characters when representing the Text object.

This attribute is interpreted as a zero-based index in the look-up table defined in the PaletteRef attribute or as a direct colour value, depending on the attribute type.

- Integer or OctetString. An Integer will be interpreted as an index in a Palette; an OctetString will be interpreted as a direct colour value.
- Initial value: OriginalTextColour.

BackgroundColour

Colour to use for the background area when representing the Text object.

This attribute is interpreted as a zero-based index in the look-up table defined in the PaletteRef attribute or as a direct colour value, depending on the attribute type.

- Integer or OctetString. An Integer will be interpreted as an index in a Palette; an OctetString will be interpreted as a direct colour value.
- Initial value: OriginalBackgroundColour.

FontAttributes

This attribute is used to set the specific Font attributes such as style, character size, text colour and background colour.

The exact encoding format of the FontAttribute attribute is related to the value of the type of Font object mentioned by the Font attribute.

- OctetString.
- Initial value: OriginalFontAttributes.

### 11.12.8.3 Changes to "Effect of MHEG-5 actions"

The following actions are added:

SetBackgroundColour(NewBackgroundColour)

Change the BackgroundColour attribute of the target Text (or derived class) object to NewBackgroundColour. The object is redrawn.

Provision of use:

- The target object shall be an available Text object.

**ASN.1 form:**

```
set-background-colour [237] SetBackgroundColour

SetBackgroundColour ::= SEQUENCE
{
      target GenericObjectReference,
      new-background-colour NewColour
}
```

**Text form:**

```
":SetBackgroundColour" "(" Target NewColour ")" .
```

SetTextColour(NewTextColour)

Change the TextColour attribute of the target Text (or derived class) object to NewTextColour. The object is redrawn.

Provision of use:

- The target object shall be an available Text object.

**ASN.1 form:**

```
set-text-colour [240] SetTextColour

SetTextColour ::= SEQUENCE
{
      target GenericObjectReference,
      new-text-colour NewColour
}
```

**Text form:**

```
":SetTextColour" "(" Target NewColour ")" .
```

SetFontAttributes(NewFontAttributes)

Change the FontAttributes attribute of the target Text (or derived class) object to NewFontAttributes. The object is redrawn.

Provision of use:

- The target object shall be an available Text object.

**ASN.1 form:**

```
set-font-attributes [241] SetFontAttributes

SetFontAttributes ::= SEQUENCE
{
      target GenericObjectReference,
      new-font-attribute GenericOctetString
}
```

**Text form:**

```
":SetFontAttributes" "(" Target GenericOctetString ")" .
```

## 11.12.9 Changes to the Stream class

To support Stream objects as a source of events without Video, Audio or RTGraphics objects the multiplex component is made optional.

#### 11.12.9.1 Changes to "Own exchanged attributes"

The following exchanged attribute is changed:

Multiplex

Change the bulleted list to read:

- Optional attribute.
- Sequence of inclusions of Video, Audio and RT-Graphics objects…

**ASN.1 form:**

```
A.30 Stream Class

StreamClass ::= SET
{
      COMPONENTS OF PresentableClass
       (WITH COMPONENTS {…, original-content PRESENT}),
      multiplex [92] SEQUENCE SIZE(1..MAX) OF StreamComponent OPTIONAL,
      storage [93] Storage DEFAULT stream,
      looping [94] INTEGER {infinity(0)} DEFAULT 1
}
```

**Text form:**

```
StreamClass ::= "{:Stream" Presentable [Multiplex] [Storage] [Looping]
"}" .
```

### 11.12.10 Changes to the Video class

#### 11.12.10.1 Changes to "Own internal attributes"

The following internal attribute is added:

VideoDecodeOffset

Position of the top left corner of the decoded and scaled video with respect to the top left corner of the Video object.

- Pair of Integers (XOffset, YOffset).
- Initial value: (0,0).

### 11.12.10.2 Changes to "Effect of MHEG-5 actions"

The following actions are added:

SetVideoDecodeOffset(NewXOffset, NewYOffset)

Change the location of the decoded and scaled video with respect to the target Video object. The offset parameters may be negative.

Execute the following sequence of actions:

1) Set the VideoDecodeOffset attribute according to NewXOffset and NewYOffset.

2) If the Video object is active, redraw the graphic widget representing the object on the screen in the bounding box defined by the BoxSize and Position attributes and according to its position in the DisplayStack of the active Application object.

Provisions of use:

- The Target object shall be an available Video object.
- NewXOffset and NewYOffset shall correspond to an offset interpreted in the Scene coordinate system defined by the SceneCoordinateSystem attribute of the currently active Scene.

**ASN.1 form:**

```
set-video-decode-offset [242] SetVideoDecodeOffset
SetVideoDecodeOffset ::= SEQUENCE
{
   target GenericObjectReference,
   new-x-offset GenericInteger,
   new-y-offset GenericInteger
}
```

**Text form:**

```
":SetVideoDecodeOffset" "(" Target NewXOffset NewYOffset ")" .
NewXOffset::= GenericInteger .
NewYOffset::= GenericInteger .
```

**Example 1**

Figure 4 illustrates quarter-screen video with masking to prevent display of pixels normally hidden at fullscreen size due to overscan.

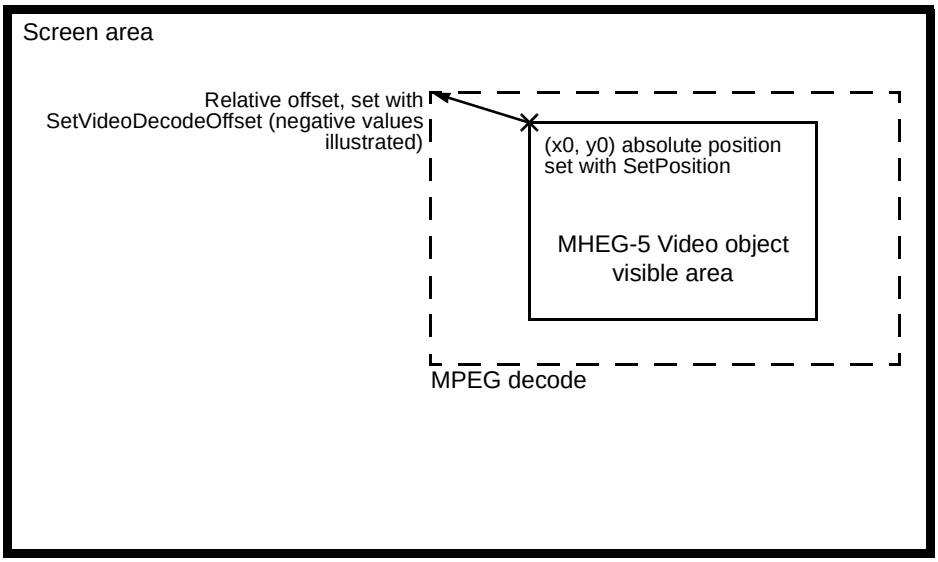



**Figure 4**

Example code:

```
:SetBoxSize (10 0 0)
:ScaleVideo (10 360 288)
:SetVideoDecodeOffset (10 8 2)
:SetPosition (10 368 102)
:SetBoxSize (10 344 284)
```

**Example 2**

Figure 5 illustrates full-screen video with masking to display one quarter of the image. This allows a quarterscreen presentation of four separate pictures under application control.

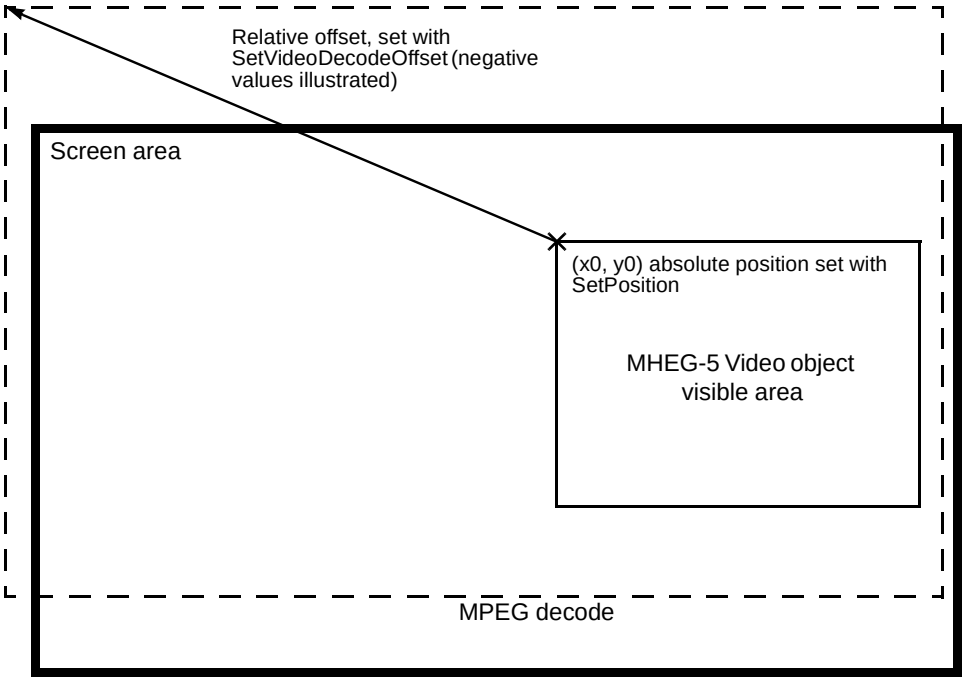



**Figure 5**

Example code:

```
:SetBoxSize (10 344 284)
:SetPosition (10 360 100)
:SetVideoDecodeOffset (10 368 290)
```

**Example 3**

Figure 6 illustrates double size video masked to show a quarter of the video image full-screen.

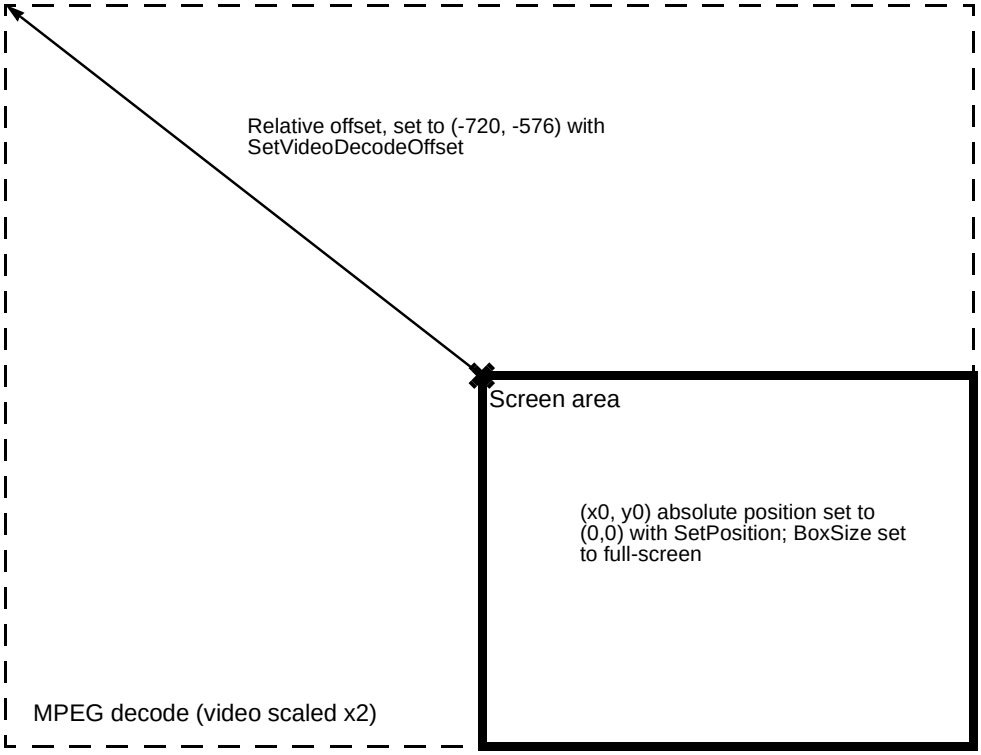



**Figure 6**

Example code:

```
:SetBoxSize (10 0 0)
:ScaleVideo (10 1440 1152)
:SetVideoDecodeOffset (10 720 576)
:SetPosition (10 0 0)
:SetBoxSize (10 720 576)
```

GetVideoDecodeOffset(XOffsetVar, YOffsetVar)

Returns the location of the decoded and scaled video with respect to the target Video object. The offset values may be negative.

Set the Variables referenced by XOffsetVar and YOffsetVar to the value of the X and Y decode offset of the target Video object respectively.

Provisions of use:

- The Target object shall be an available Video object.
- XOffsetVar and YOffsetVar shall refer to active IntegerVariable objects.

**ASN.1 form:**

```
get-video-decode-offset [243] GetVideoDecodeOffset
GetVideoDecodeOffset ::= SEQUENCE
{
   target GenericObjectReference,
   x-offset-var ObjectReference,
   y-offset-var ObjectReference
}
```

**Text form:**

```
":GetVideoDecodeOffset" "(" Target XOffsetVar YOffsetVar ")" .
XOffsetVar::= ObjectReference .
YOffsetVar::= ObjectReference .
```

## 11.12.11 Changes to the Slider class

### 11.12.11.1 Changes to "Own exchanged attributes"

The following changes are made:

| | |
|---|---|
| MinValue | The MinValue exchanged attribute is renamed OriginalMinValue. |
| MaxValue | The MaxValue exchanged attribute is renamed OriginalMaxValue. |
| StepSize | The StepSize exchanged attribute is renamed OriginalStepSize. |

NOTE:The above changes have no impact on the textual notation and the ASN.1 tags.

### 11.12.11.2 Changes to "Own internal attributes"

The following internal attributes are added:

MinValue — Lowest value that the SliderValue attribute may be set to.

- Integer value.
- Initial value: Value of the OriginalMinValue attribute.

MaxValue — Greatest value that the SliderValue attribute may be set to.

- Integer value.
- Initial value: Value of the OriginalMaxValue attribute.

StepSize — The smallest value by which the value of the SliderValue internal attribute may be increased or decreased.

- Integer value.
- Initial value: Value of the OriginalStepSize attribute.

#### 11.12.11.3 Changes to "Events"

The following event is added:

SliderValueChanged

This event is generated when the SliderValue attribute of the Slider changes due to user interaction, or if any of the Step, SetSliderValue, SetPortion or SetSliderParameters actions are invoked.

- Associated data: SliderValueTag - Integer. The value of the associated data shall be set to the SliderValue attribute.
- Event type: Asynchronous.

**ASN.1 form:**

Add the following to the EventType enumeration in clause A.6 before the final closing brace:

```
slider-value-changed(33)
```

**Text form:**

Add the following to the EventTypeEnum in clause B.4.6 before the final full stop:

```
|"SliderValueChanged"
```

#### 11.12.11.4 Changes to "Internal behaviour"

The following internal behaviour is changed:

Interaction

Execute the following sequence of actions:

1) Apply the Interaction behaviour as defined in the Interactible class.
2) Allow the user to interact with the Slider object by moving the marker along the main axis. Exactly how this user interaction takes place is not specified by this part of ISO/IEC 13522-5 [14]. However, the smallest marker displacement shall be proportional to the value of the StepSize attribute. Each time the marker is moved:

   a) set the SliderValue attribute to a value that corresponds to the new marker position, and

   b) generate a SliderValueChanged event.
3) When user interaction has completed, either because the user terminates the interaction or because the application terminates it using the SetInteractionStatus action:

   a) set the InteractionStatus attribute to False, and

   b) generate an InteractionCompleted event.

#### 11.12.11.5 Changes to "Effect of MHEG-5 actions"

The following actions are changed:

Step

Change Point 4 of the sequence of actions to read:

"4. Generate a SliderValueChanged event."

SetSliderValue — Change Point 3 of the sequence of actions to read:

"3. Generate a SliderValueChanged event."

SetPortion — Change Point 3 of the sequence of actions to read:

"3. Generate a SliderValueChanged event."

The following action is added:

SetSliderParameters(NewMinValue, NewMaxValue, NewStepSize)

Change the Slider's lowest and greatest values, and the smallest value by which the value of the SliderValue attribute may be increased or decreased.

Execute the following sequence of actions:

1) Set the MinValue, MaxValue and StepSize attributes according to NewMinValue, NewMaxValue and NewStepSize respectively.
2) Set the SliderValue attribute to MinValue.
3) If the Slider is active, redraw the Slider taking into account the new values, and according to its position in the DisplayStack of the active Application object.
4) Generate a SliderValueChanged event.

Provisions of use:

- The Target shall be an available Slider object.
- NewMinValue, NewMaxValue, NewStepSize shall conform to the following criteria:

  NewMinValue < NewMaxValue

  N × NewStepSize = (NewMaxValue - NewMinValue), where N is some positive integer.

  NewMaxValue - NewMinValue >= Portion

**ASN.1 form:**

```
set-slider-parameters [248] SetSliderParameters
SetSliderParameters ::= SEQUENCE
{
   target GenericObjectReference,
   new-min-value GenericInteger,
   new-max-value GenericInteger,
   new-step-size GenericInteger
}
```

**Text form:**

```
":SetSliderParameters" "(" Target NewMinValue NewMaxValue NewStepSize
")" .
NewMinValue ::= GenericInteger .
NewMaxValue ::= GenericInteger .
NewStepSize ::= GenericInteger .
```

## 11.12.12 Changes to the HyperText class

### 11.12.12.1 Changes to "Own internal attributes"

The following internal attribute is added:

FocusPosition

Index of the currently highlighted anchor where an index of 1 represents the first anchor in the content. A value of 0 means that there is no anchor to highlight. When content with at least 1 anchor is available this attribute will be in the range [1,N] where N is the number of anchors in the content. In all other situations (content is not available or there are no anchors within the content) this attribute will have the value 0.

- Integer.
- Initial value: 0 (no anchor to highlight).

### 11.12.12.2 Changes to "Events"

The following event is added:

FocusMoved

Signals that the value of the FocusPosition attribute has been updated. This event will be generated when either the user moves the anchor highlight, or the SetFocusPosition ElementaryAction is invoked.

- Associated data: the new value of FocusPosition - Integer.
- Event type: Asynchronous.

**ASN.1 form:**

Add the following to the EventType enumeration in clause A.6 before the final closing brace:

```
focus-moved(32)
```

**Text form:**

Add the following to the EventTypeEnum in clause B.4.6 before the final full stop:

```
|"FocusMoved"
```

### 11.12.12.3 Changes to "Internal behaviours"

Interaction step 2

The following text replaces the definition of the HyperText Interaction behaviour Step 2 in ISO/IEC 13522-5 [14]:

"Allow the user to move the focus through the set of anchors in the Hypertext object and to select the focused anchor. Each time the focus moves a FocusMoved event is generated.

Each time an anchor is selected an AnchorFired event is generated."

ContentPreparation

The ContentPreparation internal behaviour semantics have changed from this object's base class as follows:

- Apply Steps 1 to 3 of the ContentPreparation behaviour of the base class *synchronously.*

The following steps are *asynchronous* and occur when the content of the object has been fully retrieved:

- Apply Steps 4 and 5 of the ContentPreparation behaviour of the base class.
- Step 6: if there are no anchors then set the FocusPosition internal attribute to zero otherwise set it to one. If the FocusPosition internal attribute changes then generate a FocusMoved event.
- Step 7: generate a ContentAvailable event.

### 11.12.12.4 Changes to "Effect of MHEG-5 actions"

The following actions are added:

GetFocusPosition(FocusPositionVar)

Set the variable referenced by FocusPositionVar to the value of the FocusPosition attribute.

Provisions of use:

- The target object shall be an available HyperText object.
- FocusPositionVar shall refer to an active IntegerVariable object.

**ASN.1 form:**

```
get-focus-position[244] GetFocusPosition
GetFocusPosition ::= SEQUENCE
{
   target GenericObjectReference,
   focus-position-var ObjectReference
}
```

**Text form:**

```
GetFocusPosition  := ":GetFocusPosition" "(" Target FocusPositionVar ")" .
FocusPositionVar ::= ObjectReference .
```

SetFocusPosition(NewFocusPosition)

Set the FocusPosition attribute to the value of NewFocusPosition. This change is to be reflected in the visual representation of the object. If NewFocusPosition is greater than N, where N is the number of anchors available in the content for the target object then it shall be treated as N.

A FocusMoved event will be generated if the value of the FocusPosition is altered by this elementary action.

Provision of use:

- The target object shall be an available HyperText object.

**ASN.1 form:**

```
set-focus-position[245] SetFocusPosition

SetFocusPosition ::= SEQUENCE
{
   target GenericObjectReference,
   new-focus-position GenericInteger
}
```

**Text form:**

```
SetFocusPosition ::= ":SetFocusPosition" "(" Target NewFocusPosition ")" .
NewFocusPosition ::= GenericInteger .
```

# 11.13 Clarifications, restrictions and amendments

## 11.13.1 Additional semantics for the SetTimer action

If the TimerValue attribute of the SetTimer action is negative then the engine shall ignore the action regardless of the setting of the AbsoluteTime flag.

NOTE: The MHEG-5 corrigenda (ISO/IEC 13522-5:1997/Cor.1:1999(E) [21]) contains the following text which shall be followed:

"*If the time indicated in TimerValue has already passed and the AbsoluteTime is set to True, the TimerFired event shall not be raised.*"

## 11.13.2 CounterPosition attribute

As there is no support for managing NPT there is no mapping between the MHEG-5 internal attribute CounterPosition of the Stream class and NPT. As such, the Stream class CounterPosition attribute will always remain in an undefined state, and any MHEG-5 actions which depend on this attribute (i.e. Stream class SetCounterTrigger, SetCounterPosition and SetCounterEndPosition) shall be ignored.

## 11.13.3 Synchronous event processing

The behaviour of MHEG-5 Engines while processing synchronous events has been the source of some confusion in the past, with many engine implementations differing in how multiple synchronous events are handled in large elementary actions. e.g. Launch, Spawn etc. Two interpretations of ISO/IEC 13522-5 [14] are allowed in this Profile to cater for the ambiguity. Each interpretation is described below as pseudo-code for the engine's event processing loop, and for the "Send Event" primitive as it appears in the Elementary Actions in ISO/IEC 13522-5 [14]. Implementations need not use the example code and data structures below, but shall behave as if they do. Engine implementations may use either interpretation but shall not introduce new interpretations.

#### 11.13.3.1 Preferred interpretation

```
Send Synchronous Event (as a result of Executing ElementaryAction):
   Examine Links.
   Append ElementaryActions to ‘Temporary Action Queue’

Processing Loop:
   FOREACH AsyncEvent DO
      Create ‘Main Action Queue’ from resulting ElementaryActions.
      FOREACH ElementaryAction in ‘Main Action Queue’ DO
         Create ‘Temporary Action Queue’
         Execute ElementaryAction
         Prepend ‘Temporary Action Queue’ to ‘Main Action Queue’
      ENDFOR
   ENDFOR
```

#### 11.13.3.2 Alternative interpretation

```
Send Synchronous Event (as a result of Executing ElementaryAction):
   Append Event to ‘Synchronous Event Queue’

Processing Loop:
   FOREACH Asynchronous Event DO
      Create ‘Main Action Queue’ from resulting ElementaryActions.
      FOREACH ElementaryAction in ‘Main Action Queue’ DO
         Execute ElementaryAction
         Create ‘Temporary Action Queue’
         FOREACH Event in ‘Synchronous Event Queue’ DO
            Examine Links
            Append ElementaryActions to ‘Temporary Action Queue’
         ENDFOR
         Prepend ‘Temporary Action Queue’ to ‘Main Action Queue’
      ENDFOR
   ENDFOR
```

#### 11.13.3.3 Explanation

The two models satisfy the requirements set out in ISO/IEC 13522-5 [14] and differ in exactly when a synchronous event is handled. In the preferred interpretation, synchronous events are handled as they are raised and the resulting Actions are made ready to run. In the alternative interpretation, synchronous events are queued and handled after the ElementaryAction that raised them has completed.

The main difference in observed behaviour is found during large Actions such as Launch. In the alternative interpretation, each "IsRunning" event is handled after the Launch Actions has completed. At this point, all "InitiallyActive" Link objects in the Application are active and will fire if set to source from the "IsRunning" event. If the preferred interpretation is used then the event is handled at the point it is raised, and a Link can only fire if it appears before the source Ingredient in the Items attribute of the Application. See also clause 17.13.1.

### 11.13.4 Actions that generate more than one synchronous event

Some actions lead to the generation of more than one synchronous event. For example, the Move or MoveTo actions on a TokenGroup lead to both TokenMovedFrom and TokenMovedTo events.

MHEG-5 describes the order in which the events are generated (in this case TokenMovedFrom followed by TokenMovedTo).

The effect of each of these events is equivalent to the following:

```
   For each synchronous event
      Place any elementary actions that result in a queue
   For each elementary action in the queue
      Apply the action
```

So, for example, if the one of the actions resulting from the TokenMovedFrom event is to deactivate the link responding to the TokenMovedTo event the actions of the TokenMovedTo event will still be applied as they will have been queued before the actions from the TokenMovedFrom event start executing.

## 11.13.5 TransitionTo deactivation of shared=FALSE ingredients

The Shared attribute of the Ingredient class indicates whether the Ingredient object is intended for continuous presentation across a Scene transition. Whenever a TransitionTo action is executed, the Deactivation behaviour is applied to all active Ingredient objects of the currently active Application object that have the Shared parameter set to False. This happens regardless of whether there is a Scene active before the TransitionTo action is executed.

## 11.13.6 Interactibles

In accordance with ISO/IEC 13522-5 [14], MHEG-5 objects that belong to the class Interactible may be in a certain state, called "interacting", which is signalled by the InteractionStatus attribute of the object being True. When an object is in this state, no UserInput events shall be generated by the MHEG-5 engine.

A restriction on the receiver implementation of interaction methods for any Interactible is that the Text, Red, Green, Yellow and Blue key functions shall not be used. Further, even when the InteractionStatus is true, the EngineEvents corresponding to all key functions shall still be generated even if UserInput events are not - as described above.

The engine behaviour is undefined if SetData is targeted at an EntryField or a HyperText object while its InteractionStatus attribute is set to True.

## 11.13.7 Clarification of StreamPlaying and StreamStopped events

In this Profile, Stream objects generate StreamPlaying and StreamStopped events in accordance with ISO/IEC 13522-5 [14] and ISO/IEC 13522-5:1997/Cor.1:1999(E) [21]. Events shall be generated for all Stream objects whether of storage type stream or memory.

A StreamPlaying event shall only be generated when a Stream successfully starts playing.

A StreamStopped event shall only be generated when a previously-playing Stream stops. If the storage type is Stream this can only happen as the result of a Stop action (as broadcast streams have no logical end).

A Stop action on a Stream that failed to start playing shall not raise a StreamStopped event.

See also clause 14.2.4.

## 11.13.8 Use of NextScenes to preload content

In this Profile the Group Identifiers within the NextScenes attribute of the Scene class shall be regarded as context-free file names, and the files referenced can contain either MHEG-5 code or content.

## 11.13.9 Application defaults

See table 50.

**Table 50: Application defaults**

<table>
<tr><th>Attribute</th><th>Default value</th></tr>
<tr><td>Font</td><td>rec://font/uk1</td></tr>
<tr><td>FontAttribute</td><td>
<table>
<tr><th>Attribute</th><th>Value</th></tr>
<tr><td>Size</td><td>24 pt</td></tr>
<tr><td>Line spacing</td><td>28 pt</td></tr>
<tr><td>Letterspace</td><td>0</td></tr>
<tr><td>Style</td><td>plain</td></tr>
</table>
</td></tr>
<tr><td>TextColour</td><td>'=FF=FF=FF=00'</td></tr>
<tr><td>HighlightRefColour</td><td>'=FF=FF=FF=00'</td></tr>
<tr><td>SliderRefColour</td><td>'=FF=FF=FF=00'</td></tr>
<tr><td>CharacterSet</td><td>10</td></tr>
</table>

**Table 50: Application defaults**

| Attribute | Default value |
|---|---|
| TextContentHook | 10 |
| BitmapContentHook | 4 |
| StreamContentHook | 10 |
| Desktop Colour (NOTE) | '=00=00=00=00' |
| NOTE : i.e. the colour of the bottom of the display stack. | |

### 11.13.10 Video and bitmap scaling

In this Profile, SetData actions targeted to Stream and Bitmap objects do not reset any scaling factors set using ScaleBitmap or ScaleVideo. This overrides clauses 16.1 and 20.8 of ISO/IEC 13522-5:1997/Cor.1:1999(E) [21].

### 11.13.11 Clarification of TransitionTo, Launch and Spawn behaviour

ISO/IEC 13522-5 [14] defines the sequences of actions that take place in response to TransitionTo, Launch and Spawn ElementaryActions. In this Profile receivers shall ensure that the file containing the new Group is loaded and, where possible, is syntactically valid before beginning step one of those sequences. Such ElementaryActions that fail at this point are ignored. See also clause 17.15.

## 11.14 Debug package

The debug package provides MHEG-5 developers with a mechanism to debug their application. This package contains a Debug Resident Program to support this functionality.

### 11.14.1 Debug package resident programs

**Table 51: Debug package resident programs**

| Resident program | | Invocation | | | Reference |
|---|---|---|---|---|---|
| | | Typical use | | Never Fork | |
| Description | Name | Call | Fork | | |
| Debug | DBG | ✓ | | | clause 11.14.1.1. |

#### 11.14.1.1 Debug

Allows output of debug messages.

This resident program provides a mechanism for application authors to obtain debug output. Exactly where the debug message appears will depend on the hardware configuration: a set top box may use a serial port, or a PC TV card may open a text window.

##### 11.14.1.1.1 Synopsis

DBG( [argument] … )

#### 11.14.1.1.2 Arguments

Arguments for Debug are shown in table 52.

**Table 52: Debug parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericBoolean or GenericInteger or GenericOctetString or GenericObjectReference or GenericContentReference | argument | The first of the optional list of arguments. |

#### 11.14.1.1.3 Description

This resident program outputs a list of zero or more input arguments. The exact output representation for each variable-type is implementation dependent. Table 53 provides examples of suitable output.

**Table 53: Example debug output**

| Type | Output |
|---|---|
| GenericBoolean | False |
| GenericInteger | 180999 |
| GenericOctetString | It's evolving |
| GenericObjectReference | (DSM://debug.mhg, 20) |
| GenericContentReference | /a/logo.png |

The debug output is not implicitly terminated by a newline, allowing a sequence of calls to concatenate output on the same line. If newlines and tabs are required in the output, these shall be passed to the Debug resident program using octet string arguments containing the hexadecimal values 0x0D and 0x09 respectively.

## 11.15 Service Information package

The Service Information package provides access to certain SI data. It provides two resident programs to obtain service and event information.

This package is subject to refinement and extension as part of the programme of work for the next revision of this Profile.

### 11.15.1 Service Information resident programs

**Table 54: Service Information package resident programs**

| Resident program | | Invocation | | | Reference |
|---|---|---|---|---|---|
| | | Typical use | | Never Fork | |
| Description | Name | Call | Fork | | |
| SI_GetServiceInfo | SeI | ✓ | | | clause 11.15.1.1 |
| SI_GetEventInfo | GEI | ✓ | | | clause 11.15.1.2 |

#### 11.15.1.1 SI_GetServiceInfo

Retrieves for a service the service name, provider and type.

Support for this resident program is optional.

#### 11.15.1.1.1 Synopsis

SeI( serviceIndex, serviceName, serviceProvider, serviceType )

#### 11.15.1.1.2 Arguments

Arguments for SI_GetService Info are shown in table 55.

**Table 55: SI_GetServiceInfo parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | serviceIndex | This integer is a receiver specific identifier for the service about which basic SI is required (see clause 11.10.8.1). |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | serviceName | Returns the service name (derived from the service's SI service descriptor). |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | serviceProvider | Returns the service provider name (derived from the service's SI service descriptor). |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | serviceType | Returns the service type (derived from the service's SI service descriptor). |

#### 11.15.1.1.3 Description

The resident program returns a series of values from a service's SI service descriptor. The service is identified by means of a receiver specific "ServiceIndex". This integer can be determined by means of the SI_GetServiceIndex resident program (see clause 11.10.8.1).

### 11.15.1.2 SI_GetEventInfo

Provides information for the present or following event for a service.

Support for this resident program is optional.

#### 11.15.1.2.1 Synopsis

GEI( serviceIndex, porf, eventName, shortDescription, parentalRating, startDate, startTime, duration, category, freeNotCA )

#### 11.15.1.2.2 Arguments

Arguments for SI_GetEventInfo are shown in table 56.

**Table 56: SI_GetEventInfo parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| input | GenericInteger | serviceIndex | This integer is a receiver specific identifier for the service about which basic SI is required (see clause 11.10.8.1). |
| input | GenericInteger | porf | Returns if the present or following event information should be retrieved (0 = present, 1 = following). |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | eventName | Returns the name of the event extracted from the EIT short event descriptor. A zero length string may be returned if this information is not available. |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | shortDescription | Returns the short description of the event extracted from the EIT short event descriptor (if any). A zero length string may be returned if this information is not available. |

**Table 56: SI_GetEventInfo parameters**

| In/out / in-out | Type | Name | Comment |
|---|---|---|---|
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | parentalRating | Returns the parental rating (appropriate for the country where the receiver is installed) for the event extracted from the EIT parental rating descriptor. The value zero is returned if no rating is available. |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | startDate | Returns the start date for the event. The coding of this is identical to that used by the GetCurrentDate resident program (see clause 11.10.4.2). |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | startTime | Returns the start time for the event. The coding of this is identical to that used by the GetCurrentDate resident program (see clause 11.10.4.2). |
| output | GenericInteger (Shall provide an IndirectReference to an IntegerVariable) | duration | Returns the duration of the event in seconds. |
| output | GenericOctetString (Shall provide an IndirectReference to an OctetStringVariable) | category | Returns the content nibbles from the DVB SI content descriptor. |
| output | GenericBoolean (Shall provide an IndirectReference to a BooleanVariable) | freeNotCA | Indicates if the event is CA controlled or not (true indicates free). This does not imply a CA query has to be performed (so it does not inform about entitlement to access the event). |

#### 11.15.1.2.3 Description

The resident program returns a the present or following event data for a service, determined by the serviceIndex integer. This integer can be determined by means of the SI_GetServiceIndex resident program (see clause 11.10.8.1).

The encoding for the DVB SI content nibbles, returned in the category parameter, is:

- content_nibble_level_1 is encoded the character code 0x0041 + content_nibble_level_1 (i.e. the content_nibble_level_1 yields a character "A" to "P")
- content_nibble_level_2 is encoded the character code 0x0061 + content_nibble_level_2 (i.e. the content_nibble_level_1 yields a character "a" to "p")

If there are multiple content facets additional letter pairs shall be returned (for example, the returned string might be "BgGe" for a programme classified as "movie\romance", "Music/Ballet/Dance\Jazz").

# 12 MHEG-5 engine graphics model

## 12.1 The graphics plane

The "graphics plane" is used to represent all visibles except video streams and MPEG I-frame bitmap objects. (MPEG I-frames and video are assumed to reside in a separate truecolour display buffer.)

NOTE 1: The drawing area available for applications has pixel dimensions of 720 x 576.

NOTE 2: Visible area: see clause 17.3.2.

NOTE 3: Graphics/video pixel alignment: see clause 14.4.4.

NOTE 4: Colour range: the graphics plane shall be able to support colours at least subjectively equivalent to the 256 colours in clause 12.2.

## 12.2 The colour palette

The graphics plane shall support at least 256 colours.

NOTE: The present document allows the graphics plane to be implemented as either an 8-bit indexed store or a truecolour store.

To accommodate receivers with an 8-bit indexed graphics system a single 256 colour CLUT is defined. This can be shared between the possibly concurrent demands of the MHEG-5 engine, subtitle decoding and manufacturer specific receiver applications.

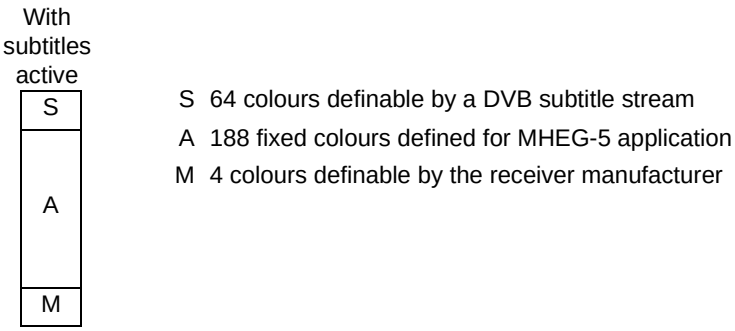



**Figure 7: Colour palette division between concurrent processes**

### 12.2.1 Reservation for MHEG-5 applications

188 locations (the "A" palette in figure 7) are reserved for use when displaying MHEG-5 applications. Receivers may support more than the minimum 188 colours in the "A" palette.

#### 12.2.1.1 Fidelity of reproduction

When an application invokes a colour in the "A" palette it shall be reproduced exactly. If applications invoke colours that are not in the currently active palette they shall be reproduced in an implementation dependent way.

#### 12.2.1.2 Palette definition

Table 57 defines the colour combinations in the "A" palette.

**Table 57: "A" palette construction rules**

| Transparency | | Additional grey levels (R=G=B) | Red | Green | Blue | Number of colours |
|---|---|---|---|---|---|---|
| 0 % | 0x00 | 42, 85, 170,212 | 0, 63, 127, 191, 255 | 0, 31, 63, 95, 127, 159, 191, 223, 255 | 0, 127, 255 | 139 |
| 30 % | 0x4C (NOTE ) | -- | 0, 85, 170, 255 | 0, 51, 102, 153, 204, 255 | 0, 255 | 48 |
| 100 % | 0xFF | -- | -- | -- | -- | 1 |
| | | | | | Total | 188 |
| NOTE : Where the receiver cannot implement this "ideal" value of semi-transparency it shall replace it with the nearest value of semi-transparency it can implement. Note that the 30 % transparency level shall not be approximated as either 0 % or 100 % transparency. | | | | | | |

Figure 8 illustrates the opaque colours in the palette

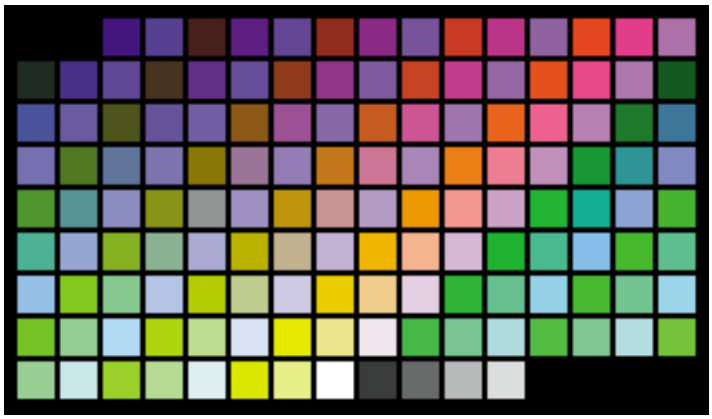


**Figure 8: "A" palette (showing opaque colours only)**

### 12.2.2 Reservation for DVB subtitles

64 locations (the "S" palette in figure 7) are reserved for use when displaying DVB subtitles. Some DVB subtitle encoding constraints restrict broadcasts to use colour indices in the range 0 to 63. The "S" palette may be dynamically loaded during subtitle decoding.

### 12.2.3 Subtitle priority for transparency

Where the "S" palette contains values of semi-transparency different from those in the "M" and "A" palettes and subtitles are enabled for presentation (see clause 14.3.3) then the subtitle decoding shall have priority if the receiver is not able to meet both sets of requirements.

### 12.2.4 Reservation for manufacturer use

Four locations ("M" in figure 7) are reserved for receiver manufacturer use.

# 12.3 Colour representation

## 12.3.1 Colour space

**The engine is responsible for converting between colour spaces as is required.**

Depending on the content type the MHEG-5 engine handles colours in both RGB (colour for buttons, text etc. and PNG graphics) and $YC_rC_b$ (MPEG stills and DVB subtitles) colour spaces (see figures 9 and 10).

Broadcasts shall use colorimetry as defined by ITU-R Recommendation BT.470-4 System I (which is explicitly the same as ITU-R Recommendation BT.470-2 System B,G). This defines the relationship between RGB and Y-Cr-Cb, which shall be used wherever a transformation from one representation to the other is required.

The RGB components defining the receiver colour palette are in the range 0 to 255. This range shall map linearly into the range 0 to 1 used for the ITU specified transformation.

The present document does not comment on the colour representation used by the receiver as long as the relationship between colours in the graphics and video planes is maintained.

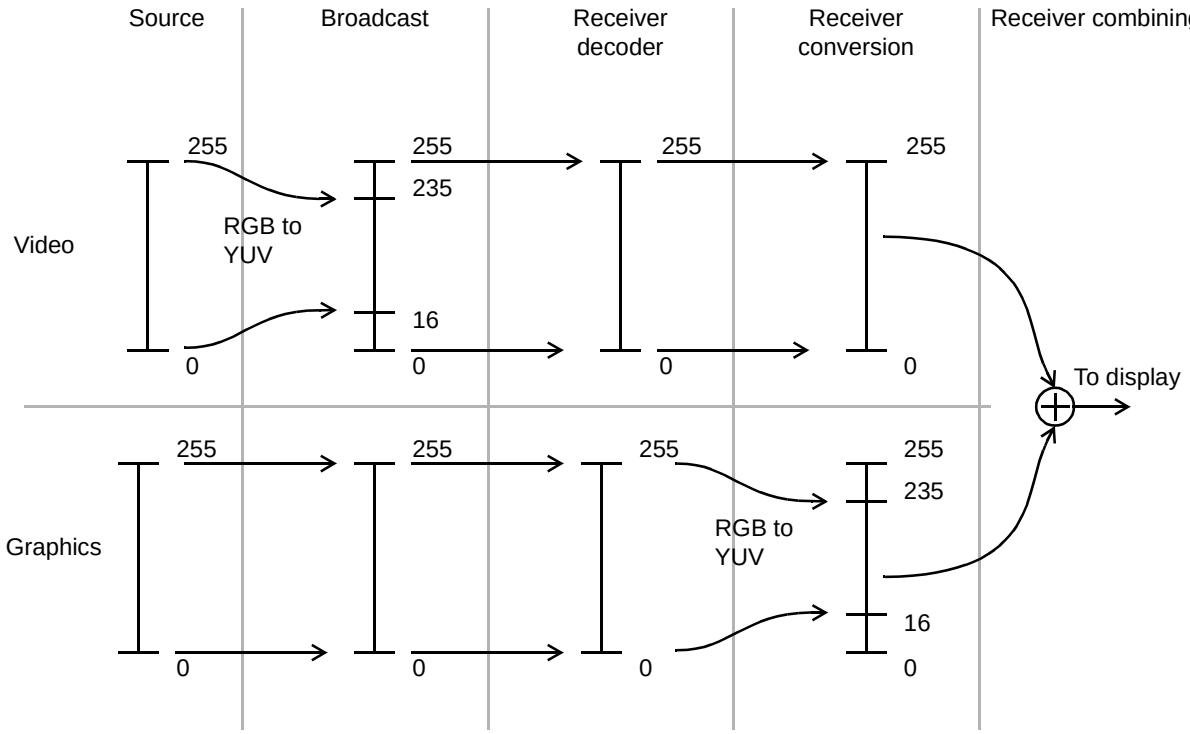



**Figure 9: A receiver with video and graphics combined in YUV form**

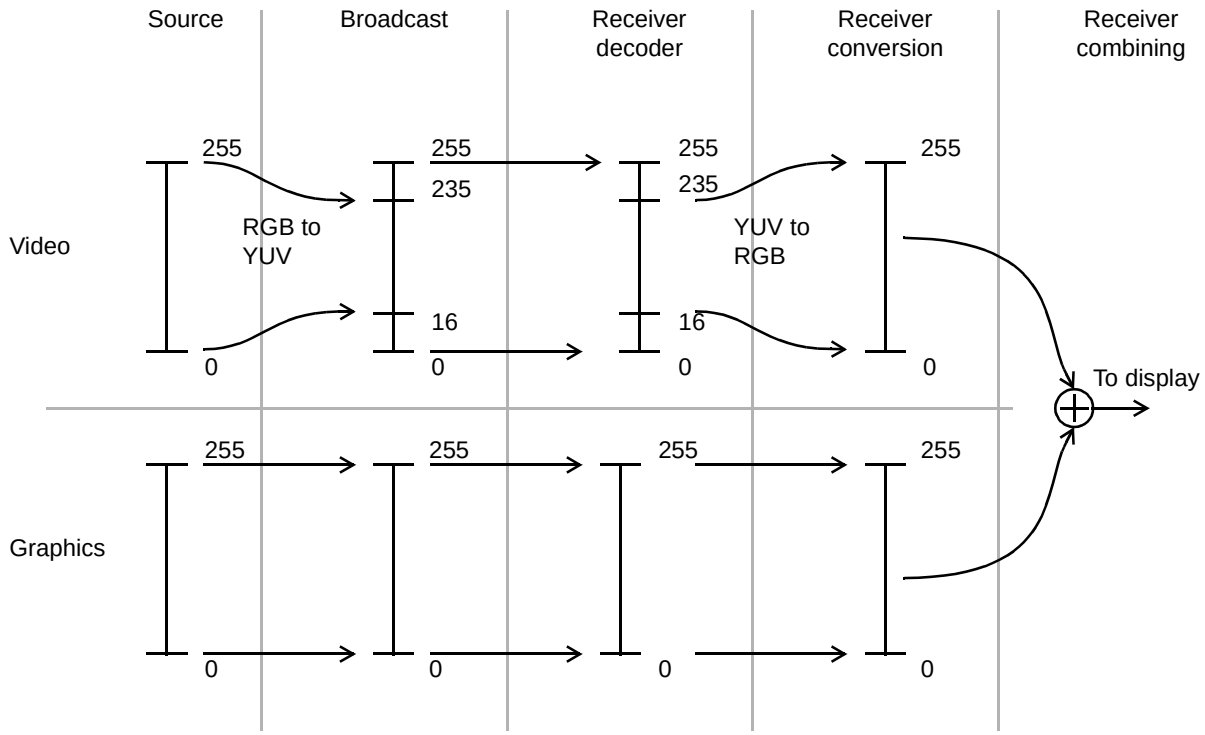



**Figure 10: A receiver with video and graphics combined in RGB form**

## 12.3.2 Gamma

MPEG video and DVB subtitles deliver $YC_rC_b$ data in accordance with ITU-R Recommendation BT.601. These signals are gamma pre-corrected. **Receivers shall assume that ALL RGB values invoked by MHEG-5 applications are comparably gamma pre-corrected.**

Application authors are advised to pre-correct RGB values (such as those in PNG graphics) to be consistent with the pre-correction applied to the MPEG video.

## 12.3.3 Direct/absolute colours

Direct/absolute colour values in MHEG-5 applications shall be expressed as a 4-byte OctetString constructed as shown in figure 11.

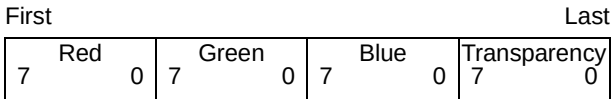



**Figure 11: Absolute colour format**

The Red, Green and Blue codes are values in the range 0 to 255.

The Transparency byte codes are values in the range 0 to 255 representing transparency. 0 is opaque.

## 12.3.4 Approximation of transparency

Receivers are required to implement three levels of transparency: 0 % (opaque), 30 % and 100 % (completely transparent). Implementation of additional intermediate levels of transparency is optional. The MHEG-5 encoding of colours within PNG bitmaps can convey a wider range of transparency.

Where the receiver cannot implement an encoded value of semi-transparency it shall replace it with the nearest value of transparency it can implement.

However, if the encoded value of transparency is in the range 10 % to 90% / 0x19 to 0xE6 it shall not be approximated as either 0 % or 100 % transparency.

Therefore, 9 % may be approximated as 0 % but 10 % shall be represented with a value in the range 10 % to 90 % such as 30 %. Similarly, 91 % may be approximated as 100 %.

### 12.3.5 PNG modes

See clause 12.7.2.

## 12.4 Overlapping visibles

### 12.4.1 Transparency and overlapping visibles

#### 12.4.1.1 Overlaying visibles

When visibles overlap, the MHEG-5 rules for rendering Visibles shall be observed where transparency is 0 % or 100 % or where semi-transparent pixels are the only visible (i.e. with transparency < 100 %) pixels above MPEG video or an MPEG I-frame.

Where intermediate levels of transparency overlay other forms of Visible, certain approximations are permitted. If semi-transparent pixels overlay one or more layers of semi-transparent pixels, allowed approximations are for the top-most semi-transparent pixel to "punch through" intervening semi-transparent pixels to the video or be treated as opaque. However, semi-transparent pixels are not allowed to "punch through" opaque pixels (see figure 12).

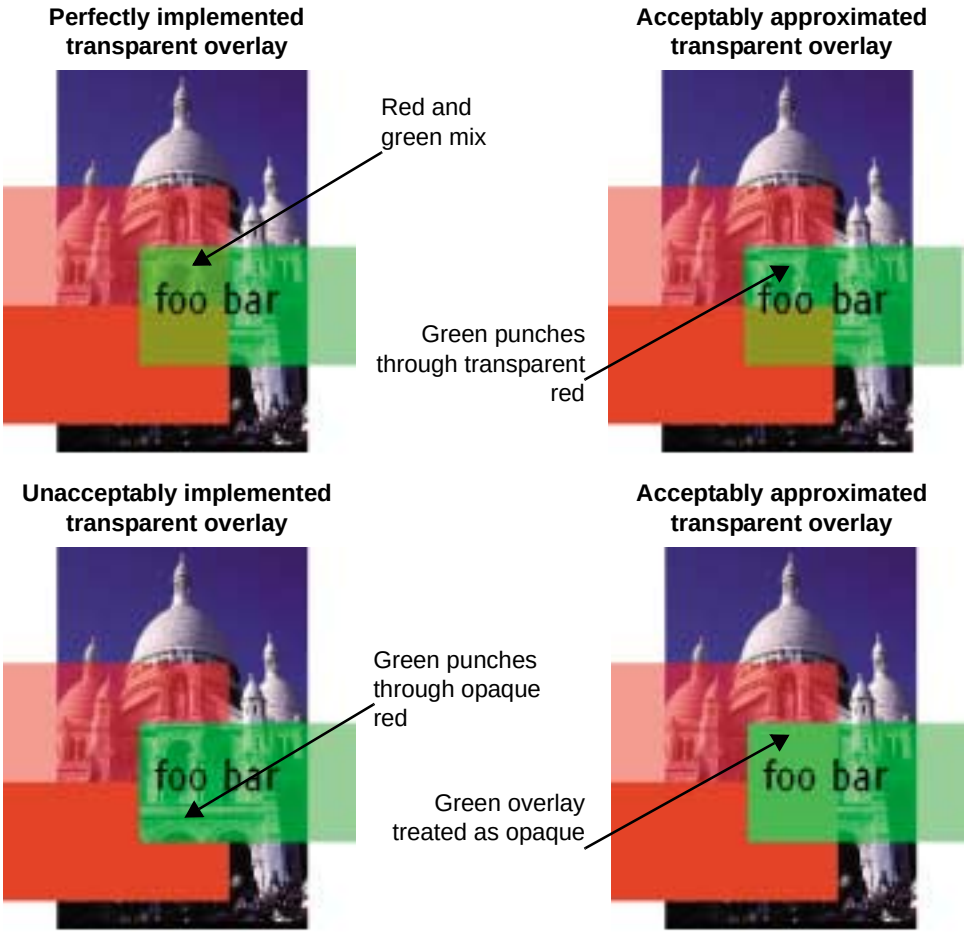



**Figure 12: Approximation of transparency**

#### 12.4.1.2 Rendering performance

Authors should be aware that the graphics drawing speed may decrease where visibles with an intermediate level of transparency are placed directly over objects other than MPEG video or MPEG I-frames.

# 12.5 LineArt and DynamicLineArt

## 12.5.1 Clarifications

### 12.5.1.1 Lineart borders

The fill colour of Rectangle and DynamicLineArt objects shall not extend into any border area. Transparent or semi-transparent borders shall be rendered directly on top of underlying objects.

### 12.5.1.2 "Fat" lines

#### 12.5.1.2.1 "Fat" lines are centred

Draw actions targeted at a DynamicLineArt object describe the course of a nominal zero width line. On top of this nominal line, a line with width LineWidth is painted. Ideally, the painted line is centred on the nominal line. Engines approximate this centring in an implementation dependent way. This behaviour applies to all draw actions.

NOTE: The behaviour of the DrawRectangle action is different from the behaviour of the Rectangle class.

#### 12.5.1.2.2 Clipping at box edge

"Fat" lines running overlapping the edge of the DynamicLineArt object or near its border will be cropped. The exact behaviour of this cropping depends on the implementation dependent way in which the "fat" line is aligned to the nominal line.

### 12.5.1.3 Line ends

The appearance of the ends of lines and the junctions in polygons and polylines is implementation dependent.

### 12.5.1.4 Bordered bounding box

The border area, LineWidth pixels wide, of a DynamicLineArt object with BorderedBoundingBox set True, clips draw actions to the DynamicLineArt object. The origin of the coordinate system for draw actions remains the Position of the object (see figure 13).

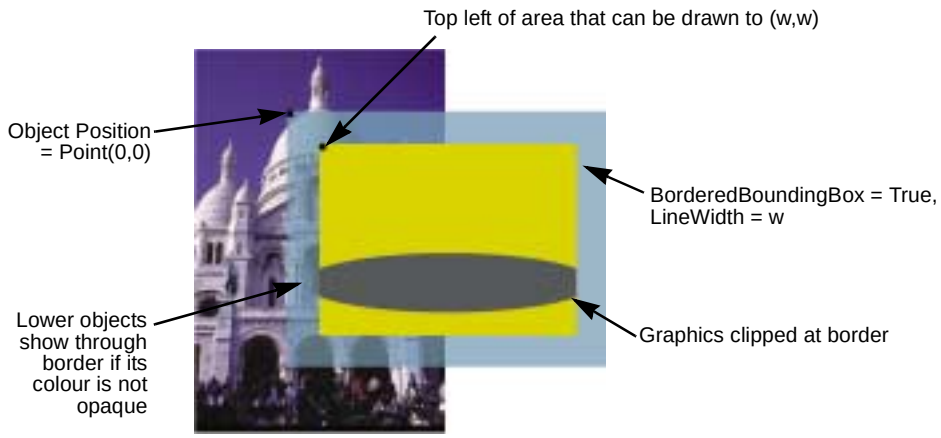



**Figure 13: Graphics clipped at border**

#### 12.5.1.5 DrawSector

Figure 14 illustrates the results of a DrawSector action where RefLineColour is green and RefFillColour is yellow.

**Figure 14: DrawSector illustrated**

#### 12.5.1.6 Effect of pixel transparency

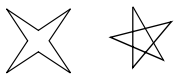
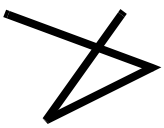
DynamicLineArt (DLA) objects are regarded as two-dimensional drawing surfaces. Drawing actions targeted to a DLA object cause pixels to be simply replaced with the current LineColour attribute.

Note that this means that no alpha mixing is performed at an intra-object level (i.e. between the pixels caused by successive actions on the same DLA), but that when the DLA object is rendered there is inter-object alpha mixing with the objects below the DLA in the display stack. For this model the transparent colour is regarded as a valid pixel colour (and not the lack of a colour).

### 12.5.2 Limitations

The allowances in table 58 are made to assist receiver implementation. Authors should take account of the implied authoring constraints.

**Table 58: Limitations on LineArt and DynamicLineArt**

| Topic | Receiver allowance | Authoring guideline |
|---|---|---|
| LineStyle attribute | Implement ALL line styles as solid. | Avoid using dashed or dotted line styles (other line attributes such as width or colour should be used to differentiate line styles). |
| Filled closed shapes | The receiver behaviour when filling certain shapes is not defined. These shapes are:<br>- concave polygons; and<br>- self-crossing polygons.<br><br>All other shapes shall be completely filled with the colour defined by the RefFillColour attribute. | Avoid using filled concave polygons and filled self-crossing polygons, i.e. avoid filling shapes such as:<br>  <br>If these shapes such as these are required filled they may be constructed from primitive elements such as triangles which are guaranteed to fill in a predictable way. |
| Self-crossing polygons and polylines | The appearance of pixels at the junction of self-crossing lines is not defined. | Avoid self-crossing lines such as:<br>  |

## 12.6 Text, EntryFields and HyperText

Text objects (and the text areas in EntryField and HyperText objects) are treated as two inseparable layers in the display stack. The lower layer is the "paper" and immediately above this the "ink". The display stack rendering rules used in this Profile apply to these layers (see clause 12.4.1).

Therefore, the "paper" colour (the BackgroundColour of the Text) (if any) is painted over the objects below it in the display stack applying whatever blending is appropriate. Then over this the "ink" colour (the TextColour of the Text) is painted, again applying whatever blending is appropriate.

# 12.7 PNG bitmaps

## 12.7.1 Specification conformance

PNG bitmaps shall be encoded in conformance with version 1.0 of PNG. The treatment of the chunks described by version 1.0 are described in table 59. *ETSIEngineProfile1* decoders shall ignore any additional chunks.

**Table 59: Treatment of PNG chunks by *ETSIEngineProfile1***

| Chunk | Comment |
|---|---|
| IHDR | Shall be used in the decoding process as described by the PNG specification. |
| PLTE | |
| IDAT | |
| IEND | |
| cHRM | Receiver shall skip. |
| gAMA | Receiver shall skip. |
| sBIT | Receiver can skip. |
| bKGD | Receiver shall skip. |
| hIST | Receiver can skip. |
| tRNS | **Receiver shall consider.** |
| pHYs | Receiver shall skip. |
| tIME | Receiver shall skip. |
| tEXt | Receiver shall skip. |
| zTXt | Receiver shall skip. |

See also clause 11.5.2.

## 12.7.2 Colour encoding

**Engines are required to support ALL of the PNG colour types defined in PNG Specification version 1 (see PNG [17] ). Engines are responsible for mapping these colours to those used by the engine's OSD.**

See table 60.

**Table 60: PNG formats**

| Colour type | Allowed bit depths | Interpretation |
|---|---|---|
| 0 | 1,2,4,8,16 | Each pixel is a grayscale sample. |
| 2 | 8,16 | Each pixel is an R,G,B triple. |
| 3 | 1,2,4,8 | Each pixel is a palette index; PLTE chunk must appear. |
| 4 | 8,16 | Each pixel is a grayscale sample, followed by an alpha sample. |
| 6 | 8,16 | Each pixel is an R,G,B triple, followed by an alpha sample. |

Any combination of PNGs with different colour types may be active at any one time. Similarly, engines are responsible for mapping RGB16 direct colour specifications to colours that the OSD can support.

Where PNG graphics use colours defined in the minimum application palette (A) these colours shall be reproduced correctly. Other colours shall be reproduced in an implementation dependent way.

## 12.7.3 Aspect ratio signalling

In this Profile the pixel aspect ratio of the bitmap (whether implicitly or explicitly defined) and size of pixels shall be ignored and pixels mapped 1-to-1 into the SceneCoordinateSystem.

# 12.8 MPEG stills

## 12.8.1 File format

The payload of a file delivering an MPEG I frame shall:

- be a valid video_sequence() including a sequence_extension();
- contain one I frame, i.e. one picture_header(), one picture_coding_extension(), and one picture_data() encoded as an intra coded frame, with picture structure = "frame".

The structure is:

```
sequence_header()
sequence_extension()
extension_and_user_data(0)
optional group_of_pictures_header() and extension_and_user_data(2)
picture_header( picture_coding_type = "I frame")
picture_coding_extension( picture_structure = "frame picture")
picture_data()
sequence_end_code()
```

## 12.8.2 Semantics

An MPEG-2 video decoder conforming to the same behaviour as the main video decoder (ETSI TR 101 154 [3]) is used to decode the fragment of data containing the I-frame.

## 12.8.3 Presentation

See clause 14.4.4 for the description of the presentation of MPEG stills.

# 12.9 MPEG video

See clause 14.4.4.

# 12.10 Appearance of Visible objects during content retrieval

Whilst content is being retrieved from the broadcast, Visible objects shall be displayed in the following way:

- for Bitmap objects: completely transparent;
- for Video objects: opaque black;
- for Text objects: as an empty Text object.

# 13 Text and interactibles

## 13.1 Text rendering overview

This clause addresses the encoding of text and how its presentation is controlled and behaves. The application of these rules to classes that **present** text is summarized in table 61.

**Table 61: Application of rules to classes**

| | Character encoding | FontAttributes | Text rendering | Text mark-up | HyperText mark-up |
|---|---|---|---|---|---|
| **Text** | ✓ | ✓ | ✓ | ✓ | |
| **EntryFields (NOTE 1)** | ✓ | ✓ | ✓ | | |
| **HyperText (NOTE 2)** | ✓ | ✓ | ✓ | ✓ | ✓ |
| NOTE 1: See clause 13.7. | | | | | |
| NOTE 2: See clause 13.8. | | | | | |

### 13.1.1 Non-presented text

No restrictions are placed on the byte values in OctetStrings that are **not** for presentation except as listed below:

- GroupIdentifiers: see clause 16.3.2;
- Type conversion: where OctetStrings are converted by the "CastToContentRef" and "CastToObjectRef" resident programs (see table 22) **no** processing is applied to the byte values (i.e. it is just a type conversion).

**It is the author's responsibility to ensure that the byte values in the resultant string are suitable for the context in which they are next used.**

## 13.2 Character encoding

Table 78 lists the minimum set of characters supported by the engine. Characters shall be encoded according to ISO/IEC 10646 [13] and the Universal Character Set Transformation Format, 8-bit format (UTF-8) which is standardized as Amendment 1 to ISO/IEC 10646 [13].

### 13.2.1 UTF-8

Table 62 reproduces the UTF-8 coding scheme. The character repertoire in table 78 will only require 1, 2 or 3-byte codes. Where text is in English, Welsh or Gaelic, the majority of characters will be coded on 1 byte.

**Table 62: UTF-8-bit distribution**

| ISO/IEC 10646 [13] value | 1st byte | 2nd byte | 3rd byte | 4th byte |
|---|---|---|---|---|
| `0000 0000 0xxx xxxx` | `0xxx xxxx` | | | |
| `0000 0yyy yyxx xxxx` | `110y yyyy` | `10xx xxxx` | | |
| `zzzz yyyy yyxx xxxx` | `1110 zzzz` | `10yy yyyy` | `10xx xxxx` | |
| `1101 10ww wwzz zzyy +`<br>`1101 11yy yyxx xxxx` | `1111 0uuu`<br>`(NOTE)` | `10uu zzzz` | `10yy yyyy` | `10xx xxxx` |
| NOTE : Where uuuuu = wwww + 1. | | | | |

### 13.2.2 Null characters

The treatment of code zero (null) differs between the ISO specification of UTF-8 and that used in Java systems. However, this is transparent to the MHEG-5 environment as null terminated strings are not used.

### 13.2.3 CharacterSet attribute

Table 63 identifies the minimum set of values of CharacterSet (as used in the Application, Text and PushButton classes) attribute that the engine shall support.

**Table 63: Engine CharacterSet attributes**

| Attribute | Character set |
|---|---|
| <10 | Reserved for future use or other application domains. |
| 10 | The subset of ISO/IEC 10646 [13] tabulated in table 78 encoded with UTF-8. |
| >10 | Reserved for future use. |

## 13.3 Fonts

### 13.3.1 Downloading

Receivers implementing *ETSIEngineProfile1* shall not support downloadable fonts or the Font class. Application references to fonts shall be direct (i.e. an OctetString representing the name of the font).

#### 13.3.1.1 Future compatibility

The measures outlined below are designed to improve interoperability with possible future services that make use of downloaded fonts. The preferred approach is the "authoring solution" (see clause 13.3.1.1.1). However, receivers shall implement the "receiver defensive response" (see clause 13.3.1.1.2).

##### 13.3.1.1.1 Authoring solution

Future applications shall be able to determine (via the GetEngineSupport mechanism) the capabilities of the receiver. Using this information applications can adapt their behaviour to avoid problems.

##### 13.3.1.1.2 Receiver defensive response

Receivers shall implement the following measures to ensure robust behaviour with any future applications that do use downloaded fonts, or font characteristics that the receiver doesn't recognize:

- the receiver shall use its in-built font;
- if the font style is recognized then it shall be used; if not, plain shall be used;
- if the font size does not match one of those of the in-built font the receiver shall substitute the next smaller size provided by the in-built font. If the required font is smaller than the smallest available, then the smallest available in-built size shall be used;
- any character not in the character repertoire in table 78 shall not be considered as part of the input.

### 13.3.2 Embedded font

#### 13.3.2.1 Font version

All receivers shall use the metrics defined in v7.51 of the Tiresias font.

NOTE: Other versions of the font are not automatically guaranteed to have compatible metrics.

It is recommended that any receiver implementing 1-bit/pixel rendering use the bitmaps also contained in this release.

The bitmaps contained in v7.51 ensure that the 1-bit/pixel representation is suitable for use with a typical TV display. This takes into account:

- interlace and limited resolution of typical TV displays - particularly for small font sizes;
- the need to minimize the risk that the physical rendering might exceed the logically available width (see clause 13.5.1).

#### 13.3.2.2 Required sizes and styles

Receivers shall implement at least the DTG/RNIB font in at least the sizes identified in table 64. This shall be the default font implemented by the engine.

**Table 64: "UK1" sizes and styles**

| Size (points) | TV lines over "Cap-V" (NOTE 1) | Informative name | Styles |
|---|---|---|---|
| | | | Plain |
| 36 | 24 | Heading / large subtitle | ✓ |
| 31 | 21 | Subtitle | ✓ |
| 26 | 18 | Body | ✓ |
| 24 | 16 | Footnote | ✓ (NOTE 2) |
| NOTE 1: The primary definition of the character size is the font size in points, the height of a capital letter "V" in TV lines is provided for information only. | | | |
| NOTE 2: The default size and style (see also clause 11.13.9). | | | |

### 13.3.3 Invoking the font

The DTG/RNIB font is invoked by using "rec://font/uk1" as the FontName for an Application object Font attribute or the OriginalFont attribute of the Text class.

The default font for engines conforming to *ETSIEngineProfile1* is "rec://font/uk1".

## 13.4 Text object attributes

### 13.4.1 FontAttributes

Receivers shall support two font attribute formats, one is textual (but verbose), the second is terser. The short and long forms shall not be mixed within an attribute string.

#### 13.4.1.1 Textual form

This string format <style>.<size>.<linespace>.<letterspace> is carried in an OctetString. For example, "plain.26.32.0" means plain 26-point text on 32-point line spacing with default letterspace.

Long form text format parameters are shown in table 65.

**Table 65: Long form text format parameters**

| Field | Set of allowed values | Meaning |
|---|---|---|
| style | "plain" | Plain text. |
| size | "24" "26" "31" "36" | Font size in points as decimal integer strings. |
| linespace | "0" to "255" (NOTE) | Space between the baselines of adjacent lines of text in points as decimal integer strings. |
| letterspace | "-32767" to "32767" (NOTE) | Increase in spacing in 1/256 points between consecutive characters expressed as a signed decimal integer string. |
| NOTE : Values outside this allowed range shall be limited to the nearest allowed value. | | |

#### 13.4.1.2 Short form

The font attributes are a 5-byte OctetString.

Short form text format parameters are shown in table 66.

**Table 66: Short form text format parameters**

| Syntax | Bits | Type | Allowed values |
|---|---|---|---|
| `style` | 8 | bslbf | See table 67. |
| `size` | 8 | uimsbf | 0x18, 0x1A, 0x1F or 0x24 |
| `linespace` | 8 | uimsbf | 0 to 255 |
| `letterspace` | 16 | tcimsbf | -32767 to 32767 |

**style:** An 8-bit string coded as shown in table 67.

**size:** An 8-bit unsigned integer giving the height of the font face in points.

**linespace:** An 8-bit unsigned integer giving the spacing between the baselines of adjacent lines of text in points.

**letterspace:** A 16-bit signed integer specifying in units of 1/256th point the required increase in the spacing between consecutive characters.

**Table 67: Coding of "style" (includes future coding use)**

| Style bit field | | | | | | | | Style | Illustrative plain text characters possibly convenient to authors when entering "style" in textual notation |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Do not care | | | | Outline | Underline | Bold | Italic (NOTE) | | |
| x | x | x | x | 0 | 0 | 0 | 0 | Plain | " " "0" "@" |
| x | x | x | x | 0 | 0 | 0 | 1 | Italic | "!" "1" "A" |
| x | x | x | x | 0 | 0 | 1 | 0 | Bold | """ "2" "B" |
| x | x | x | x | 0 | 0 | 1 | 1 | Bold-italic | "#" "3" "C" |
| NOTE : Only plain is supported by this Profile. | | | | | | | | | |

## 13.4.2 Control of text flow

### 13.4.2.1 Required flow modes

Receivers shall implement at least the required set of text flow modes identified in table 68.

**Table 68: Required set of text flow modes**

| Attribute | Required values | Optional flow modes | | Notes |
|---|---|---|---|---|
| | | Optional value (NOTE 1) | Replacement alternative (NOTE 2) | |
| HorizontalJustification | start, end, centre | justified | start | See clause 13.5.4. |
| VerticalJustification | start, end, centre | justified | start | |
| LineOrientation | horizontal | vertical | horizontal | |
| StartCorner | upper-left | upper-right, lower-left, lower-right | upper-left | |
| TextWrapping | true, false | | | See clause 13.5.6. |
| NOTE 1: Implementation of the attribute values in this column is optional. | | | | |
| NOTE 2: This column defines the flow mode that shall be implemented by an engine when requested to implement an optional flow mode that it does not support. | | | | |

See also clause 17.8.2.

# 13.5 Text rendering

## 13.5.1 Philosophy

This clause describes "logical" rules that ensure text flows identically on all receivers and defines some rendering requirements to ensure that a minimum acceptable level of text legibility is achieved even when using very simple bitmap rendering.

No restriction is placed on the rendering technology used in a receiver provided that it achieves the deterministic text flow characteristics and the minimum rendering requirements described in this clause.

The conceptual rendering process can be described as follows:

1) Based on the size of the Text object to render into and characteristics of the font, calculate:

   - the maximum number of lines of text that may be rendered;
   - the width available for rendering on each line.

   See clause 13.5.4.

2) Determine how the text to render flows, effectively defining a series of lines to render using:

   - the "logical" rules for calculating the width of rendered text (see clause 13.5.5);
   - the available width for rendering (from Step 1);
   - the rules for breaking text (see clause 13.5.6).

3) Determine where each line of text to render is placed vertically within the Text object (see clause 13.5.7). This needs to consider that if the number of lines of text to render (from Step 2) exceeds the maximum number of lines of text that may be rendered (from Step 1) "vertical truncation" may be required, i.e. discard some of the lines to render.

   The positioning of lines to render is affected by the vertical justification of the Text object.

4) Determine the placement of individual characters in a line to render (see clause 13.5.8). This needs to consider that:

   - even having correctly applied the rules relating to the flow of text, in some extreme circumstances the length of the line of text to render can be wider than the available width for rendering (from Step 1). To handle this, "horizontal truncation" may be required, i.e. discard some of the characters from the line to render;
   - the placement of characters in the line to render is affected by the horizontal justification of the Text object;
   - the placement of characters in the line to render should ensure sensible and consistent spacing between adjacent characters (see clause 13.5.10).

It is also worth pointing out that special rules exist for handling tabulation (see clause 13.5.9) which need to be taken into account in the implementation of many of these steps.

NOTE: The order of these steps is illustrative and may vary between implementations for reasons of convenience or efficiency.

## 13.5.2 Font definition

The characteristics of the fonts used by the engine shall be defined in terms of the Portable Font Resource (PFR). This font format was selected since it is a "public" data structure due to its inclusion in relevant DAVIC specifications DAVIC 1.4.1 Part 09 [16]. The PFR specification is available from Bitstream's website (www.bitstream.com).

There is no requirement for receivers to embed font data using this format and it should be considered simply as a convenient "container" for publishing.

#### 13.5.2.1 Font bounds

The PFR font definition includes a set of parameters xMin, xMax, yMin and yMax that are **properties of the font**. These define the maximum extent of the outline representation of characters within the physical font, and as such are defined in terms of outline resolution units (outlineResolution).

(xMin, yMin) and (xMax, yMax) are the bottom-left and top-right corners of an imaginary bounding rectangle within which all characters in the font can be completely enclosed (see figure 15).

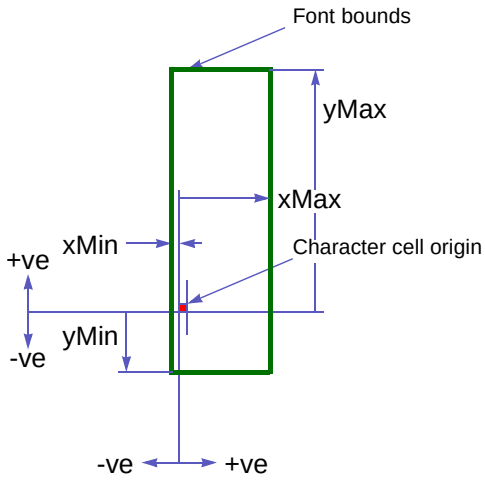



**Figure 15: Font bounds**

In this Profile these parameters are used to position the text within a Text object to guarantee that the extremities of all characters are completely within the Text object. See clause 13.5.4.

#### 13.5.2.2 "Physical" font data

"Physical" font data such as horizontal escapement and kerning is defined in the PFR in terms of metrics resolution units (metricsResolution). This is a high resolution representation, abstracted from any actual rendering system.

NOTE: **The outlineResolution and metricsResolution are not necessarily the same.**

### 13.5.3 Converting font metrics to display pixels

Many of the calculations in this clause are in a high resolution physical coordinate system, either metrics or outline resolutions. These values need to be converted into the pixel resolution of the display device to allow characters to be rendered.

Values in terms of these high level resolutions can be simply converted to values in terms of points by multiplying by the font size (in points) and dividing by the resolution, i.e. metricsResolution or outlineResolution as appropriate. However, this value in points still needs to be converted into a value in pixels.

Computer display systems typically assume a 72 pixel per inch display. So, as each point is 1/72 inch, the horizontal and vertical size of each pixel is 1 point.

#### 13.5.3.1 Vertical resolution

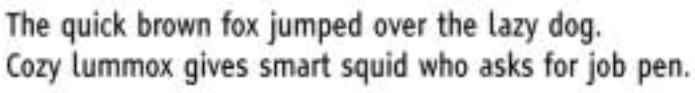
The computer convention is preserved for the vertical dimension of the MHEG-5 display (i.e. each of 576 OSD lines is considered to be 1 point high).

#### 13.5.3.2 Horizontal resolution

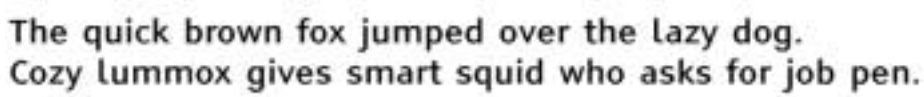
Due to the non-square pixel nature of broadcast displays the 1 pixel = 1 point convention cannot be preserved.

**To simplify the *ETSIEngineProfile1* ALL receivers shall assume that each of the 720 pixels has a uniform width of 56/45 points regardless of the scene aspect ratio and display aspect ratio.** This approximates the pixel aspect ratio for a 14:9 aspect ratio display. So, all text will be subject to moderate aspect ratio distortion. However, broadcasters will be able to predict text flow without having to author specifically for each display type. Also, this allows receivers to be implemented with font bitmaps in a single aspect ratio.

Text on a 4:3 display:




Text on a 16:9 display:




### 13.5.4 Rendering within the limits of the Text object

When typesetting for print, character extremities may extend beyond the nominal text flow area. However, print has margins so the edge of the text flow is not the technical limit to the area that can be printed. In this Profile the bounds of the Text object are treated as the technical limit to the area that can be printed. Taking this into account a "virtual margin" shall be defined using properties of the font (xMin, yMin, xMax and yMax) to ensure that all presented characters are completely rendered within the bounds of the Text object.

As stated previously, these parameters are defined in outline resolution units and so need to be converted to TV pixels. Based on the principles described previously (see clause 13.5.3) this can be achieved by using the following:

- yOffsetTop

$$OffsetTop_{pixels} = \begin{cases} div(yMax_{outlineResolution} \times fontSize, outlineResolution) & yMax > 0 \\ 0 & yMax \leq 0 \end{cases}$$

- yOffsetBottom

$$yOffsetBottom_{pixels} = \begin{cases} div(-yMin_{outlineResolution} \times fontSize, outlineResolution) & yMin < 0 \\ 0 & yMin \geq 0 \end{cases}$$

- xOffsetLeft

$$xOffsetLeft_{pixels} = \begin{cases} div(-xMin_{outlineResolution} \times fontSize \times 45, outlineResolution \times 56) & xMin < 0 \\ 0 & xMin \geq 0 \end{cases}$$

outlineResolution is extracted from the PFR, div (A, B) = ceil (A / B), "/" is a rational divide and ceil(A) is the first integral number greater than or equal to A.

In figure 16, *yOffsetTop*, *yOffsetBottom* and *xOffsetLeft* are all greater than or equal to zero and represent the number of available pixels required above, below and to the left of the character origin to prevent clipping of any character in the font. A value *xOffsetRight* could be defined along similar lines but is not relevant to this Profile.

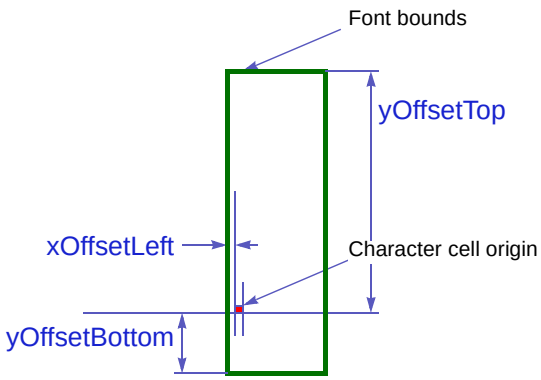



**Figure 16: Font bounds**

### 13.5.4.1 Vertical limits

The origin of any character shall be at least yOffsetTop inside the top edge of the Text object and at least yOffsetBottom inside its bottom edge. Assuming that all characters in a line of text share a common baseline then, **regardless of** VerticalJustification the number of lines of text that may be presented within a text object is:

num_lines = floor((height_Text_object - (yOffsetBottom + yOffsetTop)) / linespace) + 1

All values are in pixels. The variable linespace is an attribute of the Text object that defines the space between the baselines of consecutive lines of text. The function floor(A) rounds A to the first integral number less than or equal to A.

NOTE: Linespace is defined in units of points but as described previously (see clause 13.5.3), these map one-to-one with pixels in the vertical direction.

#### 13.5.4.2 Horizontal limits

The number of characters that may be rendered on a line is not simply dependent upon the width of the Text box and the horizontal escapement for each character, but also needs to consider that the rendering of the first character in a line may extend to the left of its origin. Thus, **regardless of** HorizontalJustification the space available for rendering a line of text within a Text object is:

available_width = width_Text_object - xOffsetLeft

All values are in pixels. available_width may then be used with the "logical" text width rules to determine text flow.

### 13.5.5 "logical" text width rules

To ensure that text will flow identically (i.e. lines and words will break at the same character position) on different receivers and authoring stations, regardless of the quality of the character rendering, a set of "logical" text width rules are defined here.

These rules are a simplification of the rules that might be applied in a typographic rendering system. The objective of these simplifications is to reduce the receiver complexity required to ensure exact correlation of text flow behaviour.

The "logical" text width shall be used in the following cases:

- to determine when to wrap lines of text within a Text object;
- to determine which tab stops text has passed when implementing tab characters.

The calculation of "logical" text width is based on "physical" font data. This data provides a description of the font at a very high resolution, abstracted from any actual rendering system. Consequently, the calculation of the "logical" width of a string of characters involves, computing their width at this high resolution and then converting to units appropriate to the rendering system, e.g. TV pixels, before making decisions about text flow (see clause 13.5.3).

#### 13.5.5.1 Computing "logical" text width

The key parameters when calculating the width of a string of N characters are:

- text font size (one of the values in table 64);
- charSetWidth (carried by the "character record": see clause 13.5.5.1.2);
- the metricsResolution (carried by the "physical font record": see clause 13.5.5.1.2);
- any kerning adjustment (see "pair kerning data" in clause 13.5.5.1.3).

##### 13.5.5.1.1 Font sizes

Font sizes are expressed as the size of an "Em", which broadly speaking, is the minimum distance between the baselines of consecutive lines of text in the given font, in units of "points", which are archaic typographical units. Traditionally there were 72.27 points to an inch; computerized systems now use 72 points per inch for simplicity. If text is 48 point, the Em at that size is 48 points.

##### 13.5.5.1.2 Character widths

The physical font record carries a character record for each character code. This gives the width of each character relative to the size of an Em in metricsResolution units. Therefore, if metrics are specified in 1/1000ths of an Em, a character with a width of 0.6 Em will have a set width of 600.

#### 13.5.5.1.3 Kerning

For certain character combinations (a "kerning pair") a kerning adjustment may also be provided. Typically kerning reduces character spacing for pairs such as AV instead of AV.

The pair kerning data within the pairKernData extra data items in the physical font record defines this adjustment. This defines a baseAdjustment (shared by several character pairs) and an adjustment (for a particular character pair). These provide a signed adjustment to the nominal charSetWidth of the first character.

Like charSetWidth kerning adjustments are in terms of metricsResolution units.

Kerning adjustments only apply between characters, not between the start of a line of text and the edge of the text object.

#### 13.5.5.1.4 Letter spacing

Letterspace (defined in the font attribute of the text object, see clause 13.4.1) allows for an expansion/condensation of the character spacing for all of the characters in a text object.

In printing, this is sometimes referred to as tracking.

### 13.5.5.2 Logical text width

The equation below shows how the width of a string of N characters is computed.

$$\text{logical width of N characters}_{\text{points}} =$$

$$\text{div}((N-1)\times \text{letter space},256) + \text{div}\left(\text{fontsize}\times\left(\sum_{1}^{N}\text{charSetWidth[i]} + \sum_{1}^{N-1}\text{kern[i,i+1]}\right),\text{metricsResolution}\right)$$

$$\text{logical width of N characters}_{\text{pixels}} = \text{div}(\text{logical width of N characters}_{\text{points}}\times 45,56)$$

Where in div( A, B ):

- B is unsigned and A is signed; and
- div( A, B ) = ceil( A / B ).

Where '/' is a rational divide and ceil(A) is the first integral number greater than or equal to A. So, the calculations round up when reducing precision and tend to over estimate the width of text.

The width of pixels is 56/45 points, see clause 13.5.3.

## 13.5.6 Line breaking

### 13.5.6.1 TextWrapping false

Where the TextWrapping attribute of a Text object is set False, text shall break onto a new line only where a Carriage Return character is present in the text. The number of lines to render shall be equal to the number of Carriage Returns present, plus one.

### 13.5.6.2 TextWrapping true

Where the TextWrapping attribute is set to True, the text flow may additionally be broken onto a new line according to the following wrapping rules. The text wrapping behaviour is independent of the Text object's justification settings and is considered to take place before any truncation (see clause 13.5.7).

The effect of the text wrapping rules is to remove breaking characters and insert additional Carriage Returns. (This happens only as far as rendering is concerned; the GetTextData action on a Text object shall return the *actual* text content.)

Firstly, based on the "logical" width of the text, receivers shall determine for each line, the first contiguous sequence of non-breaking characters that:

- would not completely fit within the available width; and
- that follows one or more breaking characters.

If such a sequence exists, the breaking character preceding it shall be replaced with a Carriage Return character.

Secondly, **all trailing** breaking characters shall be discarded from all lines of text. (Preceding breaking characters are not affected.)

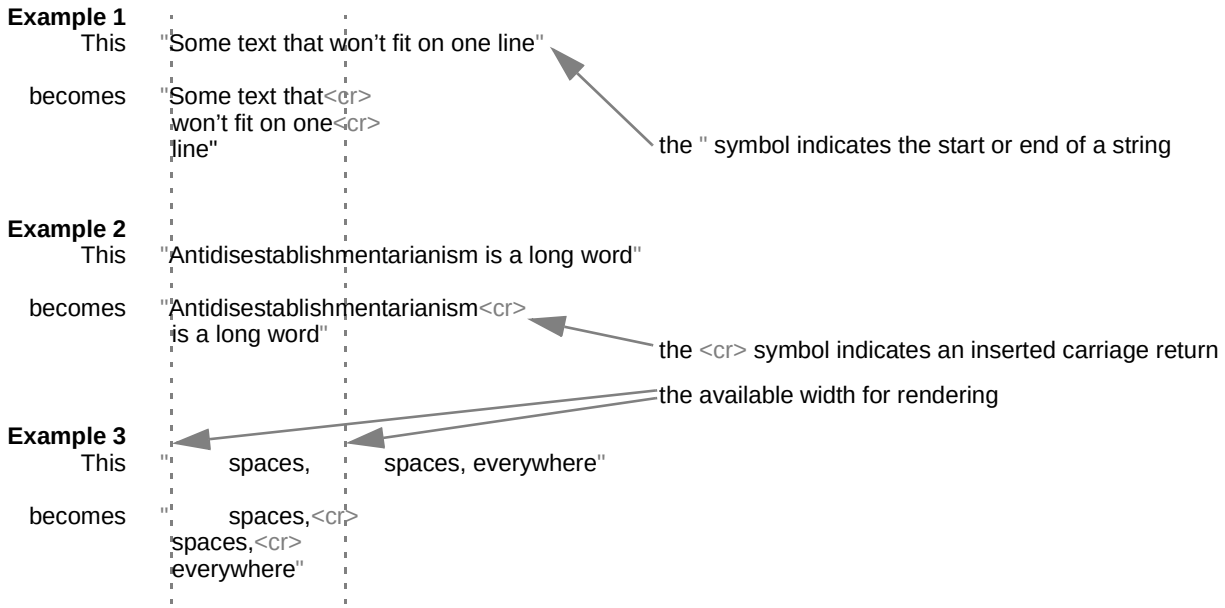



**Figure 17: Text wrapping examples**

After the text wrapping pre-processing is performed, the text truncation rules in clause 13.5.7 still apply. This Profile does not support hyphenation and sequences of non-breaking characters that exceed the available width are not wrapped (see Example 2 in figure 17) but will subsequently be truncated according to clause 13.5.7.

## 13.5.7 Positioning lines of text vertically within the Text object

### 13.5.7.1 Truncation

When the number of lines of text to be rendered exceeds the available height within the text box, lines shall be dropped from the end of the text for "start" and "centre" VJustification settings and from the beginning for "end" VJustification.

The truncation operation will result in a number of lines that can be displayed within the box and these shall be presented according to clause 13.5.7.2. No partial lines shall be displayed.

### 13.5.7.2 Positioning

Vertical measures are illustrated in figure 18. When VerticalJustification = start (top aligned text) the baseline of the first (top most) line shall be yOffsetTop inside the top edge of the Text object and each following line shall be spaced according to the value of linespace.

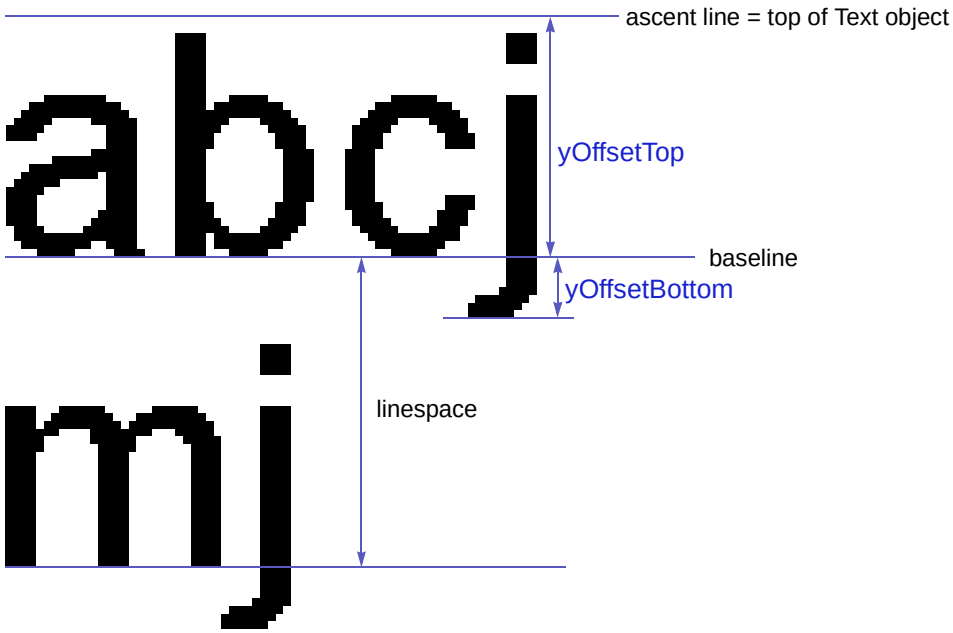



**Figure 18: Vertical measures**

When VerticalJustification = end (bottom aligned text) the origin of the last (bottom most) line shall be yOffsetBottom inside the bottom edge of the Text object and each previous line shall be spaced according to the value of linespace.

When VerticalJustification = centre (vertically centred text), the positioning of the lines shall be such that the space above the first line of text equals the space below the last line of text (to within one pixel) as shown in figure 19:

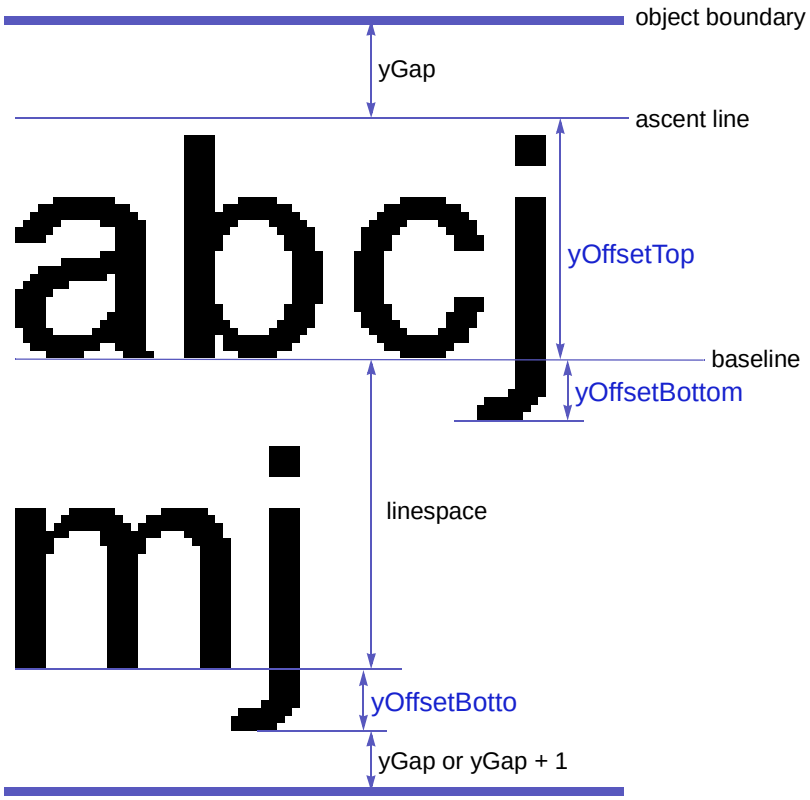



**Figure 19: Vertical measures**

### 13.5.7.3 Examples

See examples in figures 20 and 21.

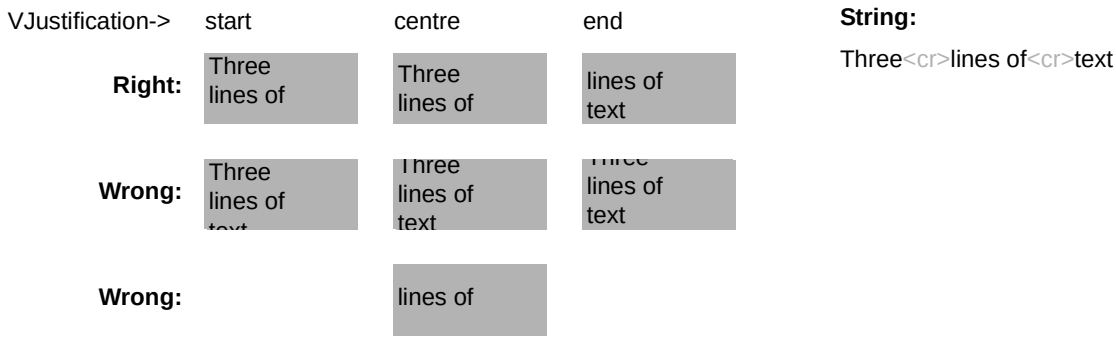



**Figure 20: Vertical positioning example 1**

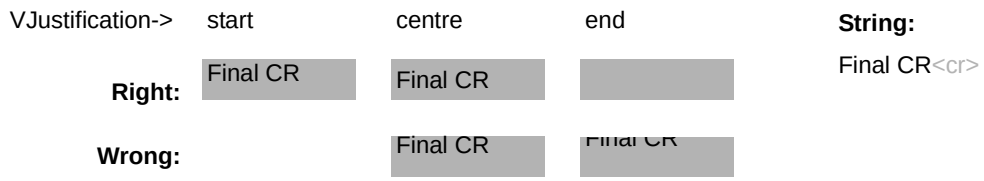



**Figure 21: Vertical positioning example 2**

## 13.5.8 Rendering lines of text horizontally

### 13.5.8.1 Truncation

Where a line of text is too long to fit within the available_width of the Text object (see clause 13.5.4.2), the line shall be truncated. The result shall be as if the complete line were aligned appropriately on the text box (taking into account its HJustification setting) with only those characters whose origin falls to the right of xOffsetLeft and whose right hand edge falls inside the right hand edge of the box, being rendered. Thus, in the "end" HJustification case, a portion of text from the end of the string will be rendered with its right hand edge aligned to the text box. In the "centre" case the centre of the string will appear at the centre of the box with excess characters being dropped from each end.

### 13.5.8.2 Placement

When HorizontalJustification = start (left aligned text) the origin of the first (left most) character shall be xOffsetLeft inside the left edge of the Text object.

When HorizontalJustification = end (right aligned text) the origin of the last (right most) character shall be as necessary to ensure that it is completely visible when rendered.

When HorizontalJustification = centre (horizontally centred text), the positioning of the characters shall be such that the gap to the left of the text equals the gap to the right (to within one pixel). Note that this may place the first character's origin to the left of xOffsetLeft.

### 13.5.8.3 Examples

See figure 22.

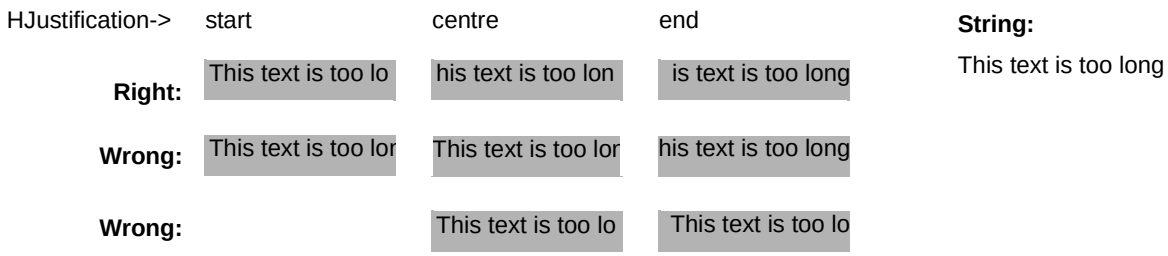



**Figure 22: Horizontal positioning examples**

## 13.5.9 Tabulation

In left aligned text (HorizontalJustification = start) tab stops are defined horizontally every 56 points (45 pixels) from the left edge of the text box. "Horizontal Tabulation" advances the origin of the next character to be rendered to the next tab stop in the direction that the text is currently flowing (the character repertoire in table 78 only requires left to right text). See figure 23.

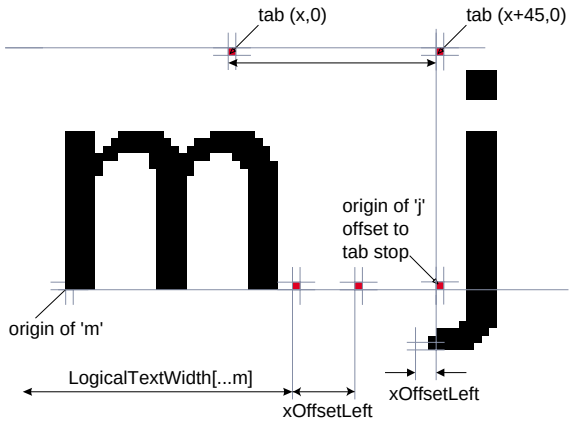



**Figure 23: Effect of horizontal tabulation**

Tab characters only have meaning in left aligned text. If the text is right aligned or centred then tab character shall be treated as a space character.

A tab logically advances the rendering of the text by at least the width xOffsetLeft. If the normal origin of the next character to be rendered after the tab character is after a tab stop, a tab character will advance the rendering to the subsequent tab stop.

The tab stops are at regular intervals from the left edge of the Text object and are not affected by the xOffsetLeft offset to the origin of the first character.

## 13.5.10 Placing runs of characters and words

A run of characters starts from a well-defined point:

- the start edge of the text object (see clause 13.5.4);
- a tab stop.

After this origin the fine positioning of character cells and the gaps between words is not fully specified, as shown in figure 24.

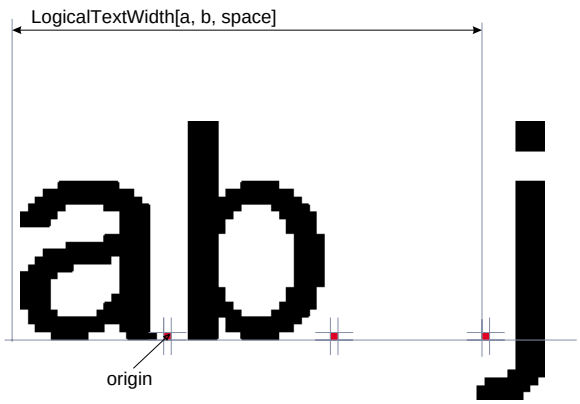



**Figure 24: Calculation of character placement**

However, the following rendering requirements shall be observed to ensure that a minimum acceptable level of text legibility is achieved:

- the spacing between any pair of characters shall "appear" to be consistent wherever that pair of characters is displayed;
- at the default character spacing no two non-whitespace characters shall "appear" to touch;
- the physical rendering of a run of text as determined by the "logical" rules shall be achieved completely within the space used for the "logical" calculation;
- no partially rendered characters shall be presented.

It is worth pointing out that the "logical" position of each character as determined by the "logical" rules is likely to be a non-integer. For renderers unable to support suitable sub-pixel positioning, e.g. a 1-bit/pixel bitmap renderer, this is impossible to implement. Thus whilst the "logical" rules must be used to determine the flow of text, they need not be observed for individual character placement within this flow and other strategies may be employed as long as the requirements above are satisfied.

# 13.6 Text mark-up

## 13.6.1 White space characters

Certain non-printing characters have special meaning. These are identified in table 69.

Table 69 also indicates those characters that may be considered as candidates for line breaking when TextWrapping is enabled.

**Table 69: Special characters**

| UTF8 Value(s) | Character | Name | Breaking/ non-breaking | Meaning |
|---|---|---|---|---|
| 0x09 | 0x0009 | Tab | Breaking | See clause 13.5.9. |
| 0x0D | 0x000D | Carriage return | N/A | Causes the text flow to break. The origin for the next character to be rendered moves to a new baseline "linespace" below that just rendered. The horizontal position of the next line will depend on the horizontal alignment setting. |
| 0x20 | 0x0020 | Space | Breaking | Spaces text by the width defined for the space character. When an object has TextWrapping set to "true" lines may be broken at a space. See clause 13.5.6. |
| 0xC2A0 | 0x00A0 | Non-breaking space | Non-breaking | Identical spacing characteristics to 0x20 but is not seen as word boundary for deciding a position to break a line of text.<br><br>(0xC2A0 is the UTF-8 representation of 0x00A0) |
| 0xE28087 | 0x2007 | Figure space | Non-breaking | Can be used in a string of numerals as an alternative to using comma to denote "thousands". This character is not treated as a word boundary when deciding a position to break a line of text. |

## 13.6.2 Marker characters

The codes 0x1C to 0x1F are zero width, non-spacing, non-printing characters available for use by MHEG-5 authors as markers in text objects, i.e. when using string handling RPs.

### 13.6.3 Non-printing characters

Certain characters (or character sequences) have no immediate visual representation. These include:

- 0x0A Line Feed (LF). Note that the character sequence CRLF shall be rendered identically to a single Carriage Return line breaking character, i.e. the Line Feed is ignored;
- 0x1C to 0x1F marker characters (see clause 13.6.2);
- format control and hypertext mark-up (see clause 13.6.4 and clause 13.8);
- other characters not recognized by the receiver.

When presenting text that includes these characters the character placement shall be as if the non-printing characters were eliminated from the text before rendering. In particular, the character spacing and inter character kerning shall be computed as if the non-printing characters were not present.

### 13.6.4 Format control mark-up

Within text objects mark-up codes can be used to control the presentation of text. The sequence in table 70 marks the start of some marked-up text. For each "start of mark-up" a corresponding "end of mark-up" is defined. The byte sequence for the "end of mark-up" is illustrated in table 71. The minimum number of supported mark-up instances, where each instance is a start and end mark-up pair, is 256. Text object mark-up codes are given in table 72

Engines shall ignore mark-up which does not conform to the present document by skipping past the N + 3 byte mark-up sequence.

**Table 70: General format for start of text mark-up**

| | Bits | Value | Note |
|---|---|---|---|
| `start_of_markup` | 8 | 0x1B | Escape |
| `markup_start_identifier` | 8 | 0x40 to 0x5E | "@" to "^" |
| `parameters_length` | 8 | N | |
| `for( i=0; i<N; i++ ) {` | | | |
| `  parameter_byte` | 8 | 0x00 to 0xFF | |
| `}` | | | |

**Table 71: General format for end of text mark-up**

| | Bits | Value | Note |
|---|---|---|---|
| `end_of_markup` | 8 | 0x1B | Escape |
| `markup_end_identifier` | 8 | 0x60 to 0x7E | " to ~ |

**Table 72: Text object mark-up codes**

| Min. nesting | Start mark-up | End mark-up | Description |
|---|---|---|---|
| | 0x1B 0x42 0x00 | 0x1B 0x62 | Applies "bold" style to the text enclosed (NOTE). |
| 16 | 0x1B 0x43 0x04 0xrr 0xgg 0xbb 0xtt | 0x1B 0x63 | Applies colour to the text enclosed. 0xrr specifies the red intensity, 0xgg the green, 0xbb the blue and 0xtt the transparency. |
| NOTE : Not supported in this Profile | | | |

### 13.6.5 Future compatibility

Compatible extensions to the set of mark-up codes may be defined in future Profiles. For each pair the markup_end_ identifier will be 32 (0x20) greater than its corresponding markup_start_identifier. Engines shall ignore unrecognized mark-up and display any text enclosed within it.

## 13.7 EntryFields

### 13.7.1 Supported characters

All receivers shall provide entry field input for the character repertoires shown in table 73.

**Table 73: Characters supported by EntryFields**

| InputType | Set of characters | Comment |
|---|---|---|
| numeric | 0 - 9 (UTF-8 coded Unicode 0x30 to 0x39) | The UTF encoding for each of these characters is one byte long. |
| alpha | | Not supported |
| any | | |
| listed | | |

### 13.7.2 Appearance

See figure 25.

The entry field is to be presented as a simple single line Text object with the size and position specified. If the text is too big to fit in the EntryField the appearance of any text that overlaps the edge is not defined, the engine shall not provide scrolling or line wrapping.

Since the EntryField inherits from the Text class, its background colour and text colour are defined by the BackgroundColour and TextColour attributes. The usable text area of an EntryField behaves as a Text object inset by 4 pixels on all sides from the BoundingBox.

The 4 pixel area between the BoundingBox and the usable text area shall be filled with the HighlightRefColour when both the HighlightStatus and EngineResp attributes are True and shall be filled with the BackgroundColour otherwise

If the ObscuredInput attribute is set True each character (including any OriginalContent) shall be represented by the asterisk character (0x002A).

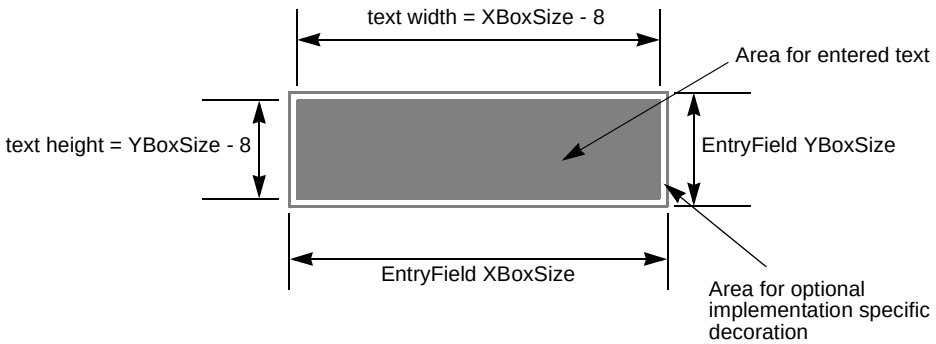



**Figure 25: Area for text in an EntryField**

The shape of the caret is implementation specific but its colour shall be the EntryField's TextColour. The caret shall not affect the flow of text within the entry field.

## 13.7.3 Behaviour

### 13.7.3.1 Character encoding

The OriginalContent attribute shall not include any of the mark-up characters described in clause 13.6.

The OriginalContent may contain characters not permitted for InputType, for example "#".

Values used for EntryField attributes, such as EntryPoint and MaxLength, and associated elementary actions, refer to numbers of characters and not bytes. Also, TextData and all associated elementary actions are to use UTF-8 encoding. In this Profile, however, the set of characters supported all encode to one byte and so the two are usually equivalent. Where this may not be true is if the OriginalContent contains characters outside of the InputType character set, which may encode into more than one byte.

### 13.7.3.2 Semantics of EntryFieldFull and MaxLength

This Profile defines the following additional semantics for the EntryFieldFull event and the MaxLength attribute:

- the EntryFieldFull event will only be triggered when the number of characters in an EntryField goes from less-than MaxLength to MaxLength. If the number of characters is already MaxLength and the viewer attempts to insert additional characters then the event is not triggered;
- the EntryFieldFull event is only triggered as the result of viewer interaction and is never triggered as the result of a SetData action or if OriginalData is greater than MaxLength;
- MaxLength defines the absolute limit to the number of characters that the EntryField will allow in its TextData attribute;
- a MaxLength value less than 0 shall be treated the same as a value of 0;
- the MHEG-5 engine shall not provide feedback to the user to indicate that the field is full; this is the responsibility of the application;
- SetData on an EntryField will truncate the new string if it has more than **MaxLength** characters.

### 13.7.3.3 EntryPoint

In addition to the semantics of EntryPoint laid out in ISO/IEC 13522-5 [14], setting the value of EntryPoint to less than 0 has the same effect as setting it to 0.

### 13.7.3.4 Successive character entry

When an EntryField accepts input characters they are added to the TextData in the normal order for the LineOrientation and StartCorner of the object.

In this Profile the only required LineOrientation and StartCorner are **horizontal** and **upper-left** (i.e. normal Latin script left to right text). Therefore, if the characters "1", "2", "3", and "4" are pressed in that order, the string "1234" shall be added to the TextData.

### 13.7.3.5 Only SetData when inactive

As specified in clause 11.13.6 the engine behaviour is undefined if SetData is targeted at an interactible while its InteractionStatus attribute is set to True.

### 13.7.3.6 User input

When an EntryField object is being interacted with it shall observe the generic behaviour for Interactibles as described in clause 11.13.6. In addition the following key presses have special meaning (see table 19):

- Select simply terminates interaction;
- Cancel terminates interaction and also sets the TextData attribute to an empty string;
- Left results in the character to the left of the insertion point (if any) being deleted regardless of the state of OverwriteMode.

### 13.7.3.7 Numerics of the EntryField

The following interpretation of the EntryField class Interaction behaviour and EntryPoint and OverwriteMode attributes shall be used:

- The insertion point is just before the character referenced by EntryPoint.
- The EntryPoint is the zero-based index of a character.
- After a new character is entered the EntryPoint shall be incremented.
- If the OverwriteMode attribute is set to True any new character replaces the character most recently referenced by the EntryPoint (i.e. the character just after the insertion point is replaced and then the insertion point moves one character space so that it is just after the character just entered).
- If the OverwriteMode attribute is set to False any new character is inserted before the character originally referenced by the EntryPoint.
- During interaction the behaviour is as if there is a non-printing, zero-width, end-of-text character after the last character in the TextData. This character can have the insertion point (i.e. be referenced by the EntryPoint) but cannot be overwritten. In this way the insertion point can be positioned after the last character in the TextData. This conceptual character is never to be considered part of the TextData attribute.

In the examples given in table 74 the initial string is "abc" and the entered string is "123".

**Table 74: EntryField interaction examples**

| OverwriteMode | Initial EntryPoint | Resulting String | Resulting EntryPoint |
|---|---|---|---|
| False | 0 | 123abc | 3 |
| False | 1 | a123bc | 4 |
| False | 3 | abc123 | 6 |
| True | 0 | 123 | 3 |
| True | 1 | a123 | 4 |
| True | 3 | abc123 | 6 |

## 13.8 HyperText

In addition to the mark-up codes identified for Text objects in clause 13.6 HyperText objects can also include the mark-up in table 75. See clause 17.8.

**Table 75: Additional HyperText mark-up codes**

| Start mark-up | End mark-up | Description |
|---|---|---|
| 0x1B 0x41 0xnn tag_bytes | 0x1B 0x61 | Associates the OctetString tag_bytes with the Anchor text enclosed between the start mark-up and the end mark-up.<br>0xnn is the length of the tag_bytes OctetString. |
| 0x1B 0x44 0xnn body_attr_bytes | 0x1B 0x64 | This mark-up is only interpreted if it is the first mark-up in the text. It conveys attributes of the "body" of the hypertext. The coding of these attributes is in table 76.<br>0xnn is the length of the body_attr_bytes OctetString. |

**Table 76: Body attribute encoding**

| | Bits | Type |
|---|---|---|
| '0' | 1 | bslbf |
| anchor_colour_flag | 1 | bslbf |
| active_anchor_colour_flag | 1 | bslbf |
| visited_anchor_colour_flag | 1 | bslbf |
| anchor_wrapping_flag | 1 | bslbf |
| reserved | 3 | bslbf |
| if( anchor_colour_flag == '1' ) { | | |
| anchor_colour | 32 | bslbf |
| } | | |
| if( active_anchor_colour_flag == '1' ) { | | |
| active_anchor_colour | 32 | bslbf |
| } | | |
| if( visited_anchor_colour_flag == '1' ) { | | |
| visited_anchor_colour | 32 | bslbf |
| } | | |
| for( i=0; i<N; i++ ){ | | |
| reserved_byte | 8 | |
| } | | |

Details of body attribute encoding as shown in table 76 are as follows:

### 13.8.1 anchor_wrapping_flag

When set to True this flag indicates that the input focus wraps to the opposite end of the list of anchors when the user attempts to navigate past the boundaries of the object. The top and bottom events shall not be fired.

When this flag is set to False the focus remains on the limiting visible anchor and the top event or bottom event (as appropriate) shall be fired.

### 13.8.2 anchor_colour

The 32-bit integer in this field, if present, specifies the colour for unvisited hypertext anchors. If this field is not present then the colour for unvisited hypertext anchors is the default given in clause 13.8.7.2.

The colour encoding is shown in figure 11.

### 13.8.3 active_anchor_colour

The 32-bit integer in this field, if present, specifies the suggested colour for hypertext anchors when they have the user focus.

Support for this field is optional since the receiver may have an alternative means of representing active anchors. However, if text colour is used to represent that an anchor has the focus then the colour in this field shall be used if present. If this field is not present then the colour for unvisited hypertext anchors is the default given in clause 13.8.7.2.

The colour encoding is shown in figure 11.

### 13.8.4 visited_anchor_colour

The 32-bit integer in this field, if present, specifies the suggested colour for previously visited hypertext anchors.

This field shall be ignored even if present and visited anchors shall appear the same colour as unvisited ones (see anchor_colour in table 76).

### 13.8.5 reserved_byte

Bytes reserved for future use.

## 13.8.6 HyperText anchors

After the application transfers the focus of user interaction to a HyperText object (by using the SetInteractionStatus action) the engine is responsible for managing interaction with the HyperText object.

The OctetString tag_bytes is the "tag" of the anchor as defined by MHEG-5. When the anchor "fires" the tag_bytes are returned as the associated data of the AnchorFired event and are placed in the LastAnchorFired internal attribute.

The engine is responsible for:

- the movement of the user focus amongst the anchors in the HyperText in response to user input;
- providing feedback to the user as their focus moves;
- allowing the user to select an anchor.

## 13.8.7 Appearance

### 13.8.7.1 Visual appearance of anchors

Anchor markup shall have a higher priority than any other visual markup. That is to say that the visual representation for an anchor shall override any other visual parameters which may be in force at that point in the text. For example if an anchor appeared in a section of text which had its colour changed with the colour change mark-up the anchor would still appear in the colours defined for an anchor.

### 13.8.7.2 Default anchor colours

See table 77.

**Table 77: Default anchor colours**

| | |
|---|---|
| anchor_colour | 0x0000FF00 (saturated blue) |
| active_anchor_colour | 0xFF000000 (saturated red) |

### 13.8.7.3 Highlight

The HighlightStatus attribute shall not affect the visible appearance of HyperText objects in this Profile.

## 13.8.8 Behaviour

The exact behaviour during interaction is dependent upon the input device being used. Principally the user **identifies** and **selects** anchors. The present document specifies behaviour for an input device offering up, down, left, right and select actions. Behaviour for input devices not conforming to this pattern will be considered as the need arises.

### 13.8.8.1 Anchor identification

The user identifies an anchor by moving the "focus" between anchors by means of a sequence of direction key navigational steps. The "up" or "left" user interface functions will move the focus to the previous anchor. The "down" or "right" user interface functions will move the focus to the next anchor. The focus can move to anchors that are not currently visible on screen in this way.

### 13.8.8.2 Behaviour

When a Hypertext object is being interacted with it shall observe the generic behaviour for Interactibles as described in clause 11.13.6. In addition:

- each time focus is moved it generates a FocusMoved event. In the case where the focus is at a boundary, the user attempts to move across that boundary and focus wrapping is switched off this event is not generated;
- anchors shall be fired by the user activating the "Select" user interface function;
- interaction with the HyperText object shall be terminated if the user activates the "Cancel" user interface function.

### 13.8.8.3 Special behaviour at boundaries

NOTE: This clause addresses the case when the anchor_wrapping_flag is set to False.

In addition to generating anchor fired events when the user selects an anchor in the hypertext, the engine shall also generate these events if the user tries to navigate past the first or last anchors when the anchor_wrapping_flag is set to "0".

- top event

  When the user focus is on the first (i.e. "top") anchor in the hypertext, and the user navigation direction is "up" (up or left arrow key) then an AnchorFired event shall be generated by the HyperText object with EventData "rec://htext/top";

- bottom event

  When the user focus is on the last (i.e. "bottom") anchor in the hypertext, and the user navigation direction is "down" (down or right arrow key) then an AnchorFired event shall be generated by the HyperText object with EventData "rec://htext/bot";

- common behaviour

  The visual feedback of the location of the user focus, and the logical position of the user focus, shall remain on the anchor just inside the boundary after a user navigation that tries to cross the boundary. The top event or bottom event (as appropriate) shall be fired instead of the FocusMoved event.

  The application author is responsible for appropriately "using" the "boundary" event. For example, a Link might be provided to set the interaction status of the HyperText object to False and to then highlight some other Interactible.

  If the interaction status of the hypertext object is still true after the "boundary" event has fired (i.e. no Link was active for the event, or its effect did not set the InteractionStatus of the HyperText object to False) the HyperText object continues to operate with the user focus remaining on its last position.

  These events shall be generated even if there are no anchors in the text. In this case pressing up/left while the HyperText object has InteractionStatus set to True shall generate the top event and pressing down/right shall generate the bottom event.

Therefore, if the user navigation direction is "up" (up or left arrow key) when the focus is on the top most anchor there is no visual feedback unless specifically authored into the application. A subsequent "down" (down or right arrow key) user input will cause the user focus to move to the next anchor within the HyperText object unless the application has used the "boundary" event to stop interaction with the HyperText.

Where the navigation method of the engine does not use 2D navigation of the focus between anchors, and hence does not use navigation concepts such as "up" and "down" an alternative method shall be provided to ensure that the "rec://htext/top" (or "rec://htext/bot" as appropriate) event is generated as the result of user navigation away from the object. For example, an engine using cursor/mouse based interaction might generate an AnchorFired event with EventData "rec://htext/top" when the cursor leaves the upper half of the bounding box of the HyperText object and "rec://htext/top" when the cursor leaves the lower half of the bounding box.

# 13.9 Slider

## 13.9.1 Appearance

Visually the Slider shall be implemented as a rectangular "Thumb" within an invisible "Guide", i.e. there shall be no background to the Slider. This allows application developers to implement a slider function with an application-specific background.

The Thumb shall be a rectangular area, the colour of which is defined by the SliderRefColour attribute.

The "width" of the Thumb shall be the size of the Slider object in the dimension perpendicular to the axis defined by the Orientation attribute.

The "length" of the Thumb is its dimension along the axis defined by the Orientation attribute. The Thumb's position and length will vary depending upon the SliderStyle attribute as follows:

- If the SliderStyle is set to normal, the Thumb is rendered as a 9 pixel long «marker» which is centred on the position on the «main axis» corresponding to the SliderValue attribute. At the extremes of SliderValue the Thumb shall remain within the bounding box of the Slider and still be rendered completely.
- If the SliderStyle is set to thermometer, the Thumb's position and length are as defined in the SliderStyle attribute in ISO/IEC 13522-5 [14].
- If the SliderStyle is set to proportional, the Thumb's position and length are as defined in the SliderStyle attribute in ISO/IEC 13522-5 [14].

The Slider shall be highlighted if both the HighlightStatus and EngineResp attributes are True. The appearance of this highlight shall be to change the colour of the rectangular area representing the Thumb to be that defined by the HighlightRefColour.

## 13.9.2 Behaviour

When a Slider object is being interacted with it shall observe the generic behaviour for Interactibles as described in clause 11.13.6. In addition:

- The SliderValue attribute can be modified using user input functions as follows:
  - If the Orientation attribute is set to "left" then the "left" and "right" user input functions shall respectively increase and decrease the SliderValue by the value of the StepSize attribute.
  - If the Orientation attribute is set to "right" then the "right" and "left" user input functions shall respectively increase and decrease the SliderValue by the value of the StepSize attribute.
  - If the Orientation attribute is set to "up" then the "up" and "down" user input functions shall respectively increase and decrease the SliderValue by the value of the StepSize attribute.
  - If the Orientation attribute is set to "down" then the "down" and "up" user input functions shall respectively increase and decrease the SliderValue by the value of the StepSize attribute.
- Interaction with the Slider object shall be terminated if the user activates either the "Select" or "Cancel" user interface function.

## 13.10 Character repertoire

Table 78 lists the characters that shall be implemented by all decoders conforming to *ETSIEngineProfile1*. The characters in the shaded cells (□) are shown only to emphasize gaps. They are not part of the supported character set and shall be ignored by receivers. See clause 13.6.3.

**Table 78: Set of characters supported by the engine (Sheet 1 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 0000 | 00 | | Non-printing control codes |
| to | | | |
| 0008 | 08 | | |
| 0009 | 09 | | Horizontal tabulation |
| 000A | 0A | | Non-printing control codes |
| to | | | |
| 000C | 0C | | |
| 000D | 0D | | Carriage return |
| 000E | 0E | | Non-printing control codes |
| to | | | |
| 001B | 1B | | |
| 001C | 1C | | Zero width, non-spacing, non-printing characters available for use as markers |
| to | | | |
| 001F | 1F | | |
| 0020 | 20 | | Space |
| 0021 | 21 | ! | Exclamation mark |
| 0022 | 22 | " | Quotation mark |
| 0023 | 23 | # | Number sign |
| 0024 | 24 | $ | Dollar sign |
| 0025 | 25 | % | Percent sign |
| 0026 | 26 | & | Ampersand |
| 0027 | 27 | ' | Apostrophe |
| 0028 | 28 | ( | Left parenthesis |
| 0029 | 29 | ) | Right parenthesis |
| 002A | 2A | * | Asterisk |
| 002B | 2B | + | Plus sign |
| 002C | 2C | , | Comma |
| 002D | 2D | - | Hyphen-minus |
| 002E | 2E | . | Full stop |
| 002F | 2F | / | Solidus |
| 0030 | 30 | 0 | Digit zero |
| 0031 | 31 | 1 | Digit one (NOTE 1) |
| 0032 | 32 | 2 | Digit two (NOTE 1) |
| 0033 | 33 | 3 | Digit three (NOTE 1) |
| 0034 | 34 | 4 | Digit four (NOTE 1) |
| 0035 | 35 | 5 | Digit five (NOTE 1) |
| 0036 | 36 | 6 | Digit six (NOTE 1) |
| 0037 | 37 | 7 | Digit seven (NOTE 1) |
| 0038 | 38 | 8 | Digit eight (NOTE 1) |
| 0039 | 39 | 9 | Digit nine (NOTE 1) |

**Table 78: Set of characters supported by the engine (Sheet 2 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 003A | 3A | : | Colon |
| 003B | 3B | ; | Semicolon |
| 003C | 3C | < | Less-than sign |
| 003D | 3D | = | Equals sign |
| 003E | 3E | > | Greater-than sign |
| 003F | 3F | ? | Question mark |
| 0040 | 40 | @ | Commercial at |
| 0041 | 41 | A | Latin capital letter A |
| 0042 | 42 | B | Latin capital letter B |
| 0043 | 43 | C | Latin capital letter C |
| 0044 | 44 | D | Latin capital letter D |
| 0045 | 45 | E | Latin capital letter E |
| 0046 | 46 | F | Latin capital letter F |
| 0047 | 47 | G | Latin capital letter G |
| 0048 | 48 | H | Latin capital letter H |
| 0049 | 49 | I | Latin capital letter I |
| 004A | 4A | J | Latin capital letter J |
| 004B | 4B | K | Latin capital letter K |
| 004C | 4C | L | Latin capital letter L |
| 004D | 4D | M | Latin capital letter M |
| 004E | 4E | N | Latin capital letter N |
| 004F | 4F | O | Latin capital letter O |
| 0050 | 50 | P | Latin capital letter P |
| 0051 | 51 | Q | Latin capital letter Q |
| 0052 | 52 | R | Latin capital letter R |
| 0053 | 53 | S | Latin capital letter S |
| 0054 | 54 | T | Latin capital letter T |
| 0055 | 55 | U | Latin capital letter U |
| 0056 | 56 | V | Latin capital letter V |
| 0057 | 57 | W | Latin capital letter W |
| 0058 | 58 | X | Latin capital letter X |
| 0059 | 59 | Y | Latin capital letter Y |
| 005A | 5A | Z | Latin capital letter Z |
| 005B | 5B | [ | Left square bracket |
| 005C | 5C | \ | Reverse solidus |
| 005D | 5D | ] | Right square bracket |
| 005E | 5E | ^ | Circumflex accent |
| 005F | 5F | _ | Low line |
| 0060 | 60 | ` | Grave accent |
| 0061 | 61 | a | Latin small letter A |
| 0062 | 62 | b | Latin small letter B |
| 0063 | 63 | c | Latin small letter C |
| 0064 | 64 | d | Latin small letter D |
| 0065 | 65 | e | Latin small letter E |
| 0066 | 66 | f | Latin small letter F |

**Table 78: Set of characters supported by the engine (Sheet 3 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 0067 | 67 | g | Latin small letter G |
| 0068 | 68 | h | Latin small letter H |
| 0069 | 69 | i | Latin small letter I |
| 006A | 6A | j | Latin small letter J |
| 006B | 6B | k | Latin small letter K |
| 006C | 6C | l | Latin small letter L |
| 006D | 6D | m | Latin small letter M |
| 006E | 6E | n | Latin small letter N |
| 006F | 6F | o | Latin small letter O |
| 0070 | 70 | p | Latin small letter P |
| 0071 | 71 | q | Latin small letter Q |
| 0072 | 72 | r | Latin small letter R |
| 0073 | 73 | s | Latin small letter S |
| 0074 | 74 | t | Latin small letter T |
| 0075 | 75 | u | Latin small letter U |
| 0076 | 76 | v | Latin small letter V |
| 0077 | 77 | w | Latin small letter W |
| 0078 | 78 | x | Latin small letter X |
| 0079 | 79 | y | Latin small letter Y |
| 007A | 7A | z | Latin small letter Z |
| 007B | 7B | { | Left curly bracket |
| 007C | 7C | \| | Vertical line |
| 007D | 7D | } | Right curly bracket |
| 007E | 7E | ~ | Tilde |
| 007F | 7F | | Non-printing control codes |
| to | | | |
| 009F | C29F | | |
| 00A0 | C2A0 | þ | No-break space |
| 00A1 | C2A1 | ¡ | Inverted exclamation mark |
| 00A2 | C2A2 | ¢ | Cent sign |
| 00A3 | C2A3 | £ | Pound sign |
| 00A4 | C2A4 | ¤ | Currency sign |
| 00A5 | C2A5 | ¥ | Yen sign |
| 00A6 | C2A6 | ¦ | Broken bar |
| 00A7 | C2A7 | § | Section sign |
| 00A8 | C2A8 | ¨ | Diaeresis |
| 00A9 | C2A9 | © | Copyright sign |
| 00AA | C2AA | ª | Feminine ordinal indicator |
| 00AB | C2AB | « | Left-pointing double-angle quotation mark |
| 00AC | C2AC | ¬ | Not sign |
| 00AD | C2AD | - | Soft hyphen |
| 00AE | C2AE | ® | Registered sign |
| 00AF | C2AF | ¯ | Macron |
| 00B0 | C2B0 | ° | Degree sign |
| 00B1 | C2B1 | ± | Plus-minus sign |

**Table 78: Set of characters supported by the engine (Sheet 4 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 00B2 | C2B2 | 2 | Superscript two |
| 00B3 | C2B3 | 3 | Superscript three |
| 00B4 | C2B4 | ´ | Acute accent |
| 00B5 | C2B5 | µ | Micro sign |
| 00B6 | C2B6 | ¶ | Pilcrow sign |
| 00B7 | C2B7 | · | Middle dot |
| 00B8 | C2B8 | ¸ | Cedilla |
| 00B9 | C2B9 | 1 | Superscript one |
| 00BA | C2BA | º | Masculine ordinal indicator |
| 00BB | C2BB | » | Right-pointing double-angle quotation mark |
| 00BC | C2BC | ¼ | Vulgar fraction one quarter |
| 00BD | C2BD | ½ | Vulgar fraction one half |
| 00BE | C2BE | ¾ | Vulgar fraction three quarters |
| 00BF | C2BF | ¿ | Inverted question mark |
| 00C0 | C380 | À | Latin capital letter A with grave |
| 00C1 | C381 | Á | Latin capital letter A with acute |
| 00C2 | C382 | Â | Latin capital letter A with circumflex |
| 00C3 | C383 | Ã | Latin capital letter A with tilde |
| 00C4 | C384 | Ä | Latin capital letter A with diaeresis |
| 00C5 | C385 | Å | Latin capital letter A with ring above |
| 00C6 | C386 | Æ | Latin capital letter AE |
| 00C7 | C387 | Ç | Latin capital letter C with cedilla |
| 00C8 | C388 | È | Latin capital letter E with grave |
| 00C9 | C389 | É | Latin capital letter E with acute |
| 00CA | C38A | Ê | Latin capital letter E with circumflex |
| 00CB | C38B | Ë | Latin capital letter E with diaeresis |
| 00CC | C38C | Ì | Latin capital letter I with grave |
| 00CD | C38D | Í | Latin capital letter I with acute |
| 00CE | C38E | Î | Latin capital letter I with circumflex |
| 00CF | C38F | Ï | Latin capital letter I with diaeresis |
| 00D0 | C390 | Ð | Latin capital letter Eth |
| 00D1 | C391 | Ñ | Latin capital letter N with tilde |
| 00D2 | C392 | Ò | Latin capital letter O with grave |
| 00D3 | C393 | Ó | Latin capital letter O with acute |
| 00D4 | C394 | Ô | Latin capital letter O with circumflex |
| 00D5 | C395 | Õ | Latin capital letter O with tilde |
| 00D6 | C396 | Ö | Latin capital letter O with diaeresis |
| 00D7 | C397 | ¥ | Multiplication sign (NOTE 1) |
| 00D8 | C398 | Ø | Latin capital letter O with stroke |
| 00D9 | C399 | Ù | Latin capital letter U with grave |
| 00DA | C39A | Ú | Latin capital letter U with acute |
| 00DB | C39B | Û | Latin capital letter U with circumflex |
| 00DC | C39C | Ü | Latin capital letter U with diaeresis |
| 00DD | C39D | | Latin capital letter Y with acute |

**Table 78: Set of characters supported by the engine (Sheet 5 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 00DE | C39E | Þ | Latin capital letter Thorn |
| 00DF | C39F | ß | Latin small letter Sharp S |
| 00E0 | C3A0 | à | Latin small letter A with grave |
| 00E1 | C3A1 | á | Latin small letter A with acute |
| 00E2 | C3A2 | â | Latin small letter A with circumflex |
| 00E3 | C3A3 | ã | Latin small letter A with tilde |
| 00E4 | C3A4 | ä | Latin small letter A with diaeresis |
| 00E5 | C3A5 | å | Latin small letter A with ring above |
| 00E6 | C3A6 | æ | Latin small letter AE |
| 00E7 | C3A7 | ç | Latin small letter C with cedilla |
| 00E8 | C3A8 | è | Latin small letter E with grave |
| 00E9 | C3A9 | é | Latin small letter E with acute |
| 00EA | C3AA | ê | Latin small letter E with circumflex |
| 00EB | C3AB | ë | Latin small letter E with diaeresis |
| 00EC | C3AC | ì | Latin small letter I with grave |
| 00ED | C3AD | í | Latin small letter I with acute |
| 00EE | C3AE | î | Latin small letter I with circumflex |
| 00EF | C3AF | ï | Latin small letter I with diaeresis |
| 00F0 | C3B0 | ð | Latin small letter Eth |
| 00F1 | C3B1 | ñ | Latin small letter N with tilde |
| 00F2 | C3B2 | ò | Latin small letter O with grave |
| 00F3 | C3B3 | ó | Latin small letter O with acute |
| 00F4 | C3B4 | ô | Latin small letter O with circumflex |
| 00F5 | C3B5 | õ | Latin small letter O with tilde |
| 00F6 | C3B6 | ö | Latin small letter O with diaeresis |
| 00F7 | C3B7 | ÷ | Division sign (NOTE 1) |
| 00F8 | C3B8 | ø | Latin small letter O with stroke |
| 00F9 | C3B9 | ù | Latin small letter U with grave |
| 00FA | C3BA | ú | Latin small letter U with acute |
| 00FB | C3BB | û | Latin small letter U with circumflex |
| 00FC | C3BC | ü | Latin small letter U with diaeresis |
| 00FD | C3BD | ý | Latin small letter Y with acute |
| 00FE | C3BE | þ | Latin small letter Thorn |
| 00FF | C3BF | ÿ | Latin small letter Y with diaeresis |
| 0131 | C4B1 | | Latin small letter dotless I |
| 0152 | C592 | Œ | Latin capital ligature OE |
| 0153 | C593 | œ | Latin small ligature OE |
| 0174 | C5B4 | Ŵ | Latin capital letter W with circumflex |
| 0175 | C5B5 | ŵ | Latin small letter W with circumflex |
| 0176 | C5B6 | Ŷ | Latin capital letter Y with circumflex |
| 0177 | C5B7 | ŷ | Latin small letter Y with circumflex |
| 0178 | C5B8 | Ÿ | Latin capital letter Y with diaeresis |
| 066B | D9AB | . | Decimal separator (NOTE 1) |

**Table 78: Set of characters supported by the engine (Sheet 6 of 6)**

| UCS2 | UTF8 | Glyph | Unicode name for character |
|---|---|---|---|
| 1E80 | E1BA80 | Ẁ | Latin capital letter W with grave |
| 1E81 | E1BA81 | ẁ | Latin small letter W with grave |
| 1E82 | E1BA82 | Ẃ | Latin capital letter W with acute |
| 1E83 | E1BA83 | ẃ | Latin small letter W with acute |
| 1E84 | E1BA84 | Ẅ | Latin capital letter W with diaeresis |
| 1E85 | E1BA85 | ẅ | Latin small letter W with diaeresis |
| 1EF2 | E1BBB2 | Ỳ | Latin capital letter Y with grave |
| 1EF3 | E1BBB3 | ỳ | Latin small letter Y with grave |
| 2007 | E28087 | | Figure space (NOTE 1) |
| 2013 | E28093 | – | En dash |
| 2014 | E28094 | — | Em dash |
| 2018 | E28098 | ‘ | Left single quotation mark |
| 2019 | E28099 | ’ | Right single quotation mark |
| 201A | E2809A | ‚ | Single low-9 quotation mark |
| 201C | E2809C | “ | Left double quotation mark |
| 201D | E2809D | ” | Right double quotation mark |
| 201E | E2809E | „ | Double low-9 quotation mark |
| 2022 | E280A2 | • | Bullet |
| 2026 | E280A6 | … | Horizontal ellipsis |
| 2030 | E280B0 | ‰ | Per mille |
| 2039 | E280B9 | < | Single left-pointing angle quotation mark |
| 203A | E280BA | > | Single right-pointing angle quotation mark |
| 2044 | E28184 | / | Fraction slash |
| 20AC | E282AC | € | Euro-currency sign |
| 2122 | E284A2 | ™ | Trade-mark sign |
| 2190 | E28690 | ← | Leftwards arrow |
| 2191 | E28691 | ↑ | Upwards arrow |
| 2192 | E28692 | → | Rightwards arrow |
| 2193 | E28693 | ↓ | Downwards arrow |
| 2212 | E28892 | - | Minus sign (NOTE 1) |
| 2214 | E28894 | + | Plus (monospaced) (NOTE 1) |
| 2215 | E28895 | / | Division slash (NOTE 1) |
| 221E | E2889E | • | Infinity |
| 266B | E299AB | ♫ | Beamed eighth notes |
| 2713 | E29C93 | ✓ | Check mark |
| 2717 | E29C97 | ✗ | Ballot X |
| NOTE 1: Has the same width as the digit zero (0x0030) | | | |

# 14 Receiver requirements

## 14.1 Introduction

This clause describes the measures that shall be taken by the receiver to ensure good application behaviour.

## 14.2 Management of stream decoders

### 14.2.1 Application killed by receiver

A number of scenarios can result in an application being killed by the receiver (see clause 8.1.1). In this situation the receiver shall set all stream decoders to decode the default components for the default service as determined by the receiver and set their presentation (including any video scaling and/or offset, and audio volume control) to the default state. The receiver shall ensure that the presentation of any stream components selected under application control ends before removal of the OSD resource from the application.

#### 14.2.1.1 On change of service

This is a special case of the above, caused by the receiver executing a tune to a new service, i.e. if the user interacts with the receiver's navigator functions to change channel the video, audio and subtitle decoders are all stopped and then restarted on the new service.

### 14.2.2 Effect of lockscreen

The continued rendering of any active Visible Stream components, i.e. Video, shall not be affected by the LockScreen elementary action and the displayed image shall continue to change in response to data delivered by the referenced stream.

NOTE: Clause 10.1.3 of ISO/IEC 13522-5 [14] says "*…the updating of the graphical presentation of these objects during locked screen is optional. On some engines their physical rendering continues running, on some others the image is stopped until the screen is unlocked.*"

This Profile requires that the physical rendering continues running.

### 14.2.3 Stream inheritance on Application object activation

A launched application may inherit the streams as previously defined. Whether this is an auto-boot application that inherits from the default service settings or a launched/spawned application that inherits from the previous application, the behaviour is the same.

When an MHEG-5 application is launched the Video, Audio and DVB Subtitling stream decoders shall continue operating in their current state until the Application object's activation phase is complete.

Once the Application object is activated, stream decoders will only continue running if there are initially active MHEG-5 Video and/or Audio objects within an initially active MHEG-5 Stream object or objects. This includes the case where the MHEG-5 Stream reference is to a DSM-CC Stream object.

If the receiver determines that the application is trying to continue playing any or all of the stream components that were active when it was launched, there shall be no disruption to the decoding and presentation of these streams.

### 14.2.4 Stream continuance on Application object deactivation

When an MHEG-5 Application is deactivated all other objects, including any active Stream objects, are also deactivated. Clause 37.3 of ISO/IEC 13522-5 [14] describes the deactivation behaviour for Stream objects. In *ETSIEngineProfile1*, Stream objects with the Storage attribute set to memory observe this requirement.

If the Stream object is deactivated following Quit, Spawn or Launch actions, the following shall apply:

- If the Stream object has the Storage attribute set to Memory, observe the standard behaviour.
- If the Stream object has the Storage attribute set to Stream, override this behaviour. Specifically, the stream decoders associated with a Stream object do not automatically stop, and audio, video and subtitles (as relevant) will continue to be presented. Furthermore control over the presentation of these streams (including any video scaling and/or offset, and audio volume control) shall also remain unchanged. See also clause 14.8.

NOTE: This deactivation behaviour is irrelevant if the application is being killed by the receiver (see clause 14.2.1).

## 14.2.5 Locating components carried in Transport Streams

NOTE: This is only relevant for Stream objects with the Storage attribute set to Stream.

The MHEG-5 Stream class identifies streams using a two-step process:

1) Identify a "multiplex" via the Content internal attribute.

   The MHEG-5 term "multiplex" is used in a generic sense (i.e. a collection of elementary streams). **In this Profile** it corresponds to a DVB Service / MPEG program, and should not be confused with other uses of the term such as a DTT multiplex (which is an MPEG transport stream).

   There are three ways of defining a reference to a multiplex. In all cases the data resolves to identifying a service via the triple: original_network_id, transport_stream_id and service_id.

2) Identify individual components within this "multiplex" via the ComponentTag attribute.

### 14.2.5.1 Multiplex references

#### 14.2.5.1.1 DSM-CC Stream object

In this mechanism the OrigContent attribute identifies a DSM-CC Stream object. For this Profile the receiver is only required to support taps in the Stream object of use BIOP_PROGRAM_USE. This kind of tap maps the "multiplex" onto a single DVB Service. See table 104.

#### 14.2.5.1.2 URL explicit format

In this mechanism the OrigContent attribute contains a URL, as described in clause 16.3.3. This maps the "multiplex" onto a single DVB Service.

Although most of this functionality can be provided by the DSM-CC Stream object, it allows references to be built into the application. Also it is the same notation as used for the SI_TuneIndex Resident Program so provides a degree of consistency, i.e. the same data can be used both to preview and then to tune to a service.

#### 14.2.5.1.3 URL inheritance formats

In this mechanism the OrigContent attribute contains one of two forms of URL:

- the OctetString "rec://svc/def" maps the "multiplex" to the service most recently tuned to. For example, by the receiver's built-in navigator or the SI_TuneIndex ResidentProgram; or
- the OctetString "rec://svc/cur" maps the "multiplex" to the service currently being received. If the components are from more than one service, then the order of priority of mapping shall be video then audio.

  The service referenced by "rec://svc/cur" may be different from that referenced by "rec://svc/def" if, since the last service tune, an MHEG-5 application has set the service being decoded using a Stream object with OrigContent specifying either a service URL or a reference to a DSM-CC Stream object.

  The services referenced by "rec://svc/cur" and "rec://svc/def" will be the same if service selection was done by the receiver's built-in navigator or the SI_TuneIndex ResidentProgram.

See also clause 16.1.

#### 14.2.5.1.4 Services in other transport streams

In this Profile all forms of multiplex reference are constrained to refer to services in the same transport stream as that delivering the current object carousel. Selection of services in other transport streams is supported by SI_TuneIndex.

#### 14.2.5.1.5 StreamEvent and CounterTrigger event

Streams shall only support StreamEvents when referenced as a DSM-CC Stream object, and where the contained events are "do-it-now" events.

Because an NPT time base is not supported in this Profile MHEG-5 CounterTrigger events and DSM-CC scheduled events are not supported.

See clause 15.2.4.5 and clause 11.13.2.

### 14.2.5.2 Component references

The mapping of ComponentTag values to service components is described in clause 15.3.

EXAMPLE 1: An application object might include:

```
{:Stream 1
   :OrigContent :ContentRef( "rec://svc/cur" )
   :Shared True
   :Multiplex ( {:Audio 2 :ComponentTag -1 } )
}
```

This allows the current service's default audio to continue without disruption. However, if present in the service video and subtitles will stop at the end of application object preparation.

## 14.2.6 Locating components carried in an Elementary Stream

NOTE: This is only relevant for Stream objects with the Storage attribute set to Memory.

The stream is identified via the Content internal attribute, which provides a reference to the relevant file. The ComponentTag attribute shall be ignored.

EXAMPLE 2: An application object might include:

```
{:Stream 117
      :InitiallyActive false
      :CHook 11
      :OrigContent :ContentRef('/sound1')
      :Shared true
      :Multiplex
      (
            {:Audio 118
               :ComponentTag 2 //this tag value is ignored
            }
      )
      :Storage memory // must be this type
}
```

# 14.3 Application interaction with user control of linear content decoders

This clause addresses how the presentation of linear components (video, audio and subtitles) is affected by viewer and application controls. The figures in this clause define a logical model for each component type on the basis that an application is running. Consequently actual implementation may vary.

### 14.3.1 Video Decoder

See figure 26.

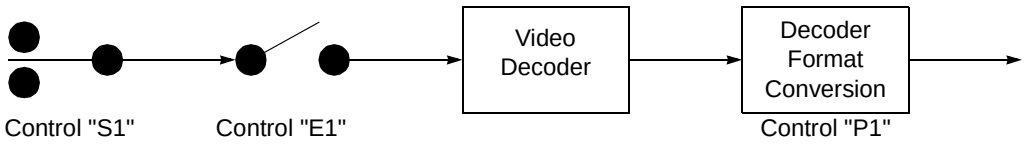



**Figure 26: Video control - logical model**

#### 14.3.1.1 Enabling controls

E1 is the application's control over whether or not video is displayed. It is achieved by the presence of an active MHEG-5 Video object within an active MHEG-5 Stream object. See clause 14.2.3.

#### 14.3.1.2 Selection controls

S1 is the selection of a video component from broadcast. This selection is based on the attributes of the MHEG-5 Stream and Video objects identified above.

#### 14.3.1.3 Presentation controls

P1 represents processing to control the scaling and positioning of the decoded video. It is based a number of factors including (but not exclusively) viewer preferences and application signalling. See clause 14.5.

### 14.3.2 Audio decoder

See figure 27.

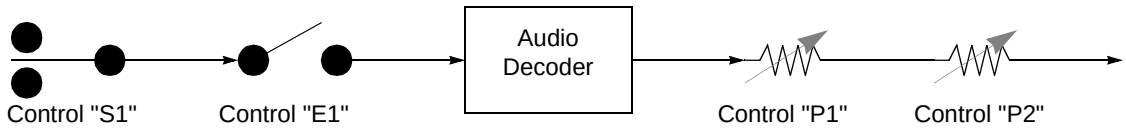



**Figure 27: Audio control - logical model**

#### 14.3.2.1 Enabling controls

E1 is the application's control over whether or not audio is presented. It is achieved by the presence of an active MHEG-5 Audio object within an active MHEG-5 Stream object. See clause 14.2.3.

#### 14.3.2.2 Selection controls

S1 is the selection of an audio component from broadcast or from memory. This selection is based on the attributes of the MHEG-5 Stream and Audio objects identified above, and viewer preferences e.g. language.

#### 14.3.2.3 Presentation controls

P1 is the application's control of the audio volume level. In *ETSIEngineProfile1* this control is only required to be on/off. The volume adjustment defined by an MHEG-5 Audio object is measured in dB and shall be interpreted as follows:

- 0 dB means leave the volume unchanged;
- <-256 dB means mute;
- >0 dB shall be implemented as 0 dB or louder: it may be approximated as 0 dB;
- <0 dB shall be implemented as quieter than 0 dB: it may be approximated as mute.

P2 is the viewer's control of audio volume level. Typically it will be operated by the "Mute", "Vol+", and "Vol-" keys on the remote controller.

### 14.3.3 Subtitle decode

See figure 28.

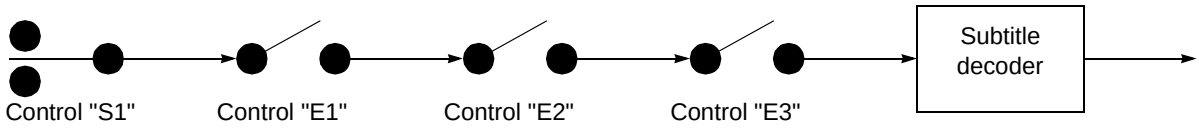



**Figure 28: Subtitle decoder control - logical model**

#### 14.3.3.1 Enabling controls

E1 is the first point of application control over whether or not subtitles are presented. It is achieved by the presence of an active MHEG-5 Video object within an active MHEG-5 Stream object, with the displayed video scaled to full-size. See clause 14.4.1.1.

E2 is the second point of application control over whether or not subtitles are presented. It is dependent on the state of subtitle presentation as determined by the SetSubtitleMode ResidentProgram. See clause 11.10.10.4.

E3 is the viewer's control of over whether or not subtitles are presented based on their preference setting. Typically this will be defined as part of receiver set-up and/or a "Subtitle" key on the remote controller.

#### 14.3.3.2 Selection controls

S1 is the selection of a subtitle component from broadcast. This selection is based on the attributes of the MHEG-5 Stream object containing the MHEG-5 Video object identified above, and viewer preferences, e.g. language.

#### 14.3.3.3 Presentation controls

None.

## 14.4 Application impact on stream decoder specification

### 14.4.1 DVB subtitles

#### 14.4.1.1 Flexibility of control

##### 14.4.1.1.1 Subtitles are a facet of full-screen video

In this Profile subtitles are treated as a facet of the video stream and hence maintain the same position in the display stack as the video component, therefore, MHEG-5 Visible objects may overlay the TV service's DVB Subtitles.

If video is scaled to other than full size by a ScaleVideo elementary action the subtitles shall not be presented.

##### 14.4.1.1.2 Subtitles have priority if enabled

If a platform does not support the simultaneous display of subtitles and MHEG-5 applications then subtitles shall be presented in preference to MHEG-5 applications if:

- the viewer has selected to view subtitles (by a subtitle key or a stored user preference); and
- a subtitle component is signalled in the PMT of the service.

If the user deactivates presentation of subtitles then normal presentation of the MHEG-5 application shall return.

Where a platform supports the simultaneous display of subtitles and MHEG-5 applications the application author may choose to disable presentation of subtitles using the SetSubtitleMode ResidentProgram.

## 14.4.2 Video decoder performance

The real-time performance of the MPEG Video ISO/IEC 13818-2 [10] decoder shall be maintained regardless of the number of visibles that obscure it.

## 14.4.3 Trick modes

Receivers shall not implement trick modes. However, as described in ISO/IEC 13522-5 [14], speed > 0 shall be treated as normal decoding, speed ≤ 0 shall pause the decoding.

### 14.4.3.1 Pause behaviour

The effects of pausing a Stream object (by using the SetSpeed(0) action) on its StreamComponents are as follows:

- if the Stream object contains an active Video object, the video decoder output (and any active subtitle graphics) freezes on, and maintains, the current, or a subsequent, frame in an implementation dependent way;
- if the Stream object contains an active Audio object, the audio output is muted.

If a Stream object has its Storage attribute set to Stream, then whilst it is paused the associated stream component decoders shall discard incoming data.

Stream component decoders that are controlled by other Stream objects shall not be affected by this behaviour.

### 14.4.3.2 Multiple stream objects

The components of a single MPEG program and components sharing a common PCR may be divided among more than one Stream object. See clause 14.7.2.

Pausing one Stream object has no effect on any other active Stream objects including those associated with the same program.

Therefore, for example, if the video and audio components of a program are associated with two different Stream objects the video can be paused (using a SetSpeed action targeted at its Stream object) while allowing the audio to continue decoding.

## 14.4.4 MPEG presentation

This clause describes the scaling of the output of the MPEG video decoder when an MHEG-5 application is executing. It applies to both MPEG video streams and bitmaps delivered as MPEG I-frames.

### 14.4.4.1 MPEG scaling reference model

The model of the relationship between the output of the MPEG Video Decoder and the On Screen Display (OSD) is shown in figure 29. Significantly, the OSD is modelled as being inserted **after** the decoder's format conversion (which typically provides functions such as centre-cut-out and letter boxing to adapt a 16:9 signal for 4:3 displays) but before the display's format conversion (which typically provides functions to adapt a 4:3 signal to a 16:9 display). The MHEG-5 ScaleVideo and ScaleBitmap actions are considered to act on the decoder's format conversion circuits, i.e. before the OSD is added.

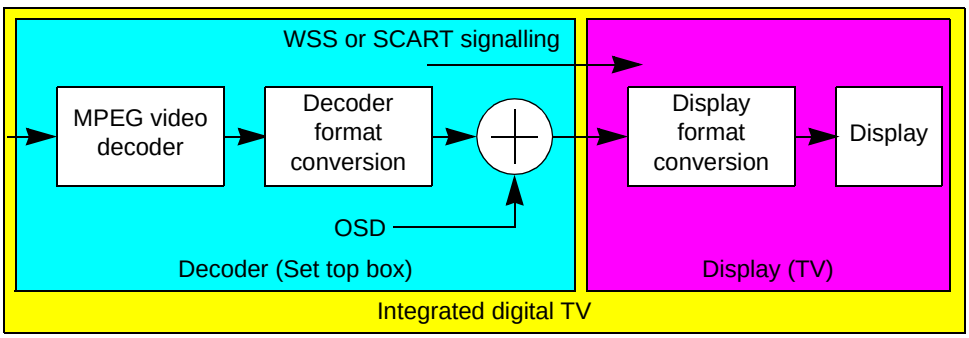



**Figure 29: Receiver and display format processing reference model**

#### 14.4.4.2 Transparency of MPEG encoding

The resolution at which the MPEG video ISO/IEC 13818-2 [10] is encoded is transparent to the application and is always considered logically to be 720 x 576. This affects MHEG-5 scaling of video and I-frames and the VideoToGraphics resident program (see clause 11.10.10.1)

The examples below illustrate this principle. However, for simplicity, they ignore the effects of letter box or centre cutout processing that may additionally be required to adapt the video aspect ratio to that of the display. See also clause 14.5.

EXAMPLE 1: Consider MPEG video encoded at 352 x 288 resolution (which ETSI TR 101 154 [3] regards as a "full screen" format). When an MHEG-5 application requests "full screen" presentation of this video it logically (from MHEG-5's point-of-view) requires 100 % scaling.

For this case, transparently to the MHEG-5 engine, the receiver must implement x2 horizontal and vertical scaling to reconstruct the image to the full-screen size. The result of the x2 scaling (a 704 x 576 image) shall be centred within the 720 x 576 display. The gap between the 720 x 576 display and the reconstructed MPEG shall be filled with black.

The gap between the 720 x 576 display and the reconstructed 352 x 288 MPEG is outside of the analogue active line and so is not generally visible. However, in theory the MPEG image might be smaller, 300 x 200 for example. This would be considered as a cropped 352 x 288 image and so reconstruct to 600 x 400. In this case the gap around the 600 x 400 image centred in the 720 x 576 display would be significant and visible.

EXAMPLE 2: Consider MPEG video encoded at 544 x 576 resolution and an MHEG-5 application requesting "full screen" presentation of this video.

Transparently to the MHEG-5 engine the receiver implements x4/3 horizontal and x1 vertical scaling to reconstruct the image to 725 x 576 from which the central 720 x 576 area is extracted for display.

#### 14.4.4.3 Quarter-screen MPEG

Where the content is MPEG a ScaleVideo action on a Video object and a ScaleBitmap action applied to a Bitmap object is logically applied to the MPEG decoding pipeline after the reconstruction of the MPEG image to full-screen size as described in clause 14.4.4.2.

The mandatory scaling factors supported by this Profile are full-screen (720 x 576) and quarter-screen (360 x 288). Receivers may also support x2 scaling (1440 x 1152).

When MPEG is shown full-screen the limits of the analogue active line and the display overscan naturally conceals defective or inactive pixels at the margins of the picture. When the picture is scaled to less than full-screen size these pixels may be revealed. To prevent this, applications should mask quarter-screen MPEG video and bitmaps using a box size of 344 x 284 and centre the MPEG decode using a position offset of (-8, -2) (see SetVideoDecodeOffset(NewXOffset, NewYOffset in clause 11.12.10.2).

##### 14.4.4.3.1 Subtitles and scaled video

See clause 14.4.1.1.

#### 14.4.4.4 BoxSize for MPEG images

If the scaled decoded image does not completely fill the area described by the Video or Bitmap object's BoxSize attribute, the remaining area of the object shall appear black.

#### 14.4.4.5 Video / I-frame object placement

All receivers are required to display images that meet the following restrictions:

- For a full-screen image (100 % scaling) the entirety of the decoded image (before any masking) is enclosed within the display area.
- For an image scaled below 100 % (i.e. quarter-screen) the entirety of the image after scaling (but before any masking) is enclosed within the display area. As a consequence:
  - this includes non-zero positions if the image is scaled to < 100 %;
  - valid positions for the image depend on its size before masking and any offset in use.
- The top line of the video is offset an even number of display lines from the origin of the display area.

In addition, receivers supporting level 1 for VideoDecodeOffset/BitmapDecodeOffset shall also allow images to be partially off-screen provided that:

- For quarter-screen image, at least half of the decoded image width and height after scaling (but before any masking) remains within the display raster.
- For full-screen image, at least half of the decoded image height and one quarter of the width remains within the display raster.

If applications erroneously invoke conditions that do not meet these criteria (for example an illegal position) the engine shall either not present the object or shall correctly present it. Note that applications may transiently position objects such that the above conditions are not met.

## 14.5 Application control of aspect ratio

In the MPEG scaling reference model (see clause 14.4.4.1), two transformation stages combine to influence the final appearance of the displayed picture:

- the Decoder Format Conversion (DecFC) governs how the video and MHEG-5 graphics are combined;
- the Display Format Conversion, influenced by Widescreen Signalling information (WSS) and viewer preferences, determines the size and shape of the final image.

The two controls, DecFC and WSS, are influenced by a combination of the following factors:

- the display aspect ratio;
- the presence of full-screen video;
- the aspect ratio of any video, as well as any associated Active Format Descriptor (AFD);
- the AspectRatio attribute of any MHEG-5 Scene;
- the current "Widescreen Alignment Mode" (see clause 11.10.10.2).

The following clauses describe the required behaviour for the three cases of no video, quarter-screen video and full-screen video. Video scaled above normal size shall be considered the same as full-screen video.

### 14.5.1 No active video object

Where there is no active Video object, only control of the WSS is relevant. If the current scene has no explicit AspectRatio, the receiver shall attempt to fill the display with the MHEG-5 scene, otherwise the receiver shall signal the specified AspectRatio to the display.

In summary, the WSS shall be as follows in table 79.

**Table 79: WSS signalling - no active Video object**

| | **Scene AspectRatio** | | |
|---|---|---|---|
| **Display aspect ratio** | **4:3** | **16:9** | **Unspecified** |
| **4:3** | 4:3 | 16:9 (NOTE) | 4:3 |
| **16:9** | 4:3 | 16:9 | 16:9 |
| NOTE: WSS signalling to 4:3 displays will in many cases be ignored. | | | |

### 14.5.2 I-frames

MPEG I-frames shall be presented with no DecFC (except for any scaling) i.e. the raster's encoded aspect ratio and any AFD shall be ignored. At full-screen size, pixels in the I-frame therefore align precisely with pixels in the MHEG-5 Scene.

### 14.5.3 Quarter-screen video

When an active Video object is scaled below full-screen size, WSS shall be the same as when no video is present (see clause 14.5.1).

The DecFC shall, as far as possible, be set so as to preserve the video aspect ratio and fill the 360 x 288 quarter-screen video decode area. It is recognized that some receivers cannot perform a pillar box transformation in the decoder and may have to distort 4:3 pictures for display in 16:9.

Table 80 illustrates the preferred DecFC transformations for each state of WSS (obtained from clause 14.5.1).

**Table 80: WSS signalling - quarter-screen video**

| | **Video coded frame** | |
|---|---|---|
| **WSS** | **4:3** | **16:9** |
| **4:3** | None | Centre cut-out |
| **16:9** | Pillar box (NOTE) | None |
| NOTE: Where a pillar box transformation cannot be supported, the preferred DecFC is None. | | |

### 14.5.4 Video full-screen or greater

Where the MHEG-5 Scene's AspectRatio is unspecified, both WSS and DecFC shall be the same as if no MHEG-5 application were running.

Where the active MHEG-5 Scene **does** specify an AspectRatio, the WSS and DecFC shall be set according to the video format and Scene AspectRatio, as follows. This mode is intended to support the alignment of MHEG-5 graphics over video.

See table 81.

**Table 81: Video full-screen or greater**

| **Video** | **4:3 coded frame** | | **16:9 coded frame** | |
|---|---|---|---|---|
| **Scene** | **4:3** | **16:9 (NOTE 1)** | **4:3** | **16:9** |
| **WSS (NOTE 2)** | 4:3 | 16:9 | 4:3 | 16:9 |
| **DecFC** | None | None | As selected (NOTE 3) | None (NOTE 4) |
| NOTE 1: The case of aligned presentation of 4:3 video with a 16:9 Scene is defined for completeness but is unlikely to be of use to applications. Video aspect ratio will not be preserved in this case. | | | | |
| NOTE 2: WSS follows the Scene AspectRatio, as in the "no video" and "quarter-screen video" cases. | | | | |
| NOTE 3: The DecFC to be used for display of 16:9 pictures on 4:3 displays is determined by the currently-selected WidescreenAlignmentMode when an explicit 4:3 Scene AspectRatio is used (see clause 11.10.10.2). | | | | |
| NOTE 4: This mode gives an anamorphic picture on 4:3 displays that do not observe WSS signalling. It is intended that this mode be used when the video is not aspect ratio sensitive. | | | | |

### 14.5.5 Decision trees

The behaviour described above can be summarized as shown in figures 30 and 31.

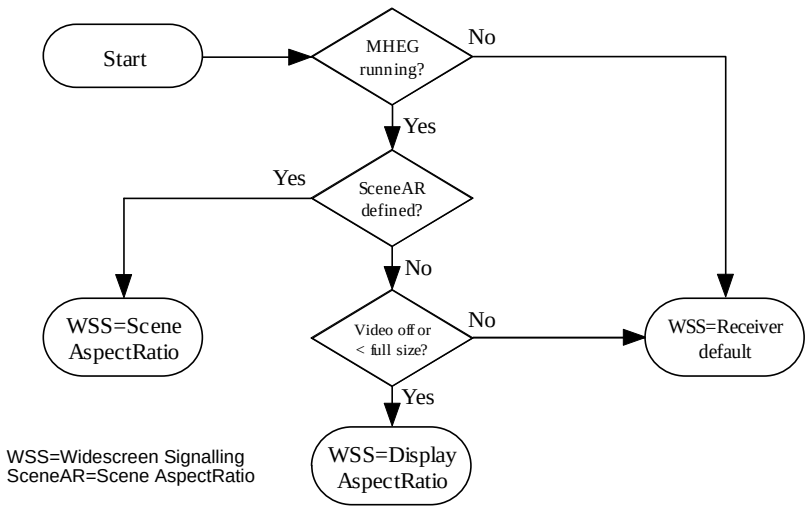



**Figure 30: Widescreen Signalling (WSS) decision tree**

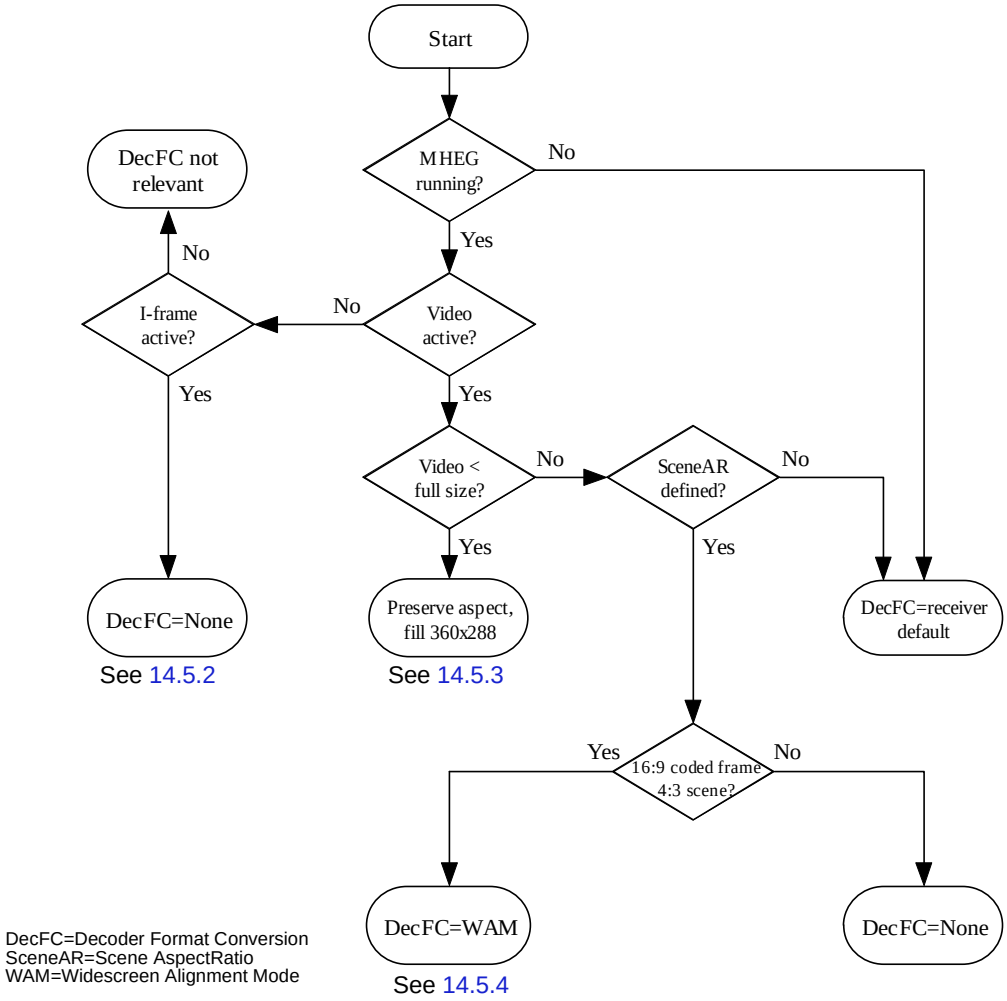



**Figure 31: Decoder Format Conversion (DecFC) decision tree**

## 14.6 Persistent storage

**The engine shall provide "persistent" storage for 1 024 bytes of data. Storage may be implemented in RAM and may be lost when the receiver is in stand-by or off.**

The file name used to access this storage shall be of the form "ram://<name>". It is the responsibility of the broadcasters to arrange a practise for the use of <name> such that there is no accidental collision of file names.

When writing a file to persistent storage the receiver shall execute the following steps:

1) If the file to be written is larger than the total size of the persistent store the action shall complete returning StoreSucceeded is False.

2) Regardless of the availability of free memory, if a file of the same name as the file to be written already exists in the persistent storage it shall be deleted.

3) If there is insufficient free memory in the persistent storage for the file to be written, existing files shall be deleted in chronological order, i.e. oldest first, as required.

4) The action shall complete returning StoreSucceeded is True.

Also note that:

- only the data is stored (not type information);
- the decoder model for memory use in the persistent store is given in table 82;
- the set of values is stored in the order they are enumerated in the ASN.1 DER encoding of the StorePersistentInVariables list;
- the behaviour of ReadPersistent() is undefined unless the set of values is enumerated in the same order as the StorePersistent() that created them.

**Table 82: Memory model for persistent storage**

| Data | |
|---|---|
| Integer | **4 bytes**<br>This shall be able to contain all of the possible values for an IntegerVariable as defined under clause 11.11. |
| Boolean | **1 byte**<br>This shall be able to contain all of the possible values for an BooleanVariable as defined in clause 11.11. |
| OctetString | **Number of bytes in OctetString + 4 bytes**<br>This shall be able to contain all of the possible values for an OctetString as defined in clause 11.11.<br><br>This model assumes an integer encoding the length of the string is stored in addition to the string data. |
| ObjectReference | **Number of bytes in GroupIdentifier+ 8 bytes**<br>This shall be able to contain all of the possible values for an Object Reference given the allowed values for GroupIdentifier and ContentReference and Integer as defined in clause 11.11.<br><br>This model assumes ObjectReferences are stored like an OctetString with an Integer. |
| ContentReference | **Number of bytes in ContentReference+ 4 bytes**<br>This shall be able to contain all of the possible values for an OctetString as defined in clause 11.11.<br><br>This model assumes ContentReferences are stored like an OctetString. |

### 14.6.1 Storage of file names

The <name> part of the file name "ram://<name>" shall be eight bytes long. The receiver shall provide storage for at least 32 such file names associated with the "ram://" persistent store. See clause 17.7.4.

## 14.7 Receiver resource model

### 14.7.1 Memory

A complete model of how receiver memory is consumed by a running application does not currently exist. However, some aspects have been defined in this Profile as follows:

- persistent file store and directory structures (see clause 14.6);
- the application identifier stack (see clause 14.8);
- buffers required to receive data from the network (see clause 15.2.7).

NOTE: The original intention was that all receivers shall provide a minimum of 1 MB of RAM as defined by this model, i.e. physical RAM requirements might be different.

### 14.7.2 Numbers of objects

The minimum number of concurrently active MHEG-5 objects using stream decoders that this Profile of engine is required to support are:

- 1 Video object (an MPEG video stream) **or** 1 Bitmap object using MPEG I-frame encoding;
- 1 Audio object (an MPEG audio stream) with source Stream or Memory.

See clause 17.3.1.

The numbers of other presentable object types (e.g. PNG bitmaps, buttons etc.) are only limited by the decoder memory model.

#### 14.7.2.1 Single PCR

More than one Stream object and StreamComponent may be playing at a single time provided that:

- the above rules for the number of concurrently active objects are observed;
- at any point in time the receiver shall have to handle at most a single PCR.

Therefore, for example, an MPEG audio and an MPEG video stream from different MPEG programs (with a common PCR_PID) could each be associated with a different Stream object. If both objects are running at normal speed the audio and video shall be synchronous.

Or, broadcast MPEG video (with a defined PCR_PID) could be presented at the same time as MPEG audio from a file, which does not rely on a related PCR.

Or, MPEG audio from a file, which does not rely on a related PCR, could be the only stream based content being presented.

See also clause 14.4.3.

### 14.7.3 Link recursion behaviour

Engines shall allow at least 256 concurrent Actions, and at least 4096 ElementaryActions, pending processing.

### 14.7.4 Timer count and granularity

Engines shall allow at least 16 concurrent MHEG-5 timers to be active.

When no more than four timers are active they shall maintain an accuracy of ±10 ms. Note that the time required to generate an MHEG-5 TimerFired event is not specified.

When more than four timers are active the accuracy may degrade in a platform-specific manner.

### 14.7.5 Timer duration

Receivers shall support timer durations up to at least 86 400 000 ms (24 hours).

## 14.8 Receiver process priority

### 14.8.1 OSD arbitration

Access to display resources is (in order of decreasing priority):

1) Other display using processes (e.g. receiver displays such as the navigator, displays produced by a CA system, CI module etc.).
2) Broadcast components, e.g. video, audio, DVB Subtitles, MHEG-5 applications. See also clause 14.4.1.1.

When a process with higher priority requests access to the display resources it shall be granted it. The present document defines three scenarios for handling this (see below). Although it does not specify in detail the actual implementation, in all cases the receiver shall manage stream decoders appropriately.

NOTE: A receiver may use different scenarios under different circumstances as convenient.

The scenarios are:

- Overlay receiver graphics on top of the MHEG-5 DisplayStack.

  In this scenario the MHEG-5 application loses the "focus" (so cannot get user interaction) but otherwise executes as normal. Use of the display by a competing process is "transparent" to the MHEG-5 application and no special management of stream decoders is required. The effects are identical to the user having left the room for a period.

- Remove the display resource from the MHEG-5 engine.

  In this scenario the receiver shall set all stream component decoders to decode the default components for the default service as determined by the receiver and set their presentation (including any video scaling and/or offset, and audio volume control) to the default state. The receiver shall ensure that the presentation of any stream components selected under application control ends before removal of the OSD resource from the application; this is to ensure that the viewer is not exposed to video normally (partially) obscured by application graphics.

  When the display resource is returned to the application, the engine is responsible for re-establishing the state of the display, including the presentation of stream components.

  NOTE: Other resources may also be removed from the MHEG engine and in the limiting case the engine is completely descheduled clause 14.8.2.

- Kill the application.

  The management of stream decoders is as described in clause 14.2.1.

When the display resource is returned to the MHEG-5 engine the receiver is responsible for restarting the appropriate default application for the service.

## 14.8.2 Event handling whilst de-prioritized

### 14.8.2.1 Transparently

If the MHEG-5 engine is "transparently" de-prioritized it ceases to get the asynchronous events that are a consequence of user interaction but receives and processes all other asynchronous events.

With respect to this Profile the events not received are: AnchorFired, EntryFieldFull, InteractionCompleted and UserInput. The events that are received are: AsynchStopped, ContentAvailable, CounterTrigger, EngineEvent, StreamEvent, StreamPlaying, StreamStopped, and TimerFired.

### 14.8.2.2 Non-transparently

If the MHEG-5 engine is "non-transparently" de-prioritized the state of the application is preserved, but some or all of the resources supporting the engine are removed. For example, demultiplexer resources may be allocated to the foreground process in which case the application may miss stream events. In the limiting case the engine may receive no processor cycle while de-prioritized.

While de-prioritized the engine's behaviour shall be self consistent. For example, if the engine's resources allow continued loading of content then the consequences of such content loading, such as generation of ContentAvailable events, shall be correct.

On being re-prioritized as the foreground process the MHEG-5 engine shall raise the "PauseResume" EngineEvent (see table 20).

# 14.9 Interaction with DVB Common Interface module system

## 14.9.1 Overview

In addition to the automatic booting of broadcast applications described in clause 9.3 a file system and an application can be introduced by a DVB CI module so that it can use MHEG-5 to interact with the user.

This clause is only relevant to receivers that implement the DVB CI.

## 14.9.2 Introduction of CI sourced file system

Under certain conditions (see clause 14.9.3.1) the application MMI mechanism described in ETSI TS 101 699 [18] can be used to:

- mount a CI module as a file source (see clause 16.3);
- launch an application object.

## 14.9.3 Guidelines for using Application MMI resource

This clause describes the use of the application MMI introduced by ETSI TS 101 699 [18] in the context of this Profile.

### 14.9.3.1 Resource contention

See 6.5.1 in ETSI TS 101 699 [18].

A module shall be guaranteed access to the application MMI in the following circumstance, in addition to those defined by BS EN 50221 [5] and ETSI TS 101 699 [18].

The reasons defined by BS EN 50221 [5] and ETSI TS 101 699 [18] are:

- following a CA_PMT message whose ca_pmt_id is "ok_mmi";
- when responding to an EnterMenu from the host;
- when responding to a GetServiceReq (see ETSI TS 101 699 [18]).

These are extended by following the receipt of a CA_PMT message whose ca_pmt_cmd_id is "ok_descrambling" when the reason for issuing the CA_PMT message is the selection of a new service.

NOTE: In the context this Profile it is guaranteed that any MHEG-5 application will have been terminated prior to the channel change occurring.

Additionally the CA_PMT may be transmitted to the module when the version number of the PMT changes or there is a change in the PMT's current_next_indicator. However, in these cases the module is not guaranteed a MMI session by the host.

### 14.9.3.2 RequestStart

See 6.5.2 in ETSI TS 101 699 [18].

#### 14.9.3.2.1 Application Domain Identifier

The string "EUMHEGP1" (0x45554D4845475031) shall be used in the RequestStart message to identify that the required application domain is *ETSIEngineProfile1* MHEG-5.

#### 14.9.3.2.2 Initial object

In addition to the defined semantics in ETSI TS 101 699 [18] it shall be possible to transmit a RequestStart message from the module to the host that contains an InitialObjectLength of 0 and therefore no InitialObject. This case modifies the normal semantic of RequestStart:

- If InitialObjectLength is 0 and an MHEG-5 application is currently running then the application is **not** killed and the CI file system is mounted and becomes available to the application.
- If InitialObjectLength is 0 and no MHEG-5 application is currently running then the CI file system is mounted and the receiver continues looking for a broadcast auto-boot application (its normal behaviour see clause 8.1.3).
- If the InitialObjectLength > 0 then the RequestStart message specifies a new application object that is to be run, any currently running application is killed to make way for the application specified by RequestStart (this is the standard semantic defined in ETSI TS 101 699 [18]). The CI file system is mounted.

Also, with reference to the "Application context" (see clause 8.1.5):

- If InitialObjectLength is 0 there is no change in the application context for any currently running application.

  In particular, the mounted object carousel (if any) remains the same and the Current source remains unchanged.
- If InitialObjectLength is > 0 then the application context becomes that of the newly introduced application.

  In particular, the service's Initial carousel (if any) is mounted and the Current source becomes the CI file system (i.e. "CI:").

When InitialObjectLength > 0 then InitialObject shall specify a valid DVB-CI Application object. The source of the file path is implicitly "CI://" so the string "foo/bar" specifies file "bar" in the sub-directory "foo" of the root of the CI device.

### 14.9.3.3 RequestStartAck

See 6.5.3 in ETSI TS 101 699 [18].

### 14.9.3.4 FileRequest

See 6.5.4 in ETSI TS 101 699 [18].

The FileRequest message will be overloaded to allow the transmission to the module of two different request types. The first is a file request as defined in ETSI TS 101 699 [18]. This is used to retrieve MHEG-5 files and content from the module. The second allows the creation of a private data pipe between the host and the module.

See table 83.

**Table 83: FileNameByte field of the FileRequest**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| `RequestType` | 8 bits | |
| `if (RequestType == "file") {` | | |
| `  for ( i = 0; i < (N –1); i++ ) {` | | |
| `    FileNameByte` | 8 bits | |
| `  }` | | |
| `}` | | |
| `if (RequestType == "data") {` | | |
| `  for ( i= 0; i < (N - 1); i++ ) {` | | |
| `    DataBytes` | 8 bits | |
| `  }` | | |
| `}` | | |

#### 14.9.3.4.1 RequestType

Defines the type of request being made by the host. See table 84.

**Table 84: RequestType field of FileRequest**

| RequestType | RequestType value |
|---|---|
| File | 0x00 |
| Data | 0x01 |

#### 14.9.3.4.2 FileNameByte

FileNameByte is a valid name for a DVB CI file as defined in clause 8.1.3.

#### 14.9.3.4.3 DataBytes

The data bytes for the module.

### 14.9.3.5 FileAcknowledge

See 6.5.5 ETSI TS 101 699 [18].

The FileAcknowledge will be overloaded to permit the transmission from the module to the host of either file request replies or data replies.The semantics shall be as defined in ETSI TS 101 699 [18] except for the following.

#### 14.9.3.5.1 FileOK

This 1-bit field is set to "1" if the file is available or this message is an acknowledgement for a FileRequest message with RequestType "data" and "0" otherwise. See table 85.

**Table 85: FileByte field of the FileAcknowledge**

| Syntax | No. of bits | Mnemonic |
|---|---|---|
| `RequestType` | 8 bits | |
| `if (RequestType == "file") {` | | |
| `  FileNameLength` | 8 bits | |
| `  for ( i = 0; i < FileNameLength; i++ ) {` | | |
| `    FileNameByte` | 8 bits | |
| `  }` | | |
| `  FileDataLength` | 32 bits | |
| `  for (i = 0; i < FileDataLength; i++ ) {` | | |
| `    FileDataByte` | 8 bits | |
| `  }` | | |
| `}` | | |
| `if (RequestType == "data") {` | | |
| `  For ( i= 0; i < (N-1); i++ ) {` | | |
| `    DataBytes` | 8 bits | |
| `  }` | | |
| `}` | | |

#### 14.9.3.5.2 RequestType

Defines the type of request being responded to by the host. See table 86.

**Table 86: RequestType field of FileAcknowledge**

| RequestType | RequestType value |
|---|---|
| File | 0x00 |
| Data | 0x01 |

#### 14.9.3.5.3 FileNameLength

The number of bytes in the name of the file.

#### 14.9.3.5.4 FileNameByte

FileNameByte is the name of the file requested by the host. This data is returned to the host to allow for implementations that may request more than one file simultaneously.

#### 14.9.3.5.5 FileDataLength

The number of bytes of data in the file.

#### 14.9.3.5.6 FileDataByte

A byte of the file requested.

#### 14.9.3.5.7 DataBytes

The data bytes for the host.

### 14.9.3.6 AppAbortRequest

See 6.5.6 in ETSI TS 101 699 [18].

No values of AbortReqCode are defined for this application domain. So, the host shall terminate any MHEG-5 application on the host that has been started as a consequence of the current application MMI session. Broadcast applications will not be terminated by this message. The host will follow the defined auto-boot procedure following the termination of an application.

### 14.9.3.7 AppAbortAck

See 6.5.7 in ETSI TS 101 699 [18].

No values of AbortAckCode are defined in this application domain.

### 14.9.3.8 Asynchronous events

A mechanism for the DVB CI module to send asynchronous events or messages to an MHEG-5 application does not currently exist.

## 14.9.4 Application Info Resource "Enter_Menu"

The host shall provide access to module applications under user control from the resident navigator software.

# 15 Object carousel profile

## 15.1 Introduction

The broadcast applications are transmitted using the DSM-CC user-to-user object carousels.

This Profile is based on the following specifications:

- ISO/IEC 13818-1 [9] - MPEG 2 systems;
- ISO/IEC 13818-6 [12] - DSM-CC;
- ETSI EN 301 192 [2] - DVB specification for data broadcasting;
- ETSI TR 101 202 [15] - Implementation guidelines for databroadcasting.

With the constraints and extensions described here.

Table 87 shows the key to certain notations that are used in the "value" columns of the syntax tables.

**Table 87: Key to notations**

| Symbol | |
|---|---|
| + | A value that is "allocated", e.g. configuration parameter of the object carousel server. |
| * | A value that is "calculated", e.g. a field whose value is calculated by the carousel server as a consequence of the number of bytes in other fields. |
| | Grey shading indicates optional fields that receivers may ignore. |

## 15.2 Object carousel profile

### 15.2.1 DSM-CC sections

All object carousels messages are transmitted using DSM-CC section format. The DSM-CC Section format is defined in clause 9.2 of the DSM-CC specification.

The DSM-CC standard provides an option to use either a CRC32 or a checksum for detecting bit errors. For this Profile, we make the restriction shown in table 88.

**Table 88: Restriction on DSM-CC section format**

| Field | Restriction | Source |
|---|---|---|
| section_syntax_indicator | 1 (indicating the use of the CRC32). | The present document |
| last_section_number | For sections transporting DownloadDataBlock fragments:<br>- all modules intended to be retrieved shall have the last section number <= 0xFE<br>- if the last section number = 0xFF then receiver behaviour is undefined | The present document |

The maximum section length is 4 096 bytes for all types of sections used in object carousels. The section overhead is 12 bytes, leaving a maximum of 4 084 bytes of payload per section.

#### 15.2.1.1 Sections per TS packet

Any single TS packet is allowed to contain parts of no more than four sections.

### 15.2.2 Data carousel

#### 15.2.2.1 General

The definitions in table 89 apply to both the dsmccDownloadDataHeader and the similar dsmccMessageHeader.

**Table 89: Restrictions on DSM-CC DownloadData and Message headers**

| Field | Restrictions | Source |
|---|---|---|
| TransactionId | See clause 15.2.6. | The present document |
| AdaptationLength | The receiver may ignore the possible contents of the dsmccAdaptationHeader field. | The present document |

#### 15.2.2.2 DownloadInfoIndication

The DownloadInfoIndication is a message that describes a set of modules and gives the necessary parameters to locate the module and retrieve it.

Restrictions are shown in table 90.

**Table 90: Restrictions on the DII**

| Field | Restrictions | Source |
|---|---|---|
| blockSize | Maximum size 4066<br>(max. section payload - DDB-header size (18)).<br><br>The recommended blockSize is 4066. | DSM-CC<br>(The present document) |
| windowSize | 0 (not used for object carousels). | DSM-CC |
| ackPeriod | 0 (not used for object carousels). | DSM-CC |
| tCDownloadWindow | 0 (not used for object carousels). | DSM-CC |
| tCDownloadScenario | 0 (not used for object carousels). | DSM-CC |
| compatibilityDescriptor():<br>compatibilityDescriptorLength | 0 (no compatibility descriptor for object carousels). | DSM-CC |
| PrivateDataLength | The receiver may ignore the possible contents of the privateData field. | DVB |

### 15.2.2.3 DownloadServerInitiate

The DownloadServerInitiate is used in the case of object carousels to provide the object reference to the ServiceGateway (i.e. root directory) of the object carousel.

Restrictions are shown in table 91.

**Table 91: Restrictions on DSI**

| Field | Restrictions | Source |
|---|---|---|
| compatibilityDescriptor(): compatibilityDescriptorLength | 0 (no compatibility descriptor for object carousels). | DSM-CC |
| privateData | Contains the ServiceGatewayInfo structure. | DSM-CC |
| serverId | Shall be set to 20 bytes with the value of 0xFF. | DVB / The present document |

### 15.2.2.4 DownloadDataBlock

Restrictions are shown in table 92.

**Table 92: Restrictions on the DDB**

| Field | Restrictions | Source |
|---|---|---|
| moduleId | Module ids are unique within the scope of the object carousel. See DSM-CC 11.2.3. | DSM-CC |

### 15.2.2.5 ModuleInfo

The moduleInfo structure is placed in the moduleInfo field of the DownloadInfoIndication of the data carousel. It contains the information needed to locate the module.

Restrictions are shown in table 93.

**Table 93: Restrictions on the DII moduleInfo field**

| Field | Restrictions | Source |
|---|---|---|
| moduleTimeOut, blockTimeOut, minBlockTime | These fields are defined in units of µs. An appropriate value must be explicitly encoded by carousel generation equipment. There is no default value that may be encoded, i.e. 0xFFFFFFFF has no special meaning. Receivers shall not employ an in-built default instead of the signalled value as there is no way to define these without knowledge of the construction of a particular carousel. See clause 17.16.3. | The present document |
| BIOP::ModuleInfo::Taps | The first tap shall have the "use" value 0x0017 (BIOP_OBJECT_USE). The id and selector fields are not used and the receiver may ignore them. The receiver may ignore possible other taps in the list. | DVB |
| BIOP::ModuleInfo:: UserInfo | The userInfo field contains a loop of descriptors. These are specified in the DVB Data Broadcasting standard. The receiver shall support the compressed_module_descriptor (tag 0x09) used to signal that the module is transmitted in compressed form. | DVB / The present document |

Syntax is given in table 94.

**Table 94: BIOP::ModuleInfo syntax**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| BIOP::ModuleInfo() { | | | | |
| moduleTimeOut | 32 | uimsbf | + | |
| blockTimeOut | 32 | uimsbf | + | |
| minBlockTime | 32 | uimsbf | + | |
| taps_count | 8 | uimsbf | N1 | ≥ 1 |
| { | | | | |
| id | 16 | uimsbf | 0x0000 | user private |
| use | 16 | uimsbf | 0x0017 | BIOP_OBJECT_USE |
| assocTag | 16 | uimsbf | + | |
| selector_length | 8 | uimsbf | 0x00 | |
| } | | | | |
| for (j=1; j<N1; j++) { | | | | Possible additional taps that may be ignored by receivers. |
| id | 16 | uimsbf | + | |
| use | 16 | uimsbf | + | |
| assocTag | 16 | uimsbf | + | |
| selector_length | 8 | uimsbf | N2 | |
| for (j=0; j<N2; j++) { | | | | |
| selector_data | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| userInfoLength | 8 | uimsbf | N3 | |
| for (k=0; k<N3; j++) { | | | | |
| userInfo_data | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |

### 15.2.2.6 ServiceGatewayInfo

The ServiceGatewayInfo structure is carried in the DownloadServerInitiate message and provides the object reference to the ServiceGateway object.

Restrictions are shown in table 95.

**Table 95: Restrictions on the ServiceGatewayInfo**

| Field | Restrictions | Source |
|---|---|---|
| BIOP:: ServiceGatewayInfo:: downloadTaps | The receiver may ignore the downloadTap list. | The present document |
| BIOP:: ServiceGatewayInfo:: serviceContextList | The receiver may ignore the service context list. | The present document |
| BIOP:: ServiceGatewayInfo:: UserInfo | The receiver may ignore the user info. | The present document |

Syntax is given in table 96.

**Table 96: ServiceGatewayInfo() syntax**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| ServiceGatewayInfo(){ | | | | |
| IOP::IOR() | | | + | See table 109. |
| downloadTaps_count | 8 | uimsbf | N1 | software download Taps |
| for (i=0; i<N1; i++) { | | | | |
| DSM::Tap() | | | | |
| } | | | | |
| serviceContextList_count | 8 | uimsbf | N2 | serviceContextList |
| for (i=0; i<N2; i++) { | | | | |
| context_id | 32 | uimsbf | | |
| context_data_length | 16 | uimsbf | N3 | |
| for (j=0; j<N3; j++) { | | | | |
| context_data_byte | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| userInfoLength | 16 | uimsbf | N5 | user info |
| for (i=0; i<N5; i++) { | | | | |
| userInfo_data | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |

### 15.2.2.7 Download Cancel

There is no semantic for this message in this Profile. Receivers may ignore them.

## 15.2.3 The object carousel

### 15.2.3.1 BIOP Generic Object Message

The BIOP Generic Object Message is a common structure used by all the BIOP (Broadcast Inter-ORB Protocol) messages.

Restrictions are shown in table 97.

**Table 97: Restrictions on the BIOP Generic Object Message**

| Field | Restrictions | Source |
|---|---|---|
| MessageHeader::byte_ order | 0 (indicating big-endian byte order). | DVB |
| MessageSubHeader:: objectKey | Maximum length of the key shall be four bytes. | DVB |
| MessageSubHeader:: objectKind | The short three-letter aliases shall be used, plus the null-terminator. | DVB |
| Access attributes | Access attributes are not transmitted in object carousels. | DSM-CC |

### 15.2.3.2 CORBA strings

In a number of places object carousel messages include text strings. These are formatted in accordance with 12.3.2 of CORBA V2.0 and using CDR-Lite encoding as specified by DSM-CC, i.e. the text is preceded by an integer specifying the length of the string and followed by a null terminator. The size of this integer depends on the string concerned and can be seen clearly in the syntax tables that follow. However, for clarity CORBA format strings and the size of their length fields are summarized in table 98.

**Table 98: Location of CORBA format strings**

| String | Length field size (bits) | Location |
|---|---|---|
| objectKind_data | 32 | table 100. |
| objectKind_data<br>id_data<br>kind_data | 32<br>8<br>8 | table 102. |
| objectKind_data | 32 | table 104. |
| objectKind_data<br>eventName_data | 32<br>8 | table 106. |
| type_id_byte | 32 | table 109. |
| id_data<br>kind_data | 32<br>32 | table 113. |

### 15.2.3.3 BIOP FileMessage

The BIOP FileMessage is used for carrying file objects.

Restrictions are shown in table 99.

**Table 99: Restrictions on the BIOP FileMessage**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader::<br>ObjectInfo | The ObjectInfo may be empty (have a length of zero). If not empty the first 8 bytes of the ObjectInfo field shall contain the DSM::File::<br>ContentSize attribute. This is optionally followed by a loop of descriptors. | The present document |
| MessageSubHeader::<br>ServiceContextList | The receiver may skip the possible serviceContextList structures. | The present document |

Syntax is given in table 100.

**Table 100: BIOP::FileMessage syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `BIOP::FileMessage() {` | | | | |
| `  magic` | 4 x 8 | uimsbf | 0x42494F50 | "BIOP" |
| `  biop_version.major` | 8 | uimsbf | 0x01 | BIOP major version 1 |
| `  biop_version.minor` | 8 | uimsbf | 0x00 | BIOP minor version 0 |
| `  byte_order` | 8 | uimsbf | 0x00 | big-endian byte ordering |
| `  message_type` | 8 | uimsbf | 0x00 | |
| `  message_size` | 32 | uimsbf | * | |
| `  objectKey_length` | 8 | uimsbf | N1 | <= 4 |
| `  for (i=0; i<N1; i++) {` | | | | |
| `    objectKey_data` | 8 | uimsbf | + | |
| `  }` | | | | |
| `  objectKind_length` | 32 | uimsbf | 0x00000004 | |

**Table 100: BIOP::FileMessage syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| objectKind_data | 4 x 8 | uimsbf | 0x66696C00 | "fil" type_id alias |
| objectInfo_length | 16 | uimsbf | N2 | |
| DSM::File::ContentSize | 64 | uimsbf | + | objectInfo |
| for (i=0; i<N2-8; i++) { | | | | |
| objectInfo_data | 8 | uimsbf | + | |
| } | | | | |
| serviceContextList_count | 8 | uimsbf | N3 | serviceContextList |
| for (i=0; i<N3; i++) { | | | | |
| context_id | 32 | uimsbf | | |
| context_data_length | 16 | uimsbf | N4 | |
| for (j=0; j<N4; j++) { | | | | |
| context_data_byte | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| messageBody_length | 32 | uimsbf | * | |
| content_length | 32 | uimsbf | N5 | |
| for (i=0; i<N5; i++) { | | | | |
| content_byte | 8 | uimsbf | + | actual file content |
| } | | | | |
| } | | | | |

### 15.2.3.4 BIOP DirectoryMessage

The BIOP DirectoryMessage is used for carrying the directory objects.

Restrictions are shown in table 101.

**Table 101: Restrictions on the BIOP DirectoryMessage**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader:: ObjectInfo | The receiver may skip the N2 possible bytes in the objectInfo field. | The present document |
| MessageSubHeader:: ServiceContextList | The receiver may skip the N3 possible serviceContextList structures. | The present document |
| BIOP::Name | The name shall contain exactly one NameComponent. | The present document |
| BIOP::Binding:: BindingType | Either "ncontext" (in the case of a Directory object) or "nobject" (in the case of a File or a Stream object). Binding type "composite" shall not be used. | DVB |
| MessageBody:: ObjectInfo | The ObjectInfo field may be empty (have a length of zero). If not empty the first 8 bytes of the ObjectInfo field shall contain the DSM:: File::ContentSize attribute. This is optionally followed by a loop of descriptors. | The present document |

Syntax is given in table 102.

**Table 102: BIOP::DirectoryMessage syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| BIOP::DirectoryMessage() { | | | | |
| magic | 4 x 8 | uimsbf | 0x42494F50 | "BIOP" |
| biop_version.major | 8 | uimsbf | 0x01 | BIOP major version 1 |
| biop_version.minor | 8 | uimsbf | 0x00 | BIOP minor version 0 |
| byte_order | 8 | uimsbf | 0x00 | big-endian byte ordering |
| message_type | 8 | uimsbf | 0x00 | |
| message_size | 32 | uimsbf | * | |
| objectKey_length | 8 | uimsbf | N1 | <= 4 |
| for (i=0; i<N1; i++) { | | | | |
| objectKey_data | 8 | uimsbf | + | |
| } | | | | |
| objectKind_length | 32 | uimsbf | 0x00000004 | |
| objectKind_data | 4 x 8 | uimsbf | 0x64697200 | "dir" type_id alias |
| objectInfo_length | 16 | uimsbf | N2 = 0 (NOTE) | objectInfo |
| for (i=0; i<N2; i++) { | | | | |
| objectInfo_data | 8 | uimsbf | + | |
| } | | | | |
| serviceContextList_count | 8 | uimsbf | N3 | serviceContextList |
| for (i=0; i<N3; i++) { | | | | |
| context_id | 32 | uimsbf | | |
| context_data_length | 16 | uimsbf | N4 | |
| for (j=0; j<N4; j++) { | | | | |
| context_data_byte | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| messageBody_length | 32 | uimsbf | * | |
| bindings_count | 16 | uimsbf | N5 | |
| for (i=0; i<N5; i++) { | | | | Binding |
| BIOP::Name() { | | | | |
| nameComponents_count | 8 | uimsbf | N6 = 1 | See table 99 |
| for (i=0; i<N6; i++) { | | | | |
| id_length | 8 | uimsbf | N7 | NameComponent id |
| for (j=0; j<N7; j++) { | | | | |
| id_data | 8 | uimsbf | + | |
| } | | | | |
| kind_length | 8 | uimsbf | N8 | NameComponent kind |
| for (j=0; j<N8; j++) { | | | | |
| kind_data | 8 | uimsbf | + | as type_id (see Table 4-4 in ETSI TR 101 202 [15]) |
| } | | | | |
| } | | | | |
| } | | | | |
| BindingType | 8 | uimsbf | + | 0x01 for nobject<br>0x02 for ncontext |

**Table 102: BIOP::DirectoryMessage syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `IOP::IOR()` | | | + | objectRef see table 109 |
| `objectInfo_length` | 16 | uimsbf | N9 | |
| `if(kind_data == 'fil'){` | | | | |
| `DSM::File::ContentSize` | 64 | uimsbf | + | 0 means that file size is not signalled |
| `for (j=0; j<N9-8; j++) {` | | | | |
| `objectInfo_byte` | 8 | uimsbf | + | |
| `}` | | | | |
| `}` | | | | |
| `else {` | | | | |
| `for (j=0; j<N9; j++) {` | | | | |
| `objectInfo_byte` | 8 | uimsbf | + | |
| `}` | | | | |
| `}` | | | | |
| `}` | | | | |
| `}` | | | | |
| NOTE: See Item 2 under 11.3.2.2, "Directory Message Format" in DSM-CC: "*The objectInfo field shall be empty*". | | | | |

### 15.2.3.5 BIOP ServiceGateway message

The syntax of the BIOP ServiceGateway message is identical to that of the BIOP DirectoryMessage (described above) with the following exceptions:

- the object kind is "srg" rather than "dir";
- use is made of the serviceContextList (see clause 9.3.4.1).

## 15.2.4 Streams and stream events

There are two versions of stream messages. The BIOP StreamMessage is used for carrying stream objects that do not use DSM-CC stream events. The BIOP StreamEventMessage is used for carrying stream objects that include a stream carrying DSM-CC stream events.

### 15.2.4.1 BIOP StreamMessage

Restrictions are shown in table 103.

**Table 103: Restrictions on the BIOP StreamMessage (Sheet 1 of 2)**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader:: ObjectInfo | The ObjectInfo field contains the DSM::Stream::Info_T structure and optionally other data after the Stream Info structure.<br><br>Within this Profile there is no defined use of the DSM::Stream::Info_T structure and the possible other object info data following it. Receivers conforming to this Profile shall ignore this information.<br><br>Broadcasts may set the duration field to zero to indicate undefined duration. | The present document |

**Table 103: Restrictions on the BIOP StreamMessage (Sheet 2 of 2)**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader:: ServiceContextList | The receiver may skip the possible serviceContextList structures. | The present document |
| MessageSubHeader:: MessageBody | The MessageBody carries a sequence of taps.<br><br>There shall be at most one tap of use BIOP_PROGRAM_USE. This tap identifies the service that provides the media stream associated with the Stream object (via a deferred_association_tags_descriptor in the PMT). The tap may only reference programs that are broadcast on the same multiplex (i.e. receivers shall not need to tune to a different multiplex in order to receive the referenced media stream).<br><br>Receivers may ignore possible other taps (such as BIOP_ES_USE and STR_NPT_USE).<br><br>Note: There should be at most one instance of a STR_NPT_USE in any particular tap even if the tap is ignored. | The present document |

Syntax is given in table 104.

**Table 104: BIOP::StreamMessage syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| BIOP::StreamMessage() { | | | | |
| magic | 4 x 8 | uimsbf | 0x42494F50 | "BIOP" |
| biop_version.major | 8 | uimsbf | 0x01 | BIOP major version 1 |
| biop_version.minor | 8 | uimsbf | 0x00 | BIOP minor version 0 |
| byte_order | 8 | uimsbf | 0x00 | big-endian byte ordering |
| message_type | 8 | uimsbf | 0x00 | |
| message_size | 32 | uimsbf | * | |
| objectKey_length | 8 | uimsbf | N1 | <= 4 |
| for (i=0; i<N1; i++) { | | | | |
| objectKey_data | 8 | uimsbf | + | |
| } | | | | |
| objectKind_length | 32 | uimsbf | 0x00000004 | |
| objectKind_data | 8 | uimsbf | 0x73747200 | "str" type_id alias |
| objectInfo_length | 16 | uimsbf | N2 | |
| DSM::Stream::Info_T { | | uimsbf | | objectInfo |
| aDescription_length | 8 | uimsbf | N3 | aDescription |
| for (i=0; i<N3; i++) { | | | | |
| aDescription_bytes | 8 | uimsbf | + | |
| } | | | | |
| duration.aSeconds | 32 | simsbf | + | May be set to 0 to indicate undefined |
| duration.aMicroSeconds | 32 | uimsbf | + | May be set to 0 to indicate undefined |
| audio | 8 | uimsbf | + | |
| video | 8 | uimsbf | + | |
| data | 8 | uimsbf | + | |
| } | | | | |
| for (i=0; i<N2-(N3+12); i++) { | | | | |
| objectInfo_byte | 8 | uimsbf | + | |
| } | | | | |

**Table 104: BIOP::StreamMessage syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| serviceContextList_count | 8 | uimsbf | N4 | serviceContextList |
| for (i=0; i<N4; i++) { | | | | |
| context_id | 32 | uimsbf | | |
| context_data_length | 16 | uimsbf | N5 | |
| for (j=0; j<N5; j++) { | | | | |
| context_data_byte | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| messageBody_length | 32 | uimsbf | * | |
| taps_count | 8 | uimsbf | N6 | |
| for (i=0; i<N6; i++) { | | | | |
| id | 16 | uimsbf | 0x0000 | Undefined |
| use | 16 | uimsbf | + | See Table 4-12 in DVB Guidelines for Data Broadcasting |
| assocTag | 16 | uimsbf | + | |
| selector_length | 8 | uimsbf | 0x00 | No selector |
| } | | | | |
| } | | | | |

### 15.2.4.2 BIOP StreamEventMessage

Restrictions are shown in table 105.

**Table 105: Restrictions on the BIOP StreamEventMessage (Sheet 1 of 2)**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader:: ObjectInfo | The ObjectInfo field contains the DSM::Stream::Info_T and DSM::Stream:: EventList_T structures followed optionally by other object info data (which may be ignored by receivers).<br><br>See table 103 regarding the DSM::Stream::Info_T. Receivers may ignore the possible other data following the DSM::Stream::EventList_T.<br><br>The EventList_T defines a sequence of event names that correlates to the sequence of event ids in the MessageBody. eventNames_count shall equal eventIds_count. | The present document |

**Table 105: Restrictions on the BIOP StreamEventMessage (Sheet 2 of 2)**

| Field | Restrictions | Source |
|---|---|---|
| MessageSubHeader:: ServiceContextList | The receiver may skip the possible serviceContextList structures. | The present document |
| MessageSubHeader:: MessageBody | The MessageBody carries a sequence of taps followed by a sequence of event ids.<br><br>The sequence of taps follows the following rules:<br><br>• There shall be at most one tap of use BIOP_PROGRAM_USE. This tap identifies the service that provides the media stream associated with the Stream object (via a deferred_association_tags_descriptor in the PMT). The tap may only reference programs that are broadcast on the same multiplex (i.e. receivers shall not need to tune to a different multiplex in order to receive the referenced media stream).<br><br>• There shall be at most one tap with use STR_EVENT_USE or STR_ STATUS_AND_EVENT_USE. This tap indicates the PID where all StreamEvent descriptors related to the StreamEvent object are broadcast.<br><br>Receivers may ignore possible other taps (such as BIOP_ES_USE and STR_NPT_USE).<br><br>Note: There should be at most one instance of a STR_NPT_USE in any particular tap even if the tap is ignored. | The present document |

Syntax is given in table 106.

**Table 106: BIOP::StreamEventMessage syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| BIOP::StreamEventMessage() { | | | | |
| magic | 4 x 8 | uimsbf | 0x42494F50 | "BIOP" |
| biop_version.major | 8 | uimsbf | 0x01 | BIOP major version 1 |
| biop_version.minor | 8 | uimsbf | 0x00 | BIOP minor version 0 |
| byte_order | 8 | uimsbf | 0x00 | big-endian byte ordering |
| message_type | 8 | uimsbf | 0x00 | |
| message_size | 32 | uimsbf | * | |
| objectKey_length | 8 | uimsbf | N1 | |
| for (i=0; i<N1; i++) { | | | | |
| objectKey_data | 8 | uimsbf | + | |
| } | | | | |
| objectKind_length | 32 | uimsbf | 0x00000004 | |
| objectKind_data | 4 x 8 | uimsbf | 0x73746500 | "ste" type_id alias |
| objectInfo_length | 16 | uimsbf | N2 | |
| DSM::Stream::Info_T { | | uimsbf | | |
| aDescription_length | 8 | uimsbf | N3 | aDescription |
| for (i=0; i<N3; i++) { | | | | |
| aDescription_bytes | 8 | uimsbf | + | See clause 15.2.4.1 |
| } | | | | |
| duration.aSeconds | 32 | simsbf | + | See clause 15.2.4.1 |
| duration.aMicroSeconds | 32 | uimsbf | + | See clause 15.2.4.1 |
| audio | 8 | uimsbf | + | See clause 15.2.4.1 |
| video | 8 | uimsbf | + | See clause 15.2.4.1 |

**Table 106: BIOP::StreamEventMessage syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| data | 8 | uimsbf | + | See clause 15.2.4.1 |
| } | | | | |
| DSM::Event::EventList_T { | | | | |
| eventNames_count | 16 | uimsbf | N4 | |
| for (i=0; i<N4; i++) { | | | | |
| eventName_length | 8 | uimsbf | N5 | |
| for (j=0; j<N5; j++) { | | | | |
| eventName_data | 8 | uimsbf | + | (Including zero terminator) |
| } | | | | |
| } | | | | |
| } | | | | |
| for (i=0; i<N2-((N3+12) length(DSM::Event::EventList_T)); i++) { | | | | |
| objectInfo_byte | 8 | uimsbf | + | |
| } | | | | |
| serviceContextList_count | 8 | uimsbf | N6 | |
| for (i=0; i<N6; i++) { | | | | |
| context_id | 32 | uimsbf | | |
| context_data_length | 16 | uimsbf | N7 | |
| for (j=0; j<N7; j++) { | | | | |
| context_data_byte | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| messageBody_length | 32 | uimsbf | * | |
| taps_count | 8 | uimsbf | N8 | |
| for (i=0; i<N8; i++) { | | | | |
| id | 16 | uimsbf | 0x0000 | Undefined |
| use | 16 | uimsbf | + | See Table 4-12 in DVB Guidelines for Data Broadcasting |
| assocTag | 16 | uimsbf | + | |
| selector_length | 8 | uimsbf | 0x00 | No selector |
| } | | | | |
| eventIds_count | 8 | uimsbf | N4 | (= eventNames_count) |
| for (i=0; i<N4; i++) { | | | | |
| eventId | 16 | uimsbf | + | |
| } | | | | |
| } | | | | |

### 15.2.4.2.1 Stream event names and event ids

The EventList_T defines a sequence of event names that correlates 1:1 to the sequence of event ids in the MessageBody. Within each BIOP::StreamEventMessage the event names uniquely associate to event id values:

- the eventNames_count shall equal eventIds_count;
- the names in the EventList_T are zero-terminated strings;
- the eventID values in the StreamEventMessage correspond to the eventID values carried in StreamEventDescriptors.

#### 15.2.4.2.2 Generating MHEG-5 StreamEvents

To generate a MHEG-5 StreamEvent, the following data are used:

- the EventSource is the MHEG-5 Stream instance associated with the DSMCC::StreamEvent instance;
- the EventData is an OctetString comprised of the data bytes BIOP::StreamEventMessage::eventName_data, excluding the zero termination byte, for the associated eventID of this event (see clarification above). Where the eventName_length is 0 or 1 the OctetString will be empty.

The OctetString shall contain a valid UTF-8 text string.

#### 15.2.4.2.3 Tap longevity

Any taps contained within a BIOP StreamMessage or BIOP StreamEventMessage are resolved during the ContentPreparation behaviour of the relevant MHEG-5 Stream object. Thus subsequent changes to the BIOP StreamEventMessage are only enacted by the receiver on future SetData actions on the MHEG-5 Stream object.

#### 15.2.4.2.4 Stream event subscription longevity

On subscribing to a BIOP StreamEventMessage event the *eventName* to *eventId* mapping is resolved during the ContentPreparation behaviour of the MHEG-5 Stream object. Thus subsequent changes to the BIOP StreamEventMessage are only enacted by the receiver on future SetData actions.

### 15.2.4.3 Identifying services using StreamMessages and StreamEventMessages

#### 15.2.4.3.1 BIOP_PROGRAM_USE tap

StreamMessages and StreamEventMessages use the BIOP_PROGRAM_USE tap to identify a service. This tap contains an association tag value that may map to an association tag contained within a deferred_association_tags_descriptor. Note that this association tag resolves to a **service** rather than an individual component, as detailed in clause 15.3.3.

### 15.2.4.4 DSM-CC sections carrying stream descriptors

#### 15.2.4.4.1 "do it now" events

The only events required to be supported in this Profile are "do-it-now" events. Receivers shall respond to the first instance of a "do it now" event detected under a particular combination of table id, table id extension and version number. Reception of subsequent copies of the particular event shall be ignored until a different version number is detected. At this point, if the event is still "subscribed", the event shall be "re-fired". See clause 15.2.4.6.

#### 15.2.4.4.2 Section number

For this Profile receivers shall only consider section number zero.

#### 15.2.4.4.3 DSM-CC sections for DSMCC_descriptor_list()

If the table_id field equals 0x3D the current_next_indicator bit shall be set to "1".

#### 15.2.4.4.4 Stream event life time

The set of stream events described in a particular BIOP::StreamEventMessage is allowed to be a subset of the events used by an application during its lifetime. Similarly, the set of stream event descriptors being transmitted at any time are not required to correspond to the set of events described by any active BIOP::StreamEventMessages.

#### 15.2.4.4.5 Encoding of table id extension

The section's table id extension field provides information on the stream descriptor(s) carried by the section (see table 107).

**Table 107: Encoding of table id extension for DSMCC_descriptor_lists**

| table_id_extension bits | | | | Payload of DSM-CC section with table ID 0x3D |
|---|---|---|---|---|
| [15] | [14] | [13 to 8] | [7 to 0] | |
| 0 | 0 | eventID[13 to 8] | eventID[7 to 0] | Section carries a single "do it now" event |
| 0 | 1 | xx xxxx | xxxx xxxx | Reserved for future use |
| 1 | 0 | xx xxxx | xxxx xxxx | Reserved for future use |
| 1 | 1 | xx xxxx | xxxx xxxx | Reserved for future use |

NOTE: The value of eventID for "do it now" events shall be in the range 0x0001 to 0x3FFF. The value 0 is not allowed (see 5.5.2.2.1 in ISO/IEC 13818-6 [12]).

#### 15.2.4.4.6 Resources to monitor stream events

##### 15.2.4.4.6.1 "do it now" events

"do it now" events are single shot events. Accordingly, receivers need to make special efforts to ensure a high probability that they can be reliably received.

Broadcasters are responsible for placing all "do it now" stream descriptors that may be of interest to an application on a single PID. This may be the same PID as is used for other DSM-CC sections.

Receivers shall dedicate a section filter to monitoring the possible transmission of "do it now" events while there is any active links waiting for such events.

See clause 17.11.

### 15.2.4.5 Stream descriptors

#### 15.2.4.5.1 Stream event descriptor

In this Profile all stream event descriptors shall only carry "do-it-now" events. Thus any eventNPT signalled in a StreamEventDescriptor shall be ignored.

As the eventId for a "do-it-now" event is signalled in the section header, and the eventNPT field is ignored, the receiver only needs to parse the section header in order to act upon the event. This stream event descriptor is included in the section to provide future compatibility.

The privateDataByte field shall be ignored by the receiver.

#### 15.2.4.5.2 NPT Reference descriptor

Receivers may ignore this descriptor if present.

#### 15.2.4.5.3 NPT Endpoint descriptor

Receivers may ignore this descriptor if present.

#### 15.2.4.5.4 Stream Mode descriptor

Receivers may ignore this descriptor if present.

#### 15.2.4.6 Mapping stream descriptors into the MHEG-5 domain

The DSM-CC Event interface provides a means for delivering events through the MPEG-2 stream. This interface has three primitives, which according to ISO/IEC 13818-6 [12] are:

- DSM Event subscribe
  - subscribe to receive an event over an MPEG stream;
- DSM Event unsubscribe
  - indicate desire to no longer receive an event; and
- DSM Event notify
  - obtain event data from a stream event descriptor.

The occurrence of subscribe and unsubscribe shall be determined by the state of MHEG-5 objects in the currently running application - essentially relevant MHEG-5 Stream and Link objects. In particular, the occurrence of the unsubscribe primitive shall be independent of the notify primitive.

The notify primitive shall occur whenever a relevant stream event descriptor is received. In the case of "do-it-now" events the notify primitive shall occur on every version change of the descriptor carrying the event, or when the descriptor is received for the first time since subscription. This allows multiple programme events to be trapped by the same MHEG-5 Link object without any need to re-subscribe.

### 15.2.5 BIOP Interoperable Object References

An Interoperable Object Reference (IOR) is a reference to an object and it contains the necessary information to locate the object. The IOR structure may contain different options to be able to point to objects that can be reached via different types of connections. For this receiver Profile, the use of IORs is limited to references to objects carried in broadcast object carousels. For object carousels, there are two types of object references: one to be used to reference objects carried in the same object carousel and one to be used to reference objects in other object carousels.

Restrictions are shown in table 108.

**Table 108: Restrictions on the BIOP IOR**

| Field | Restrictions | Source |
|---|---|---|
| IOP::IOR::type_id | Contains the objectKind of the referenced object. A short three-letter aliases shall be used, plus a null-terminator. | The present document |
| IOP::IOR:: taggedProfileList | There shall be at least 1 taggedProfile included in an IOR. For objects carried in a broadcast object carousel, the first taggedProfile shall be either a TAG_BIOP profile or a TAG_LITE_OPTIONS. If the first tagged profile is some other profile, the object is not carried in a broadcast object carousel and the receiver shall ignore the object. | The present document |

Syntax is given in table 109.

**Table 109: IOP::IOR syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `IOP::IOR {` | | | | |
| `type_id_length` | 32 | uimsbf | N1 | |
| `for (i=0; i<N1; i++) {` | | | | |
| `type_id_byte` | 8 | uimsbf | + | Short alias type_id (e.g. "dir") |
| `}` | | | | |
| `taggedProfiles_count` | 32 | uimsbf | N2 | Profile bodies |
| `IOP::taggedProfile()` | | | | For objects in broadcast carousels: either BIOPProfileBody or LiteOptionsProfileBody. |

**Table 109: IOP::IOR syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `for (n=0; n<N2-1;n++) {` | | | | Receiver may ignore other profiles (2 to N1) if present |
| `IOP::taggedProfile()` | | | | |
| `}` | | | | |
| `}` | | | | |

### 15.2.5.1 BIOPProfileBody

The BiopProfileBody is used for references to objects within the same object carousel.

Restrictions are shown in table 110.

**Table 110: Restrictions on the BIOPProfileBody**

| Field | Restrictions | Source |
|---|---|---|
| BiopProfileBody::byte_ order | 0 (indicating big-endian byte order). | DVB |
| BiopProfileBody:: LiteOptionComponents | The list shall contain a exactly 1 BiopObjectLocation and exactly 1 DSM:: ConnBinder as the first two components in that order. The receiver may ignore possible other components in the list. | The present document |
| DSM::ConnBinder | For objects carried in the broadcast object carousel, the first tap shall be of type BIOP_DELIVERY_PARA_USE. If there is another type of tap in the first position, the receiver may ignore this object reference, as it is a reference for object accessed using another type of protocol (e.g. for return channel use). The receiver may ignore possible other taps in the list. | The present document |
| DSM::Tap | In the BIOP_DELIVER_PARA_USE tap, the id field is not used and may be ignored by the receiver. | The present document |
| DSM::Tap::timeout | This field is defined in units of µs. An appropriate value must be explicitly encoded by carousel generation equipment. There is no default value that may be encoded, i.e. 0xFFFFFFFF has no special meaning. Receivers shall not employ an in-built default instead of the signalled value as there is no way to define these without knowledge of the construction of a particular carousel. See clause 17.16.3. | The present document |

Syntax is given in table 111.

**Table 111: BIOP ProfileBody syntax (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `BIOPProfileBody {` | | | | |
| `profileId_tag` | 32 | uimsbf | 0x49534F06 | TAG_BIOP (BIOP Profile Body) |
| `profile_data_length` | 32 | uimsbf | * | |
| `profile_data_byte_order` | 8 | uimsbf | 0x00 | big-endian byte order |
| `lite_component_count` | 8 | uimsbf | N1 | |
| `BIOP::ObjectLocation {` | | | | |
| `componentId_tag` | 32 | uimsbf | 0x49534F50 | TAG_ObjectLocation |
| `component_data_length` | 8 | uimsbf | * | |
| `carouselId` | 32 | uimsbf | + | |
| `moduleId` | 16 | uimsbf | + | |
| `version.major` | 8 | uimsbf | 0x01 | BIOP protocol major version 1 |
| `version.minor` | 8 | uimsbf | 0x00 | BIOP protocol minor version 0 |
| `objectKey_length` | 8 | uimsbf | N2 | <= 4 |
| `for (k=0; k<N2; k++) {` | | | | |

**Table 111: BIOP ProfileBody syntax (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| objectKey_data | 8 | uimsbf | + | |
| } | | | | |
| } | | | | |
| DSM::ConnBinder { | | | | |
| componentId_tag | 32 | uimsbf | 0x49534F40 | TAG_ConnBinder |
| component_data_length | 8 | uimsbf | N4 | |
| taps_count | 8 | uimsbf | N3 | |
| DSM::Tap { | | | | |
| id | 16 | uimsbf | 0x0000 | User private |
| use | 16 | uimsbf | 0x0016 | If BIOP_DELIVERY_PARA_USE is provided it shall be the first tap.<br><br>If there is another type of tap in the first position, the receiver may ignore this object reference, as it is a reference for an object accessed using another type of protocol (e.g. for return channel use). |
| assocTag | 16 | uimsbf | + | |
| selector_length | 8 | uimsbf | 0x0A | |
| selector_type | 16 | uimsbf | 0x0001 | |
| transactionId | 32 | uimsbf | * | |
| timeout | 32 | uimsbf | * | |
| } | | | | |
| for (n=0; n<N4-18; n++) { | | | | The receiver may skip over the possible additional taps |
| additional_tap_byte | 8 | uimsbf | | |
| } | | | | |
| } | | | | |
| for (n=0;n<N6;n++) { | | | | N6=N1-2 |
| BIOP::LiteComponent{ | | | | |
| componentId_tag | 32 | uimsbf | + | |
| component_data_length | 8 | uimsbf | N7 | |
| for (i=0; i<N7; i++) { | | | | |
| component_data_byte | 8 | uimsbf | | |
| } | | | | |
| } | | | | |
| } | | | | |

### 15.2.5.2 LiteOptionsProfileBody

The LiteOptionsProfileBody is used for making links to objects carried in other object carousels.

For this receiver profile, the following restrictions apply:

- LiteOptionsProfileBody shall only refer to objects within the same transport stream. Therefore, the use of the object carousel will not require tuning to a new transport stream.
- Target MHEG-5 objects in other ServiceGateways are restricted to being application objects. Therefore, the only MHEG-5 actions that can specify MHEG-5 objects in other ServiceGateways are Launch and Spawn. As above, this shall not require re-tuning.

Restrictions are shown in table 112.

**Table 112: Restrictions on the LiteOptionsProfileBody**

| Field | Restrictions | Source |
|---|---|---|
| LiteOptionsProfileBody:: profile_data_byte_order | 0 (indicating big-endian byte order). | DVB |
| LiteOptionsProfileBody:: LiteOptionComponents | The list shall contain a ServiceLocation component as the first component. The receiver may ignore possible other components in the list. | The present document |
| DSM::ServiceLocation | For objects carried in the broadcast object carousel, the service domain NSAP address shall follow the Carousel NSAP address format. If there is another type of NSAP address, the receiver may ignore this object reference, as it is a reference for object accessed using another type of protocol (e.g. for return channel use). The carousel NSAP address shall point to an object carousel in the same transport stream. If the NSAP address points to another transport stream, the receiver may ignore the object reference. | The present document |
| DSM::ServiceLocation:: InitialContext | The receiver may ignore the initial context. | The present document |

Syntax is given in table 113.

**Table 113: Syntax of LiteOptionsProfileBody with ServiceLocation component. (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| `LiteOptionsProfileBody {` | | | | |
| `profileId_tag` | 32 | uimsbf | 0x49534F05 | TAG_LITE_OPTIONS (Lite Options Profile Body). |
| `profile_data_length` | 32 | uimsbf | * | |
| `profile_data_byte_order` | 8 | uimsbf | 0x00 | big-endian byte order. |
| `lite_component_count` | 8 | uimsbf | N1 | |
| `DSM::ServiceLocation {` | | | | |
| `componentId_tag` | 32 | uimsbf | 0x49534F46 | TAG_ServiceLocation |
| `component_data_length` | 8 | uimsbf | * | |
| `serviceDomain_length` | 8 | uimsbf | 0x14 | Length of carousel NSAP address. |
| `serviceDomain_data()` | 160 | uimsbf | + | See table 114. |
| `CosNaming::Name() {` | | | | pathName |
| `nameComponents_count` | 32 | uimsbf | N2 | |
| `for (i=0; i<N2; i++) {` | | | | |
| `id_length` | 32 | uimsbf | N3 | NameComponent id |
| `for (j=0; j<N3 j++) {` | | | | |
| `id_data` | 8 | uimsbf | + | |

**Table 113: Syntax of LiteOptionsProfileBody with ServiceLocation component. (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| }<br>kind_length<br>for (j=0; j<N4 j++) {<br>kind_data<br>}<br>}<br>}<br>initialContext_length<br>for (n=0; n<N5 n++) {<br>InitialContext_data_byte<br>} | 32<br>8<br>32<br>8 | uimsbf<br>uimsbf<br>uimsbf<br>uimsbf | N4<br>+<br>N5 | NameComponent kind<br>As type_id (see Table 4-4 in ETSI TR 101 202 [15]). |
| }<br>for (n=0;n<N6;n++) {<br>BIOP::LiteComponent{<br>componentId_tag<br>component_data_length<br>for (i=0; i<N7; i++) {<br>component_data_byte<br>}<br>}<br>} | 32<br>8<br>8 | uimsbf<br>uimsbf<br>uimsbf | +<br>N7 | N6=N1-1 |
| } | | | | |

See table 114.

**Table 114: DVB carousel NSAP address (Sheet 1 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| DVBcarouselNSAPaddress() | | | | |
| AFI | 8 | uimsbf | 0x00 | NSAP for private use. |
| Type | 8 | uimsbf | 0x00 | Object carousel NSAP Address. |
| carouselId | 32 | uimsbf | + | To resolve this reference a carousel_id_ descriptor with the same carousel_id as indicated in this field must be present in the PMT signalling for the service identified below. |
| specifierType | 8 | uimsbf | 0x01 | IEEE OUI. |
| specifierData { IEEE OUI } | 24 | uimsbf | 0x<DVB> | Constant for DVB OUI. |
| dvb_service_location () { | | | | |
| transport_stream_id | 16 | uimsbf | + | This may be set to 0x0000 which indicates that the receiver shall not use the transport_stream_id when locating the service. For any other value then this field shall be used. |
| original_network_id | 16 | uimsbf | + | |
| service_id | 16 | uimsbf | + | (= MPEG-2 program_number) |
| reserved | 32 | bslbf | 0xFFFFFFFF | |

**Table 114: DVB carousel NSAP address (Sheet 2 of 2)**

| Syntax | Bits | Type | Value | Comment |
|---|---|---|---|---|
| }<br>} | | | | |

## 15.2.6 Assignment and use of transactionId values

### 15.2.6.1 Background (informative)

The use of the transactionId in the object carousel is inherited from its use as defined by the DSM-CC specification, and as such it can appear somewhat complex. The transactionId has a dual role, providing both identification and versioning mechanisms for download control messages, i.e. DownloadInfoIndication and DownloadServerInitiate messages. The transactionId uniquely identifies a download control message, however it is "incremented" whenever any field of the message is modified.

### 15.2.6.2 Use in this profile

The term "incremented" is used in the DSM-CC specification. Within the scope of the present document this shall be interpreted as "changed".

When a module is changed, the version number of the module needs to be changed. This implies that the DownloadInfoIndication message that references the module needs to be also updated. Since the DownloadInfoIndication is updated, the transactionId needs to be also changed. However, the transactionId of the DownloadInfoIndication message is used in other messages also, but the need to change the other messages should specifically be avoided and the implications of updating a module should be limited to the module itself and the DownloadInfoIndication that references the module. Therefore, additional rules on the usage of the transactionId have been specified as follows.

The transactionId has been split up into a number of sub-fields defined in table 115. This reflects the dual role of the transactionId (outlined above) and constraints imposed to reduce the effects of updating a module. However, to increase interoperability the assignment of the transactionId has been designed to be independent of the expected filtering in target receivers.

**Table 115: Sub-fields of the transactionId**

| Bits | Value | Sub-field | Description |
|---|---|---|---|
| 0 | User-defined | Updated flag | This must be toggled every time the control message is updated. |
| 1-15 | User-defined | Identification | This must and can only be all zeros for the DownloadServerInitiate message. All other control messages must have one or more non-zero bit(s). |
| 16-29 | User-defined | Version | This must be incremented/changed every time the control message is updated. |
| 30-31 | Bit 30 - zero<br>Bit 31 - non-zero | Originator | This is defined in the DSM-CC specification ISO/IEC 13818-6 [12] as 0x02 if the transactionId has been assigned by the network - in a broadcast scenario this is implicit. |

Due to the role of the transactionId as a versioning mechanism, any change to a control message will cause the transactionId of that control message to be incremented. Any change to a Module will necessitate incrementing its moduleVersion field. This change must be reflected in the corresponding field in the description of the Module in the DownloadInfoIndication message(s) that describes it. Since a field in the DownloadInfoIndication message is changed its transactionId must be incremented to indicate a new version of the message. Also, any change in the DownloadServerInitiate message implies that its transactionId must also be incremented. However, when the transactionId is divided into subfields as specified above, updating a message will change only the Version part of the transactionId while the Identification part remains the same.

Since the transactionId is used also for identifying the messages when referencing the messages in other structures, it is very desirable that these referenced would not need to be updated every time the control message is update. Therefore the following rule shall be applied when locating the messages based on the references:

"*When locating a message based on the transactionId value used for referencing the message, only the Identification part (bits 1 to 15) shall be matched.*"

Using this rule, the implications of updating a module can be limited to the module itself and the DownloadInfoIndication message describing the module. Also, this implies that if a receiver wants to find out if a particular module that it has retrieved earlier has changed, it needs to filter the DownloadInfoIndication message that described that module and check if it has been changed.

## 15.2.7 Mapping of objects to modules

DSM-CC object carousels allow one or more objects to be carried in each module. In order to optimize the performance and memory requirements additional requirements are specified:

1) If in the process of retrieving an object from the carousel a receiver acquires a module containing multiple objects, it should attempt to cache these since the expectation is that the other objects are related to the object requested and probably will be needed soon (see clause 17.18.2.1).

2) The size of a module that contains multiple objects shall not exceed 65 536 bytes in its decompressed form. For modules containing only a single object, there is no limit for the size (except what is determined by the memory in the receivers and the size of the length fields).

   NOTE: The size of a module does not include any overhead caused by the delivery protocol, i.e. Download Data Block message headers.

3) In addition to the limitations imposed by the 65 536 byte limit, directory and service gateway messages are limited to 512 object bindings per message.

## 15.2.8 Compression of modules

The modules may be transmitted either in uncompressed or compressed form. If the module is transmitted in compressed form, this is signalled by including the compressed_module_descriptor in the userInfo field of the moduleInfo in the DownloadInfoIndication message.

Presence of the compressed_module_descriptor indicates that the data in the module has the "zlib" structure as defined in RFC1950.

Table 116 shows the syntax of the compressed_module_descriptor.

**Table 116: compressed_module_descriptor**

| | No. of bytes | Mnemonic | Value |
|---|---|---|---|
| `compressed_module_descriptor(){` | | | |
| `descriptor_tag` | 1 | uimsbf | 0x09 |
| `descriptor_length` | 1 | uimsbf | |
| `compression_method` | 1 | uimsbf | |
| `original_size` | 4 | uimsbf | |
| `}` | | | |

Presence of the compressed_module_descriptor indicates that the data in the module has the "zlib" structure as defined in RFC 1950. Table 117 shows the syntax of the ZLIB structure.

**Table 117: zlib structure (Sheet 1 of 2)**

| | No. of bytes |
|---|---|
| `zlib structure(){` | |
| `compression_method` | 1 |
| `flags_check` | 1 |
| `compressed_data` | n |

**Table 117: zlib structure (Sheet 2 of 2)**

| | No. of bytes |
|---|---|
| check value | 4 |
| } | |

The receiver shall support the Deflate compression algorithm as specified in RFC 1951. This is signalled setting the least significant nibble of the compression_method to 0x8 (i.e. compression_method is xxxx1000). The receiver is not required to support other compression algorithms.

# 15.3 AssociationTag mapping

## 15.3.1 Association tags in "taps"

The DSM-CC U-U protocol defines a "tap", which is used to communicate with a lower layer communication channel. A tap contains an identifier, a "use" (which indicates the type of connection) and a 16-bit association tag (which uniquely identifies the lower level resource to be used).

## 15.3.2 Different uses of "taps"

In the UK Profile, taps are used to reference either a service (BIOP_PROGRAM_USE tap) or an elementary stream (all other types of tap).

NOTE: Some confusion is caused by the fact that both MHEG-5 and DVB have separately made use of the term "component tag". These fields are not directly interchangeable so care must be taken when referring to either. In this clause the term will always be preceded by "MHEG-5" or "DVB" as appropriate.

## 15.3.3 Using an AssociationTag to reference a service

### 15.3.3.1 BIOP_PROGRAM_USE tap

Stream and StreamEvent objects within a U-U object carousel use the BIOP_PROGRAM_USE tap to identify a service (see clause 15.2.4.1 and clause 15.2.4.2). This tap contains a DSM-CC association tag value that may match an association tag contained within a deferred_association_tags_descriptor, contained within the first descriptor loop of the PMT.

### 15.3.3.2 deferred_association_tags_descriptor

#### 15.3.3.2.1 Resolving a service

The deferred_association_tags_descriptor, as described by ETSI EN 301 192 [2] clause 9.3.2, is used to resolve an association_tag to a different PMT (i.e. a different service). If the association_tag contained in a BIOP_PROGRAM_USE tap matches an association_tag contained within a deferred_association_tags_descriptor then the service indicated by the appropriate "service_id", "transport_stream_id" and "original_network_id" triple is resolved.

#### 15.3.3.2.2 Default behaviour

If the association tag value in the BIOP_PROGRAM_USE tap does not match an association tag value in any deferred_association_tags_descriptor contained within the current service's PMT, the tag is resolved to the current service.

#### 15.3.3.2.3 Transport_stream_id field

If the "transport_stream_id" field of the deferred_association_tags_descriptor is set to 0x0000 then it shall be ignored and the receiver is free to choose which transport stream ID it selects.

#### 15.3.3.3 Service association tag mapping decision tree

See figure 32.

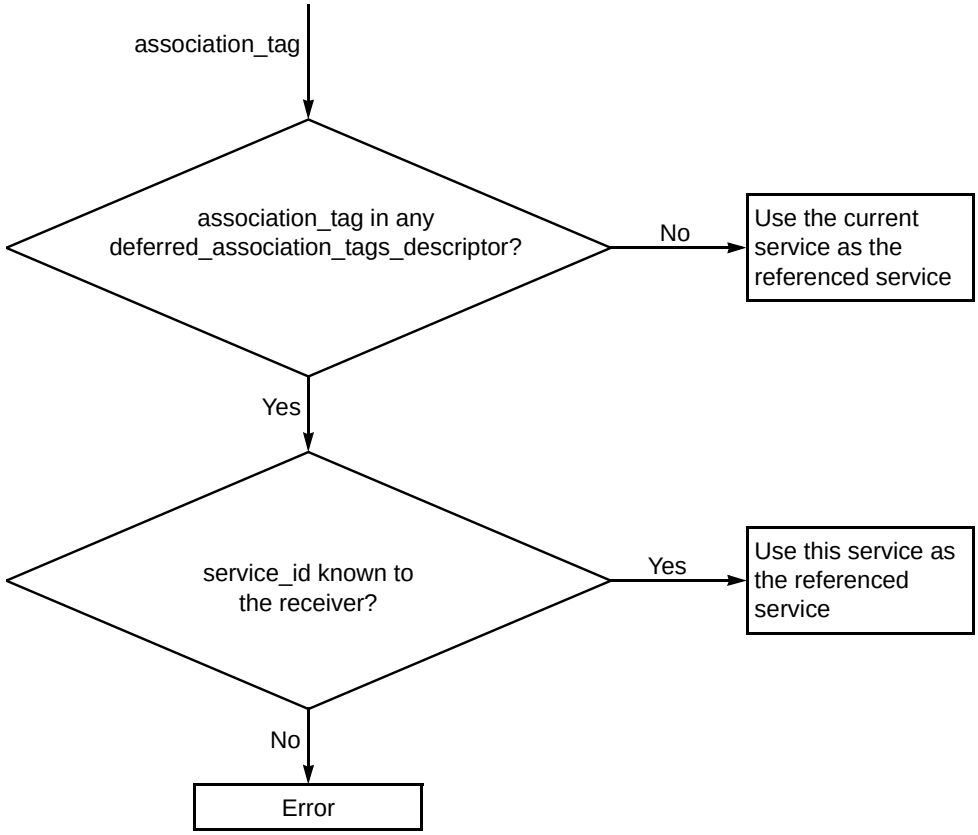



**Figure 32: Service association tag mapping decision tree**

### 15.3.4 Using an association tag to reference an elementary stream

#### 15.3.4.1 MHEG-5 ComponentTags to DSM-CC association tags

The MHEG-5 ComponentTag attribute of MHEG-5 Audio or Video objects is used to select the elementary stream which is to be decoded within the service indicated by the enclosing MHEG-5 Stream object (see clause 14.2.5). The two least signification bytes of the MHEG-5 ComponentTag are to be treated as a 16-bit association_tag and mapped to an elementary stream from the service indicated by the enclosing MHEG-5 Stream object, using the elementary stream association_tag mapping rules detailed below.

##### 15.3.4.1.1 Tag vales for default components

The special value "-1" may be used as the MHEG-5 ComponentTag attribute of Video or Audio. This associates the object with the "default" media component of the appropriate type.

NOTE: Here "default" normally means the components that the receiver would decode if the service had been selected via the receiver navigator or the SI_TuneIndex ResidentProgram. So, for example, the audio component should normally be selected with regard to the viewer's language preference.

Exceptionally, when the service is selected with "rec://svc/cur", ComponentTag "-1" means the **currently selected components rather than the default** component. No other value of MHEG-5 ComponentTag shall be used if the multiplex is specified as "rec://svc/cur". Receivers may ignore the value of ComponentTag in this case.

See also clause 14.2.5.2.

#### 15.3.4.1.2 Explicit component references

Receivers must resolve explicit (non "-1") component tag values regardless of stream type information signalled within the PMT. In this case, when components are selected under control of an application, the application author is responsible for ensuring that the components carry data suitable for the MHEG-5 stream component type.

### 15.3.4.2 Mapping DSM-CC association_tags to DVB component_tags

The DVB component_tag is an 8-bit value that maybe used to identify an elementary stream without directly referring to its PID value. Likewise, 16-bit association_tags are used by DSM-CC in order to refer to an elementary stream without directly referencing its PID value. The 16-bit association_tag value shall be used to identify an elementary stream by matching its least significant byte with a DVB component_tag.

#### 15.3.4.2.1 stream_identifier_descriptor

In UK DTT, the DVB stream_identifier_descriptor shall always be used for assigning a DVB component_tag for the elementary streams. Its is mandatory for all components referenced by an MHEG-5 application and/or object carousel.

#### 15.3.4.2.2 association_tag descriptors

However, broadcasters may choose to use association_tag_descriptors (as defined by ISO/IEC 13818-6 [12]) to indicate elementary streams. If the association_tag_descriptor is optionally used, a stream_identifier_descriptor shall still be present and the tag values shall be set consistently in each descriptor. This restriction simplifies the decision tree shown in clause 15.3.4.3 so that the second decision can be skipped.

#### 15.3.4.2.3 Elementary stream matching using the deferred_association_tags_descriptor

Use of the deferred_association_tags_descriptor to match elementary streams is not required for this Profile. All components that constitute a valid carousel broadcast to this Profile must be present in the PMT from which the carousel was mounted.

Receivers are free to implement support for elementary stream matching using the deferred_association_tags_descriptor should it be required for any other Profile or for compatibility with another standard.

#### 15.3.4.2.4 PMT changes

If the PMT changes then all active DVB component_tag references should be re-evaluated according to the elementary stream mapping decision tree.

### 15.3.4.3 Elementary stream mapping pseudo code and decision tree

See figure 33.

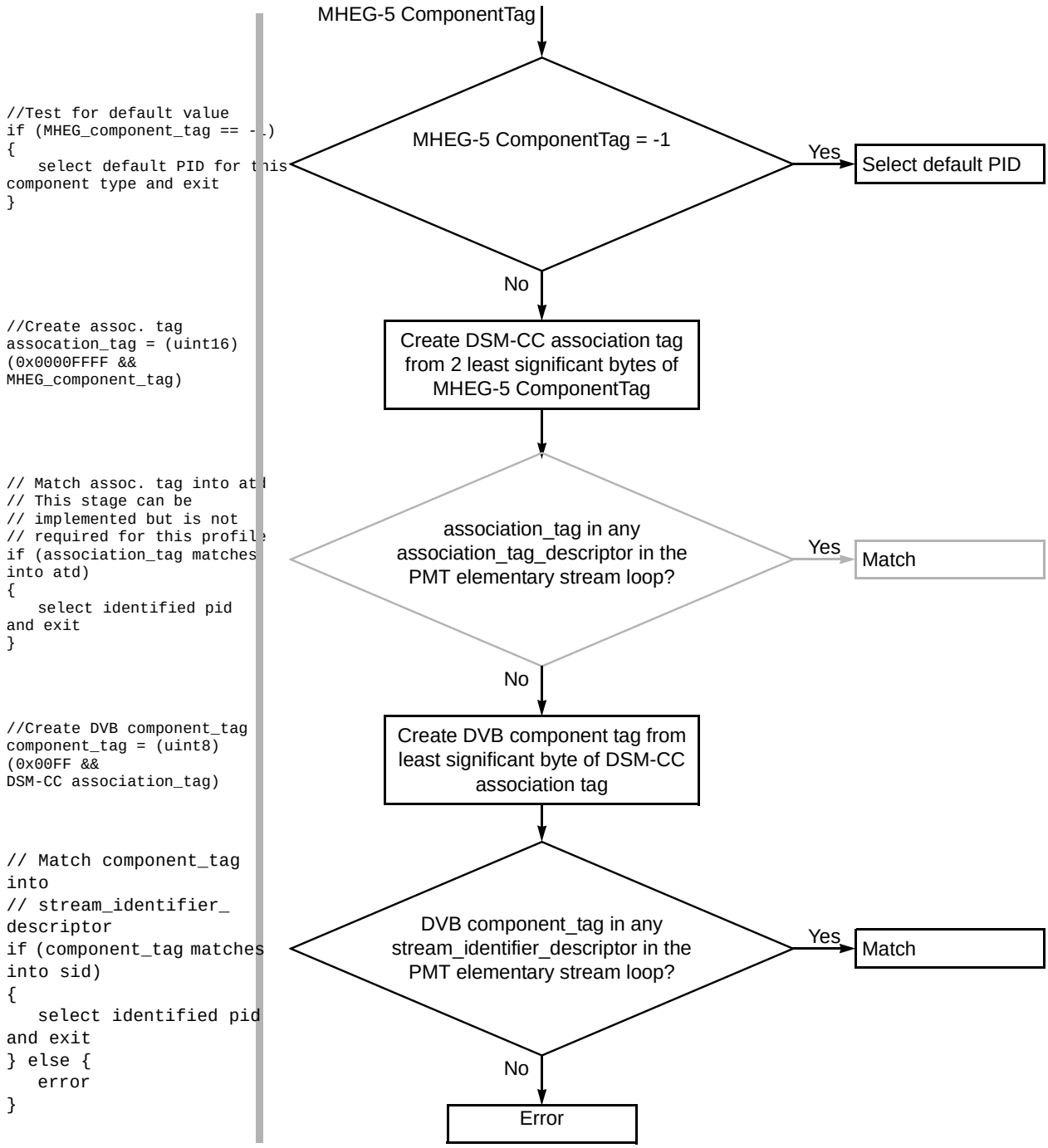



**Figure 33: Elementary stream mapping pseudo code and decision tree**

# 15.4 Caching

## 15.4.1 Transparent cache model

The default cache behaviour is "transparent", i.e. the functional behaviour of receivers shall be the same regardless of their cache provision.

So, in each of the following cases the engine shall ensure that it has an up-to-date version of the required file object:

- a group object is *prepared*;
- an ingredient object with referenced content is *prepared***;**
- the *content* internal attribute of an ingredient object with referenced content is set.

Even cache priority values (excluding zero) are reserved to indicate content or groups that are constant through the life time of the application. These can be retrieved from the cache without first verifying that the file version is current. See table 118.

## 15.4.2 Determining file version

There is no version number directly related to files (or other BIOP messages), the closest association is the **moduleVersion** in the DII that references the module that contains the BIOP message. Therefore, to ensure that a file is up to date the engine must determine that the *moduleVersion* for the appropriate module is current and reacquire if necessary.

The present document does not specify how an implementation determines that the module version is current. Valid approaches include:

- having a section filter dedicated to the acquisition of all DII messages, testing the *transactionID* field of DIIs of interest to see if the DII has been updated. If the DII has been updated then testing the *moduleVersion* field for the module of interest;
- sampling the DII *transactionID* field and module *moduleVersion* field when a file is requested.

These and other options have different implications for performance, use of resources, etc.

NOTE: The "update" flag (LSB of *transactionID* mapped into the table_id_extension field of DSM-CC section) is **not** a reliable indication of section update (as it may change more than once between receiver observations). Control message (DII/DSI) update is most reliably determined by inspecting the full *transactionID* field carried in the control message header.

### 15.4.2.1 Module acquisition

The object carousel effectively implements a "broadcast filespace". From the perspective of a running MHEG-5 service (and so the application author) requests are made for files without any real knowledge of how they are delivered. Crucially the MHEG-5 application has no concept of any versioning information and is reliant on other relevant functional blocks in the receiver to cope with this. This needs to be borne in mind when acquiring a Module (containing a requested file) from the broadcast stream.

Module acquisition from the broadcast stream is a two-step process:

1) Acquire DII and extract module download parameters.
2) Acquire module using extracted parameters.

Since these steps happen sequentially, it is possible for the module download parameters to change between the two steps – due to one of the objects within the module being updated. Continuing to use the original parameters will mean that the module is never acquired.

Thus, when acquiring a module it is important to continue to monitor the DII from which the download parameters were extracted and react to any changes by redefining any relevant filters. This ensures that filters setup to catch the DDB messages delivering the module remain appropriate.

## 15.4.3 Content cache priority

The value of ContentCachePriority in content references and SetData actions specifies the checks that the receiver must make before returning cached content and suggests how the content could be cached thereafter. A ContentCachePriority value is associated with a piece of content until overridden by another value.

See table 118.

**Table 118: Semantics for the allowed values of content cache priority**

| Cache priority | Semantic |
|---|---|
| Even values (excluding zero) | Even non-zero values of cache priority (2, 4 etc.) indicate that the object can be fetched from the local cache without reference to the broadcast stream. They also hint that the data is likely to remain static.<br><br>Application authors can use higher values to indicate content that they think it is more useful to cache. |
| Odd values | Odd values of cache priority (1, 3, etc., including the default 127) indicate that the receiver must verify that the file is current before using data from the cache and hint that the data is not expected to be static within the Application object's life time. Application authors can use higher values to indicate content that they think it is more useful to cache. |
| Zero | ISO/IEC 13522-5 [14] states that when an object is declared with ContentCachePriority of zero:<br><br>*"Specific value: 0 means caching is not allowed for external content data referenced by this Ingredient."*<br><br>Therefore, when the action SetData is targeted at an object with ContentCachePriority of zero, it is guaranteed to fetch the content from the broadcast stream rather than from any cache.<br><br>This functionality is introduced deliberately to bypass any receiver cache allowing an application to be synchronized with the broadcast carousel. It should not be used simply to ensure up-to-date content is retrieved: use an odd value for this purpose. |

## 15.4.4 Group cache priority

The following interpretation is placed on group cache priority:

- 0 transparently cached;
- odd transparently cached;
- even non-transparently cached.

The default group cache priority is 127 and hence transparent.

## 15.4.5 Cache validity

All information held in either a module and/or object cache shall only remain valid whilst the relevant object carousel remains mounted and so monitored.

For avoidance of doubt, changes to DSI messages shall not be considered an unmounting of the carousel.

This rule is provided since on remounting an object carousel it is possible (if unlikely) for objects to have changed in a way that is undetectable, i.e. version numbers to have been incremented such that they are the same as when the object was originally cached.

All information held in either a module and/or object cache shall only be valid within the context of the object carousel from which they were originally acquired.

This rule is provided since object names in object carousels are not globally unique and so it is possible that files with the same name but different content exist in different carousels.

The validity of any cached item is only dependent on the relevant object carousel and consequently is independent of the lifecycle of individual MHEG-5 applications that may be delivered as objects within that carousel.

Any item that is deemed as invalid shall be flushed from the cache.

A specific example of when the cache should be flushed is on selection of a new service since any associated object carousel will be unmounted as part of the service change process (see clause 8.1.4).

### 15.4.6 Dynamic carousel structure

The object carousel may change structure over time, i.e. both files and directories may be added or deleted. Also, modules are not guaranteed to carry the same objects over the lifetime of the carousel. Therefore receivers shall not assume that directory structures are static or that a given path will resolve always to the same object. All cached directory information shall be cached transparently. This means that before using an object that has been cached receivers shall validate the path to it.

NOTE: Validating a path does not necessarily mean downloading all elements in the path every time. For example, simply determining that none of the objects on the path have changed since it was last fully traversed is sufficient to confirm that the path itself has not changed.

## 15.5 Receiver demultiplexer resources

The present document places no upper limit on the number of elementary streams or sections used to transport object carousel data (including stream events). Receivers shall be able to support applications carried by any legal object carousel.

To ensure reasonable performance receivers shall allocate sufficient resources to acquire DSM-CC sections from at least four elementary streams to support a single MHEG-5 presentation. See also clause 17.16.1.

# 16 Name mapping

## 16.1 Names within the receiver

See table 119.

**Table 119: Names within the receiver**

| Name format | Use | Comment |
|---|---|---|
| rec://svc/def | ContentReference for a Stream object | This ContentReference identifies the receiver's **default service**, i.e. that most recently selected by a service change. See clause 14.2.5. |
| rec://svc/cur | | This ContentReference identifies the receiver's **currently selected service**. This may be different to the default since the last service change an application has explicitly set the Multiplex specified for a Stream object. See clause 14.2.5. |
| rec://svc/lcn/<LCN> | | This ContentReference identifies a service based on the associated "logical channel number" (or LCN). See clause 16.3.3.2. |
| rec://font/uk1 | Font attribute of Application class or OriginalFont attribute of Text class | This identifies the in-built receiver font described in clause 13.3.2. |
| ram://<name> | Name space for persistent storage. | See clause 14.6. |
| rec://htext/top | Anchor text for navigation boundary events within the HyperText class. | These Anchor texts identify that either the top or bottom of the HyperText object have been navigated to as described in clause 13.8.6. |
| rec://htext/bot | | |

## 16.2 MHEG-5 component tags

See clause 15.3.

## 16.3 Namespace mapping

When an application starts, it is assumed that a broadcast filespace has been mounted (via a single ServiceGateway object), and/or an application MMI session with a DVB CI module has been set up. The consequence is that there is either one or two unambiguous namespaces from which data may be retrieved.

The high-level API differentiates between three types of retrieved data:

1) objects that comply to the high-level API definition, i.e. MHEG-5;

2) the content (such as bitmaps or text) of those objects; and

3) streams (such as video and audio).

For accessing the application data on the server side, the DSM-CC Directory, File and Stream objects are used. Note that the server, in this context, does not have to be a physical server, but could be implemented, for example, as a broadcast carousel in a pure-broadcast topology.

Each file is either a Scene object, an Application object, or the content data of an Ingredient object. Each Scene object, Application object and content data is stored in a separate file.

### 16.3.1 MHEG-5 object references

MHEG-5 objects can be exchanged in two ways. Application and Scene objects are exchanged as a DSM-CC or DVB CI application MMI file object. All other objects are classed as Ingredients and are exchanged within another high-level API object, i.e. within and Application or Scene object.

MHEG-5 references objects by an ObjectReference, consisting of an optional byte string GroupIdentifier, followed by an integer, the ObjectNumber.

For the mapping on DSM-CC or DVB CI application MMI, the following additional rules are defined:

1) Each Application and Scene object shall have in its GroupIdentifier a byte string which maps on the name of the DSM-CC file which contains that object. These objects shall have their ObjectNumber set to 0.

2) Each application shall have exactly one Application object. That object shall be contained in a DSM-CC or DVB CI application MMI file object. Only one Application object shall be contained in each such file. See clause 9.3.4.2.

3) Ingredient objects may:

   - either leave out the GroupIdentifier, in which case it is assumed to be a string which maps on the name of a DSM-CC file which contains the object (Application or Scene) of which this object is a part; or
   - fill in the GroupIdentifier with such a string.

4) Ingredient objects shall have their ObjectNumber set to a value which is unique within that Group.

5) For the GroupIdentifier, the mapping rules defined in clause 16.3.2 apply.

#### 16.3.1.1 MHEG-5 content references

MHEG-5 has a separate way of referencing the actual content of objects belonging to the Ingredient class. This is done by way of a ContentReference which is simply an octet string. The format of this is defined by the mapping rules in clause 16.3.2.

#### 16.3.1.2 DSMCC Stream objects

Note that the mapping of ContentReference and GroupIdentifier allow references to both DSM-CC File and DSM-CC Stream objects. Although this is possible, applications shall not refer to DSM-CC Stream objects using a GroupIdentifier. GroupIdentifiers are always expected to refer to DSM-CC File objects.

The ContentReference of an MHEG-5 Stream object may refer to both DSM-CC File and DSM-CC Stream objects. If the ContentReference refers to a DSM-CC File object, the MHEG-5 Stream object shall have its Storage attribute set to memory. If the ContentReference refers to a DSM-CC Stream object, the MHEG-5 Stream object shall have its Storage attribute set to stream. ContentReferences of MHEG-5 Ingredients other than Stream objects must always refer to a DSM-CC File object.

### 16.3.2 Mapping rules for GroupIdentifier and ContentReference

#### 16.3.2.1 Identifier length limitations

GroupIdentifiers and ContentReferences become file names in the file system. GroupIdentifiers shall have names with lengths between 1 and 64 bytes. ContentReferences shall have names with lengths between 0 and 64 bytes.

#### 16.3.2.2 Case sensitivity

The BIOP::Name in the DSM-CC directory and service gateway messages (see clause 15.2.3.4) provides a *case sensitive* file name.

Therefore, an MHEG-5 group identifier or content references of "foo" will not match a file named "Foo". Indeed a directory might contain both "Foo" and "foo". However, clause 17.7 recommends that file names should be distinguishable on a case insensitive file system to ease development of applications.

### 16.3.2.3 Structure of file references

GroupIdentifiers and ContentReferences are composed of the following four components in sequence:

- source;
- path origin;
- path;
- filename.

Table 120 specifies the allowed contents of each of these components and their meaning.

**Table 120: Definition of reference components**

| Source | Path origin | Path (NOTE) | Filename | Meaning |
|---|---|---|---|---|
| "DSM:" | | | | The service gateway of the "Current carousel". See clause 8.1.5.2. |
| "~" | | | | Shorthand for "DSM:". |
| "CI:" | | | | The root of the file system provided by a CI module while application MMI session is open after a RequestStart has been sent by the module. See clause 14.9. |
| empty string "" | | | | Shorthand for "Current carousel" (see clause 8.1.5.2). Resolves to "DSM:" or "CI:" as appropriate. |
| | // | | | Root directory of the specified source. |
| | / | | | Shorthand for the path from "//" to the active application. |
| | | dir/ | | A component of a path to a sub directory.<br><br>The text "dir" is one or more characters long.<br><br>In the object carousel the text "dir" corresponds to a name component in the directory message binding where the bound object is a directory. Each "/" delimits a name component. |
| | | ../ | | "Move back up on directory level" (see clause 16.3.2.3.2). |
| | | | file | The name of the file.<br><br>See clause 16.3.2.1 for limitations.<br><br>In the object carousel the text "file" corresponds to a name component in the directory message binding where the bound object is a file, stream or stream event. |
| NOTE: Zero or more instances yet. | | | | |

#### 16.3.2.3.1 Resolution of file references

The process for resolving file references is equivalent to:

1) Expand any shorthand components:

   - an empty source expands to "DSM:" or "CI:";
   - the path origin "/" expands to a path such as "//weather/today/";
   - the source and path origin together might be "DSM://weather/today/";

   Therefore:

   "/cloud" might become "DSM://weather/today/cloud".

2) Allow each "../" component to consume the path component preceding it.

   Therefore:

   "/../tomorrow/app" might first become "DSM://weather/today/../tomorrow/app" before it collapses to "DSM://weather/tomorrow/app"

#### 16.3.2.3.2 Notes on resolving "../"

The general process for resolving "../" is shown above. In addition:

- the process for "../" components to consume super directory components operates left to right;
- there may be many "../" components in a path and these are not required to be contiguous;
- the reference is unresolvable (and hence invalid) if the "../" components imply "climbing above" the file system root.

  Therefore:

  - "DSM://A/B/../F/../X/Y" collapses to "DSM://A/X/Y"; and
  - "DSM://A/../../X/Y" is unresolvable

#### 16.3.2.3.3 Maximum reference length

The maximum length for a reference is 64 bytes (see clause 11.11.5).

This limitation applies to the length of the reference after complete resolution (i.e. the final "DSM://weather/tomorrow/app" form).

Implementers should be aware that the 64-byte limitation does not apply to a partially resolved reference (e.g. "DSM://weather/today/../tomorrow/app"). These may require significantly more memory.

#### 16.3.2.3.4 Character codes

For the avoidance of doubt the character codes used for the components of a reference are as follows:

- "DSM:" is 0x44534D3A
- "CI:" is 0x43493A
- "//" is 0x2F2F
- "/" 0x2F
- "~/" 0x7E2F

For example, if the fully resolved reference for the current application object is "DSM://dir1/foo" then:

"//bar" resolves to "DSM://bar";

"/bar" resolves to "DSM://dir1/bar";

"~/bar" resolves to "DSM://dir1/bar";

"~/../bar" resolves to "DSM://bar";

"//../bar" is an invalid reference.

#### 16.3.2.4 Shorthand notation

A second meaning for the abbreviation "/" meaning "absolute Path Origin" was defined by DAVIC (DAVIC 1.4.1 Part 09 [16]). This **is not permitted** as it cannot be distinguished from the meaning of "/" defined above.

### 16.3.3 URL formats for access to broadcast services

#### 16.3.3.1 Numerical format

DVB ETSI TR 101 812 [19] defines a URL format for locating DVB entities. To reference a DVB service it has the following numerical form:

```
dvb://<original_network_id>.[<transport_stream_id>].<service_id>
```

The textual_service_identifier form of reference to a service is not supported.

The values of <original_network_id>, <transport_stream_id> and <service_id> are represented as hexadecimal strings without any "0x" prefix (for example, "4d2e").

See ETSI TR 101 211 [4]. Services are unique within the scope of a single original network ID. So, transport stream ID is not required to uniquely specify a service within an original network. The transport stream id (0x1234 in this case) is optional in URLs and can be left empty.

For example, URLs specifying a service might be of the form:

dvb://abcd.1234.5679

or

dvb://abcd..5679

#### 16.3.3.2 Logical Channel Number format

Some networks support the use of a "Logical Channel Number" to identify a "channel" from the viewer perspective, e.g. Logical Channel Number 4 is "Channel Four". This is particularly useful when trying to provide a single reference to a service which is composed of a number of regional variants, e.g. "ITV1".

Note: The means for association of a value of LCN to a receiver channel is not defined in the present document.

A reference to a service may be made by the use of its Logical Channel Number using the following syntax:

```
rec://svc/lcn/<LCN>
```

where <LCN> is the Logical Channel Number of the service encoded textually as a decimal integer with no leading zeros. Therefore, a URL specifying "Channel 4" (LCN 4) would be:

```
rec://svc/lcn/4
```

Where a network does not support Logical Channel Number the URL can not be resolved.

#### 16.3.3.3 Handling duplicate services

In terrestrial networks the same service may be available to the receiver on more than one transport stream. This can lead to a particular service URL matching more than one service in the receiver's channel list regardless of whether the Numerical or Logical Channel Number format is used. The following algorithm is to be used unless otherwise stated in a localized profile of this Profile:

```
if (one of the candidate services is in current multiplex)
   return service_index of this service;
else
   return service_index of the candidate service nearest to
   the start of the receiver's ordered channel list;
```

# 17 MHEG-5 authoring rules and guidelines

## 17.1 Introduction

This clause describes the measures that should be taken by the broadcaster to ensure good application behaviour. It also provides tutorial illustrations of certain advanced techniques to clarify their use and behaviour.

The techniques described in clauses indicated as being mandatory have been found to prevent both confusion of the user and degradation of overall system performance and thus shall be observed by application authors. Receiver implementations shall not attempt to enforce these rules.

NOTE: This part of the present document is not necessarily exhaustive. It is envisaged that this clause is just the starting point for documents that record the accumulated experience of the industry.

### 17.1.1 Avoiding confusion with navigator functions

#### 17.1.1.1 Use of user inputs (mandatory)

##### 17.1.1.1.1 Requirement

The user shall be able to "surf" through services with MHEG-5 applications using either the Prog+/- keys or the number keys without a change in the user interface behaviour.

##### 17.1.1.1.2 Selecting and entering

Two concepts are introduced: *selecting* the MHEG-5 application and *entering* it.

When the user first navigates to a service that is wholly or partly implemented as an MHEG-5 application it is "selected". If the user subsequently interacts with the MHEG-5 aspects of that service (using a key available from input event register 3) they are deemed to have "entered" it.

Until the user "enters" the application:

- **the application shall only use input event register 3;**
- **the application shall ensure that the InteractionStatus of all interactibles shall be False.**

An exception to the above is that through use of "Persistent storage" an application can launch a second application that transitions immediately to an interior "entered" Scene. This shall only be done after the user has "entered" the first application.

##### 17.1.1.1.3 Leaving

Applications shall provide a means of returning to the "selected" state. Such means should be clear and simple for the viewer to use.

## 17.2 Use of the "Text" and "Cancel" functions

The "Text" user input event has a specific meaning that should be observed by all applications to avoid confusing the user. It means "toggle" the visibility of the MHEG-5 aspect of the service.

The "CancelKeyFunction" user input event has a specific meaning that should be observed by all applications to avoid confusing the user. It means "go back" to a previous or higher page or item of content, either in a hierarchical or historical manner, or terminate the "interaction" state of an interactible (see also clause 11.13.6).

If the "CancelKeyFunction" is invoked at the top level of an application, or when there is no historical content to return to, the application should be made no longer visible to the user (i.e. it would appear to the user that the application has been terminated).

### 17.2.1 The traditional "teletext" key

In figure 34 "Text" toggles between conventional TV and a full-screen MHEG-5 application. This is very similar to the behaviour of today's "teletext" button.

An available elaboration is that a single "teletext" service can be shared by a number of TV services.

#### 17.2.1.1 Entering

In this case the first Scene of the "Auto boot broadcast application" associated with the service produces no visible effect. This first "invisible" Scene responds to "Text" either by transitioning to another Scene within the same application or by launching a different application.

#### 17.2.1.2 Leaving

If the application is part of the service then the "return" when "Text" is used a second time can be implemented in various ways:

- TransitionTo the first Scene;
- quitting the application (which will automatically be restarted on the first Scene);
- launching the application (which will automatically start on the first Scene);
- service change (resident program) to the information application.

If the information application is **not** part of the service (i.e. the "invisible" application launched it) then the options are for the visible application to:

- launch the invisible application;
- service change (resident program) to the invisible application.

If there is a one-to-one relationship between the TV service (with invisible "springboard" application) and the information service then the return link can be explicitly coded into the application.

If more than one TV service directs its viewers to a single information service, then each of the "invisible" applications should store appropriate information in persistent storage to allow returning to the point of origin (see clause 14.6).

See figure 34.

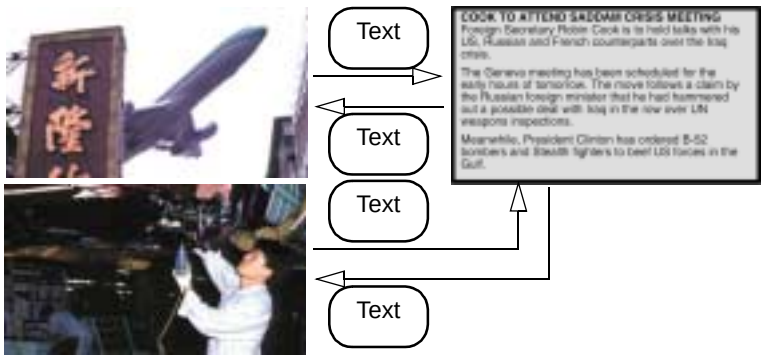



**Figure 34: "Text" accesses "teletext"**

### 17.2.2 Accessing additional programme information

In figure 35 the service by default has a visual prompt that more information is available. Using "Red" reveals the additional information. Using "Text" restores the original presentation (alternatively a coloured key might be used to take the viewer back to the original presentation and the "Text" key might take the viewer to a full-screen text service).

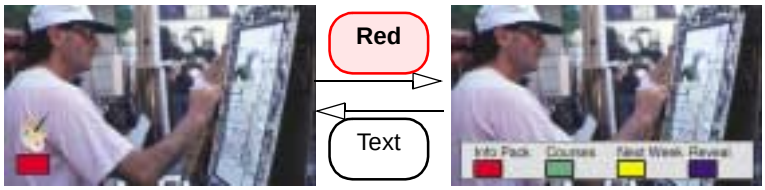



**Figure 35: "Text" provides more information**

Exactly how such visual prompts are used is the broadcaster's choice. For example, this graphic might be visible throughout the programme. Alternatively, it be shown for short periods e.g. after the programme is first selected and at the end of each "item" within the programme.

### 17.2.3 "Text" has no effect

In figure 36 the TV service has no MHEG-5 application. In this case "Text" is not required to have any effect. However, the receiver may **optionally** provide a response.

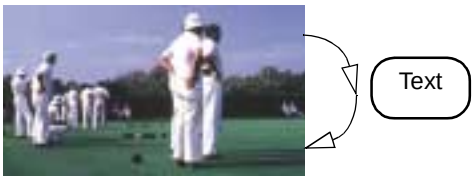



**Figure 36: TV with no MHEG-5 application**

In figure 37 a predominantly MHEG-5 service has no related TV service revert to. Alternatively, the application may have been launched by a user channel change and so has no knowledge of any previous service to which it should return. In either case "Text" cannot return the user to a "logical" TV service.


**Figure 37: MHEG-5 service has no TV alternate**

In the case shown in figure 37 it may be appropriate for the application to toggle between "selected" and "entered" conditions (even if there is no visible change in the display) to allow the user to navigate to another service using the numeric keys.

### 17.2.4 On-screen prompts

The "CancelKeyFunction" user input event is used to "go back" to a previous or higher page or item of content. If the application is displaying an interactive on-screen prompt (such as a "press red" prompt) and this user input event is invoked, the application should remove the prompt. It is also recommended that such prompts time out appropriately.

# 17.3 Use of stream decoders

## 17.3.1 Number of decoders

Receivers conforming to *ETSIEngineProfile1* are modelled as providing just one of each of the following decoders:

- MPEG video or still picture;
- MPEG audio;
- DVB subtitle.

NOTE: (Mandatory): To obtain deterministic behaviour, authors shall not build applications that attempt to activate more than one of each type of decoder, e.g. applications shall Stop a running Stream object using MPEG video before Running a Bitmap object with MPEG I-frame content.

See clause 14.7.2.

## 17.3.2 Visible area

Typically a 5 % border region is lost due to monitor overscan. This leaves a central 632 x 518 area of the graphics plane which will normally be visible provided that the user does not configure their equipment in an unusual way. For example, additional area may be lost if a 4:3 service is "zoomed" to fill a 16:9 display.

See figure 38.

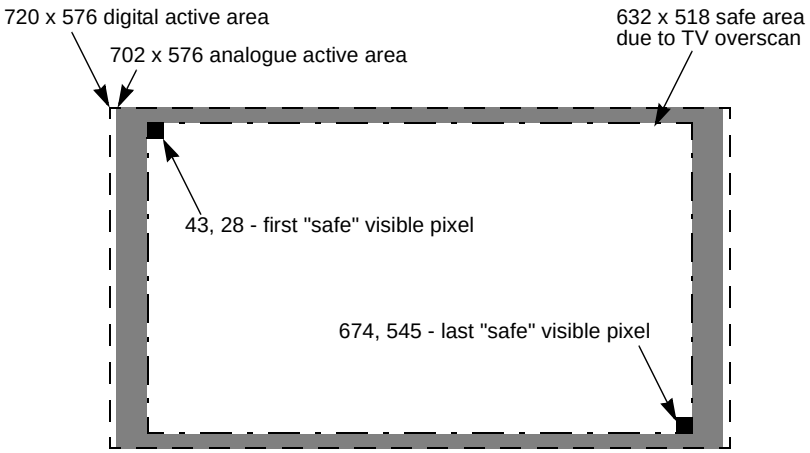



**Figure 38: Authoring "safe" area**

NOTE: These suggested figures for safe area have not been verified. Authors should use their own experiments to develop their own rules for the safe areas that apply to their circumstances.

## 17.3.3 Conflicts between subtitles and MHEG-5 graphics

Not all receivers are able to present subtitles and other MHEG-5 graphics together effectively.

Where a receiver is not able to present the two together the MHEG-5 application will not start whilst subtitles are being presented. When an application is being run, therefore, the author may assume that:

- there are no subtitles being displayed; or
- the platform is able to support simultaneous subtitles and MHEG-5 presentation.

The author may choose to suppress the subtitles for compositional reasons. Calling the resident program SetSubtitleMode(false) will ensure that subtitles are not visible even on receivers capable of simultaneous presentation. SetSubtitleMode(true) will re-enable presentation if it has been requested by the viewer (see clause 14.3.3).

## 17.4 Aspect ratio

### 17.4.1 Inheritance of video (mandatory)

Applications that inherit a default video stream and continue to present it full screen shall not change the aspect ratio of the presentation until the application has been "entered" (see clause 17.1.1.1), This means that such an application must begin with a Scene whose AspectRatio is undefined.

### 17.4.2 MHEG-5 only services

Where there is no aspect-ratio-sensitive content in an MHEG-5 Scene, the Scene's AspectRatio attribute should be left undefined. This allows the receiver to show the Scene filling the screen on both 4:3 and 16:9 displays.

Scenes that do have aspect-ratio-sensitive content may use an explicit AspectRatio. Where possible, such Scenes will be displayed in the requested ratio. However, authors should avoid frequent aspect ratio changes as many receivers take a finite time to adjust the display and the transition may not be smooth. Also, correct representation of 16:9 Scenes on 4:3 displays is unlikely to be possible.

### 17.4.3 I-frames

I-frames are combined with other MHEG-5 graphics without any decoder format conversion. 4:3 I-frames will therefore appear in the correct aspect ratio if the containing Scene is 4:3. 16:9 I-frames cannot be displayed without distortion on most 4:3 displays.

## 17.5 PNG bitmaps

### 17.5.1 Interlaced formats

Authors should be aware that good approximation of colours with dithering requires much greater receiver resources when decoding PNG bitmaps using interlace methods other than 0 (no interlace). As a consequence receivers may produce pictures of much lower quality when an interlace method (such as Adam7) is used.

## 17.6 Missed events

Applications are not guaranteed to receive all events. For example, stream events and timer events may be missed while an application is paused as a consequence of losing priority access to the display. See clause 14.8.

## 17.7 File naming

### 17.7.1 Name length (mandatory)

The length of file references is limited. See clause 11.11.5.

### 17.7.2 Name character coding

To ease development of applications on computer platforms using traditional file systems it is suggested that the set of character codes used in file names is in the range 0x21 to 0x7E but excluding the following codes as they have special meaning, or cause other problems, on some platforms: 0x22, 0x27, 0x3A, 0x3B, 0x5C (double quote, single quote, colon, semi-colon, backslash).

### 17.7.3 Case sensitive file names

File names are case sensitive (see clause 16.3.2.2).

Application developers should be aware that several desk top computer operating systems are insensitive to the case of characters in filenames. Also, some operating systems fail to accurately display the case of file names.

Using filenames that can be distinguished on non-case sensitive file systems will help authoring and content interchange.

### 17.7.4 File names in persistent storage (mandatory)

The <name> part of the file name "ram://<name>" used to access files in persistent storage (see clause 14.6) shall be managed to avoid accidental file name collision between the applications of different service providers.

The length of <name> is defined in see clause 14.6.1. The first characters of these names are allocated to multiplex operators as shown in table 121 where "xxxxx" indicates characters that may be allocated by the multiplex operator to services and applications as they choose.

**Table 121: Format of file names for persistent storage**

| <name> format | Multiplex operator |
|---|---|
| BBCxxxxx | BBC |
| D34xxxxx | Digital 3/4 |
| SDNxxxxx | SDN |
| BDBxxxxx | BDB |
| CCIxxxxx | Crown Castle |

Where an application (or co-operating applications) is (are) delivered by more than one multiplex operator the application author shall be responsible for liaising with the multiplex operators to obtain a <name> allocated within the space of one operator.

Further names "roots" may be allocated following mutual agreement between all the multiplex operators.

## 17.8 Text encoding

### 17.8.1 Mark-up

Mark-up coding for Text objects is described in clause 13.6 is functionally equivalent to a small subset of HTML re-coded to improve transmission efficiency. A simple translation process between the two formats is possible.

See table 122.

**Table 122: Text object mark-up codes (Sheet 1 of 2)**

| HTML mark-up | Broadcast mark-up codes |
|---|---|
| (no equivalent) | 0x09 (NOTE 1) |
| (no equivalent) | 0x20 (NOTE 2) |
|   | 0xC2 0xA0 |
| <BR> | 0x0D |
| <P> | |
| </P> | 0x0D 0x0D |
| <B> [text in bold] </B> | 0x1B 0x42 0x00 [text in bold] 0x1B 0x62 |
| <FONT COLOR="#rrggbbtt"> [coloured text] </FONT> | 0x1B 0x43 0x04 0xrr 0xgg 0xbb 0xtt [coloured text] 0x1B 0x63 |
| < | 0x3C |
| > | 0x3E |
| & (NOTE 3) | 0x26 |
| <A HREF="tag bytes"> [anchor text] </A> | 0x1B 0x41 0xnn tag_bytes [anchor text] 0x1B 0x61 |
| <body<br>bgcolor=colour<br>text=colour<br>link=colour<br>vlink=colour | 0x1B 0x44 0xnn body_attr_bytes 0x1B 0x64<br>HyperText object BackgroundColour<br>HyperText object TextColour<br>body_attr_bytes anchor_colour<br>body_attr_bytes visited_anchor_colour |

**Table 122: Text object mark-up codes (Sheet 2 of 2)**

| HTML mark-up | Broadcast mark-up codes |
|---|---|
| `alink=colour>` | `body_attr_bytes` active_anchor_colour |
| NOTE 1: Tab characters have meaning clause 13.5.9. | |
| NOTE 2: All space characters are significant. | |
| NOTE 3: And so on, i.e. the HTML "named character entities" can be directly represented with a simple character code where they exist in table 78. | |

### 17.8.2 Text flow control

Authors should note that it is optional for receivers to implement certain of the text flow modes, see clause 13.4.2. Applications, should not use these modes or should give acceptable behaviour on receivers that instead implement one of the alternative modes listed in table 68.

### 17.8.3 Width of row of characters

Certain characters may have a representation that extends beyond their logical width. This can potentially result in a partially rendered character if such a character was the last in a line of text to render *and* if the logical width of this line is very close (or even equal) to the available width. However, whilst this is a potential hazard for content providers no special behaviour is expected of engines conformant to this Profile. This is because for all such known characters the intended usage is in combination with other characters, i.e. they should always be directly followed by another character and so should not appear as the last in a line of text to render.

## 17.9 Reference checking

### 17.9.1 Application design issues when checking references

Where operations may take significant time to complete (e.g. loading infrequently broadcast content) applications can be designed to reduce the length of time that they "block" and to enable the user to change their mind while content is loading.

The main tools here are the use of the CheckContentRef or CheckGroupIDRef resident programs. These can be used to asynchronously check the availability of carousel files.

#### 17.9.1.1 Preloading is not mandatory

A side effect of the CheckContentRef or CheckGroupIDRef resident programs is that they may load the referenced file into receiver memory. If the receiver has done this loading subsequent SetData or TransitionTo actions may complete much more rapidly. However, as the preloading behaviour is not mandatory some receivers may still block for a significant time at this stage.

#### 17.9.1.2 Stopping a reference check

The application can be designed to allow the user to change their mind while the CheckContentRef or CheckGroupIDRef resident programs are asynchronously running (for example, to give behaviour analogous to selecting a different link on a web browser or pressing "stop").

One authoring approach is to use the Stop action to abandon the previously invoked resident program and then use Fork to start another instance of this resident program. Due to the asynchronous behaviour of the forked program it is possible that the Stop action will occur after the program has naturally completed its processing but before the AsyncStopped event has been processed. In this case an application may execute sequentially Stop then Fork actions and immediately receive an AsyncStopped event. In this case the application programmer is responsible for determining which instance of the forked resident program produced the event. In the example given below this is done by examining the parameters returned by the resident program when it completes. Other approaches are possible. For example, more than one resident program object might be used.

## 17.9.2 Code example

This clause illustrates (by way of an annotated code example) the expected use of the reference checking resident programs defined under clause 11.10.

**This example relies on particular engine behaviour described in clause 11.10.13, i.e. that a forked process will not modify variables it shares with an application while that application is executing a LinkEffect.**

```
//the CheckContentRef program
//accepts ref-to-check, returns ref-valid-var &
ref-checked-var.
//need one resident-program object per concurrent
check.
```

The instance of the resident program object and the variables used to communicate with it.

```
{:ResidentPrg 1
   :Name "CCR"
}

//variable for ref-to-check input value
{:ObjectRefVar 2
   :OrigValue ("/newscene.mheg" 0)
}

//variable for ref-valid return value
{:BooleanVar 3
   :OrigValue true
}

//variable for ref-checked return value
{:ObjectRefVar 4
   :OrigValue ("" 0)
}

//variable for fork-succeeded return value
{:BooleanVar 5
   :OrigValue true
}
```

This is where it all starts, possibly a response to a user input.

```
//...link off some event that causes reference to
be checked
{:Link 10
:EventSource ??
   :EventType ??
   :EventData ??
   :LinkEffect {
      //stop any previously invoked fork of the
resident program
      :Stop(1)
      //if required set the ref-to-check variable
here

      //invoke the check reference resident
program
      :Fork( 1 // object number of the resident
program
                  5 // fork-succeeded boolean
variable
                  :GContentRef :IndirectRef 2 //
reference to be checked
                  :GBoolean :IndirectRef 3 //
ref-valid variable
                  :GContentRef :IndirectRef 4 //
ref_checked variable
                    )
   }
}
```

When the resident program completes, it generates an AsyncStopped event which this link processes.

```
{:Link 11
   :EventSource 1
   :EventType AsyncStopped
   :LinkEffect {
      //test that the fork of the check object
resident program
      // was successful. If it was go on to other
tests before
      // ultimately going on to transition to
another Scene
      :Activate 12
      :TestVariable( 5 1 :GBoolean true )
      :Deactivate 12
   }
}

// This link fires if the test of the
ForkSucceeded variable yields true.
// It confirms that the resident program
completed successfully.
// Go on to test if this is invocation of the
resident program
// we had in mind by comparing the returned
checked object
// ref against the object ref most recently
passed to the resident
// program
{:Link 12
   :EventSource 5
   :EventType TestEvent
   :EventData true
   :LinkEffect {
      :Activate 13
      // Compare the returned object ref (4) with
the
      // reference we most recently asked to be
checked (2)
      :TestVariable( 2 1 :GObjectRef :IndirectRef
4 )
      :Deactivate 13
   }
}

// If this link fires it means that the check
reference resident
// program completed its fork successfully and
was checking the
// correct reference.
// Now see if the file was found to be available.
{:Link 13
   :EventSource 2
   :EventType TestEvent
   :EventData true
   :LinkEffect {
      // Is the ref-valid returned variable (3)
true
      :Activate 14
      :TestVariable( 3 1 :GBoolean true )
      :Deactivate 14
   }
}
```

```
// This link fires if check reference program
returned true
// This is the culmination of links 11, 12 & 13.
In effect
// we have implemented:
// if( fork returned OK &&
//       we've checked the right reference &&
//       the file referenced is available )
//       {
//          TransitionTo( the next Scene )
//    }
{:Link 14
   :EventSource 3
   :EventType TestEvent
   :EventData true
   :LinkEffect {
      :TransitionTo(:IndirectRef 2)
   }
}
```

## 17.10 Dynamically updated content

MHEG-5 operates a "pull model" for acquisition of content. Where applications require to use the latest version of changing content they must request the content after it has changed. It is the author's responsibility to determine when to request the content (e.g. through the use of StreamEvents, periodic polling, etc.).

See clause 15.4.

## 17.11 Stream events

See clause 15.2.4.

Application authors should be aware that there is no absolute guarantee that a "do-it-now" stream event will reach the application. The event may be lost, for example, due to transmission errors.

NOTE: (Mandatory): The BIOP StreamEventMessage eventId and eventName pairing shall not be altered while an application(s) is waiting for an event to be signalled. A change in the event id/name pairing (or deletion of the event) is not detected by the receiver for outstanding events. Modification may result in the receiver waiting indefinitely for an event that will never be broadcast.

## 17.12 User input events

### 17.12.1 Obtaining user input from an application with no Scene (mandatory)

According to clause 11.8, engine events relating to the members of the "Register 3 group" are generated only when there is an active Scene object. Authors may wish to create an application which does not transition to a Scene until a "Register 3 group" key is pressed - this is no longer possible. Instead the application should transition to a simple transparent Scene and wait for an EngineEvent or UserInputEvent related to the required key.

### 17.12.2 Use of user input related engine events.

An application may be driven from two classes of user input related asynchronous events:

- UserInputEvents are generated by the current Scene when there is no active interactible;
- EngineEvents (TextKeyFunction, RedKeyFunction, etc.) are generated by the current application. They are always generated even if there is an active interactible (however, see clause 17.12.1).

When both are generated the EngineEvent is generated before the UserInputEvent

## 17.13 Undefined behaviour (mandatory)

Some aspects of MHEG-5 engine behaviour are undefined both in the ISO/IEC 13522-5 [14] standard, and in this Profile. Authors shall avoid reliance on any one implementation and the behaviour observed on that platform.

### 17.13.1 Synchronous event processing

Certain Elementary Actions involve the raising of more than one synchronous events during execution, most notably the Launch, Spawn and TransitionTo actions. How these events are handled, and which links will fire is not well defined in ISO/IEC 13522-5 [14] and has led to receiver implementations varying in behaviour. Some receivers will queue all of the events, and others will mark Link objects as fired at the point the event was raised.

Authors should beware of the following example:

As part of an application:

```
{:rectangle 20
   ...
}
{:link 10
   :eventsource 20
   :eventtype isavailable
   :linkeffect (
      …
   )
}
{:link 11
   :eventsource 20
   :eventtype isrunning
   :linkeffect (
      …
   )
}
```

The Link objects 10 and 11 may or may not fire, depending on the receiver implementation.

### 17.13.2 Order of parallel links firing

If two Link objects source the same object and event, then the order of execution of the two LinkEffects is undefined. If one of those LinkEffects includes a context switch (Launch, TransitionTo etc.) then the second LinkEffect may never run depending on the receiver implementation.

## 17.14 Use of Call and Fork with ResidentPrograms

ResidentProgram objects possess the unusual property of being in the Active/Running state only whilst their procedural code is actually being executed.

Step 2 of the descriptions for the Call and Fork actions in clause 14.4 of the MHEG-5 specification (ISO/IEC 13522-5 [14]) stipulates, "*If the Program is active, disregard this action.*" Therefore, a Fork or Call to a ResidentProgram instance immediately following a Fork to the same ResidentProgram instance but before the AsyncStopped event has occurred will be ignored.

## 17.15 Catching failure of TransitionTo, Launch and Spawn

### 17.15.1 Background

It is good authoring practise to use LockScreen prior to a transition to another Group (using TransitionTo, Launch or Spawn).

However, authors should recognize that the expected transition may not succeed.

For example, the target object specified in a TransitionTo may not be available (for example, because of an OC construction error) or may not be loadable by the receiver (due to memory limitations). In this case the receiver will continue to process actions in the current Scene. If the screen is still locked the user will perceive that the receiver has locked-up.

NOTE: (Mandatory): **Authors shall write defensively to ensure that the application will continue to interact with the user if an expected transition fails.**

Typically this defensive coding will include placing one (or more) UnlockScreen elementary actions following the TransitionTo, Launch or Spawn elementary actions.

# 17.16 Elements of the object carousel

## 17.16.1 DIIs and elementary streams

The following recommendations are made to aid good receiver performance:

- Ensure that eight or fewer DIIs are required to be monitored to achieve any single presentation.
- Ensure that the DSM-CC sections that require monitoring for any single presentation are distributed over four or fewer elementary streams.

## 17.16.2 Directory structure (mandatory)

- The fully resolved path to a file shall be ≤ 64 bytes (see clause 16.3.2). This places limitations on the length of file and directory names and the depth of directory structure.

## 17.16.3 Timeouts (mandatory)

Timeouts are encoded for both DII messages and Modules (see clause 15.2.2.5 and clause 15.2.5). These values have no defaults and must be explicitly encoded when constructing the carousel. The encoded value must be chosen to allow sufficient time to download the relevant broadcast message(s) whilst ensuring that any error in carousel construction does not leave the receiver hanging unnecessarily.

## 17.16.4 Examples of object carousels

Figure 39 illustrates an object carousel that is distributed over three elementary streams belonging to the same service.

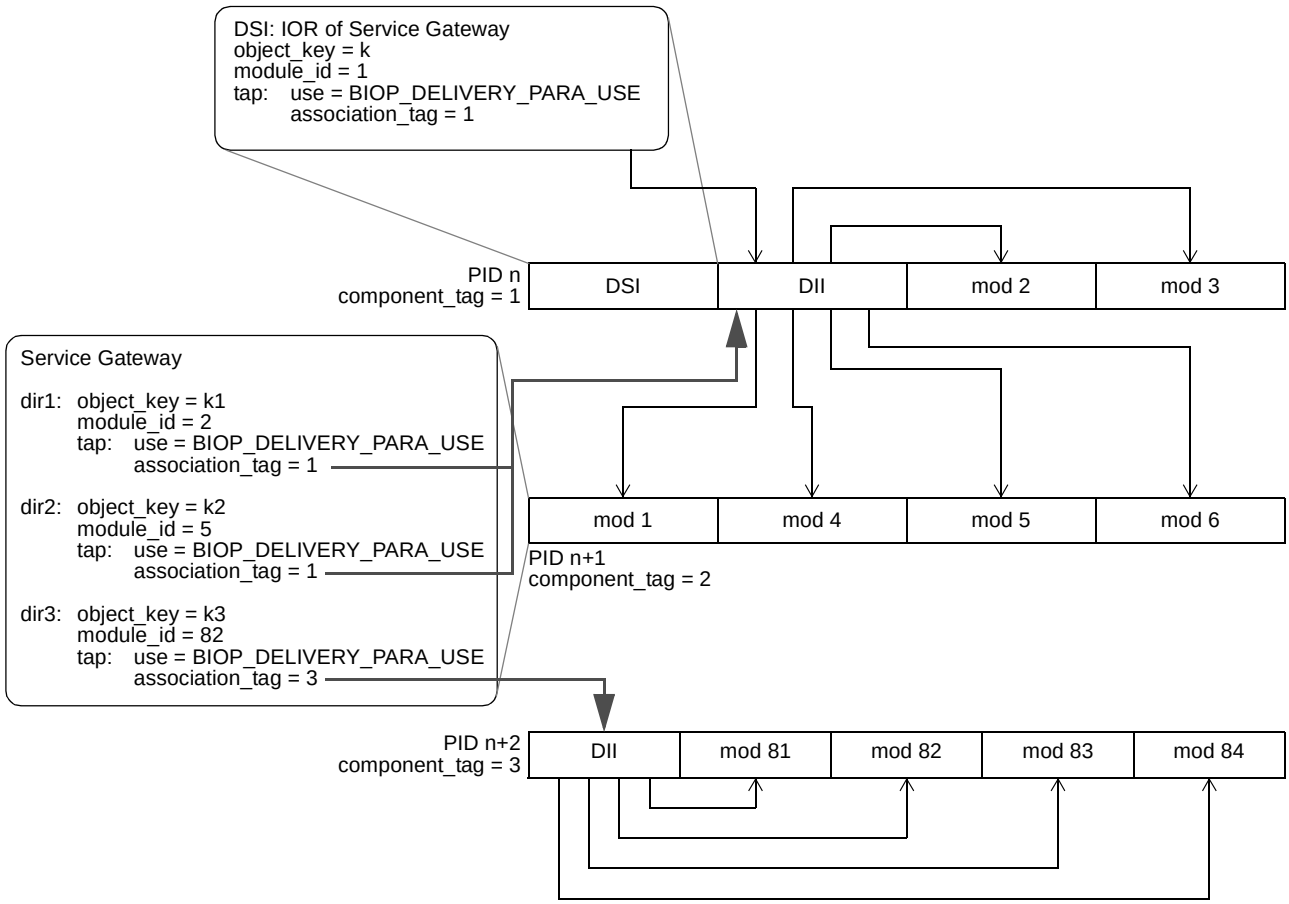



**Figure 39: First example carousel**

The DownloadServerInitiate (DSI) message is carried on the first elementary stream. It contains the object reference that points to the ServiceGateway. The tap with the BIOP_DELIVERY_PARA_USE points to a DownloadInfoIndication (DII) message that provides the information about the module and the location where the module is being broadcasted. In the example, the ServiceGateway object is in the module number 1 that is carried on the second elementary stream (indicated by a BIOP_OBJECT_USE tap structure in the DII message).

The ServiceGateway object is a root directory that, in this example, references three subdirectories. Taps with BIOP_DELIVERY_PARA_USE are used in the object references of the subdirectories to provide links to the modules via the DownloadInfoIndication (DII) message. The two first subdirectories "dir1" and "dir2" are referenced in the DII message that is carried in the first elementary stream. The third subdirectory is referenced in the DII message carried in the third elementary stream.

It is important to note that the third elementary stream may originate from a completely separate source than the first two elementary streams. The directory hierarchy and objects contained in the third elementary stream are "mounted" in the root directory by providing the "dir3" directory entry with the appropriate location information.

This type of structure could be used, for example, in a national information service that contains some regional parts. The common national parts could be carried in this example case on the two first elementary streams that are distributed unmodified in the whole country. The regional parts are carried in the third elementary stream that is locally inserted at each region. From the application's point of view, the common national parts are in the "dir1" and "dir2" subdirectories while the regional parts are in the "dir3" subdirectory.

Another example where this type of structure could be used is if the service contains multiple independent applications. In this case, each application could be placed in its own subdirectory and these subdirectories might be carried on different elementary streams.

The first example carousel does not reveal the role of the transactionId. This part of the DSM::Tap augments the information on how to locate the required DII. The assocTag first identifies the elementary stream carrying the DII, the transactionId then discriminates between DIIs on the same elementary stream, as illustrated in figure 40. To aid acquisition of DIIs the least significant two bytes of the transactionId field are reproduced in the table_id_extension field of the MPEG section carrying the DII.

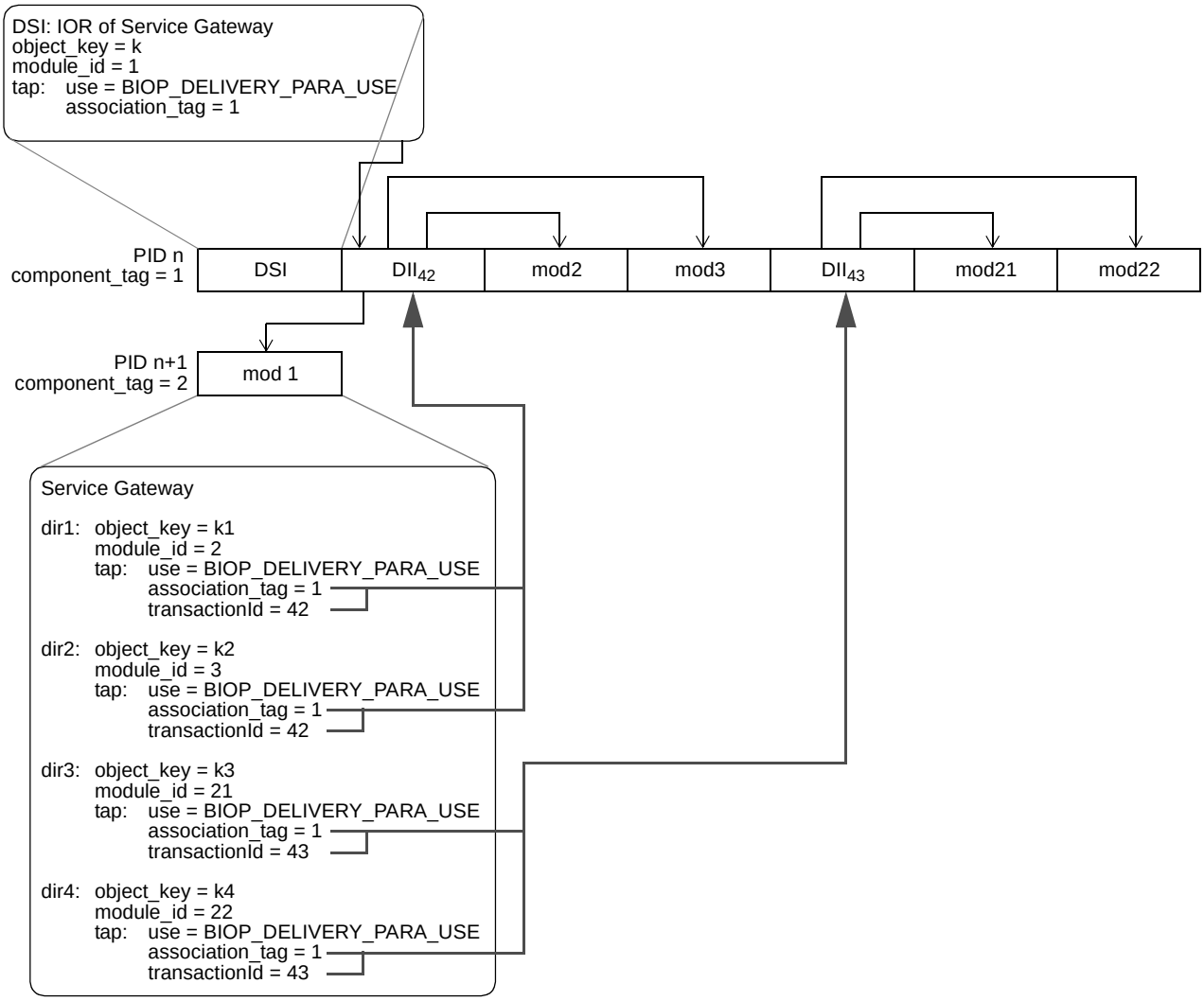



**Figure 40: Second example carousel**

# 17.17 Possible uses of persistent storage (informative)

## 17.17.1 Start-up scene reference

An application can store the ObjectReference of a specific scene within another application prior to launching that application. The launched application can retrieve the ObjectReference and TransitionTo the specified scene.

This allows transitions between scenes in two different applications. Without this, applications could only be launched to start at their root scene.

### 17.17.2 Return application reference

An application can store an ObjectReference to itself before launching another application. This allows the launched application to return to the "calling" application (although the Spawn action might be a better way of doing this).

### 17.17.3 Return application start-up scene reference

A scene can store an ObjectReference to itself before launching another application. This allows the "calling" application to TransitionTo the "calling" scene if it is restarted.

## 17.18 Hints and tips for good authoring

This clause presents ideas on how to create better performing applications. It is primarily aimed at educating authors, however, it should also help to inform receiver implementers when trying to improve receiver performance.

### 17.18.1 Structure of the file system

#### 17.18.1.1 Directories

- Structure applications into some sort of directory hierarchy.
- Use multiple directories rather than a few directories that contain large numbers of files: small directories are easier to cache and quicker to search when trying to locate a file.
- Place more-frequently accessed files earlier in the directory list: they may be found more quickly when searching.

#### 17.18.1.2 File names

- Use the shortest file name that is practical: short file names reduce broadcast bandwidth, reduce receiver memory requirements and can be tested more quickly.

### 17.18.2 Structure of the object carousel

#### 17.18.2.1 Placing associated objects in a module

Closely related objects (e.g. objects of one MHEG-5 scene) should be put in the same module (see clause 15.2.7).

#### 17.18.2.2 Cache priority and modules

If a Scene contains an object that has an initial CCP (content cache priority) of zero this may delay Scene start-up as it may force the receiver to reload a module and hence delay loading of other objects. This problem can be addressed in a number of ways:

- Only set the CCP of the required object to zero after the Scene is running.
- Place files for objects which have CCP set to zero in a different module to other assets.

SetData on a CCP=0 object may cause other commonly used objects in the same module to be flushed from the receiver. For example, this may delay Scene transitions. This can be addressed by:

- placing files for objects which have CCP set to zero in a different module to other assets.

NOTE: These hints reflect that some implementations have module rather than object based caching strategies.

It should also be noted here that this feature is provided to fulfil very specific application requirements (where other provided methods may not be suitable) and misuse of this functionality may seriously affect application/receiver performance.

#### 17.18.2.3 Object ordering

The suggested order of messages within a module is:

1) Directory messages on the path to urgently required files.

2) Files required urgently.

3) Files and directories required less urgently.

In principle, receivers can extract files from partially loaded modules. Therefore, placing these files and the directories that provide access to them early in the module provides a theoretical opportunity for receivers to deliver the files to the MHEG-5 engine more rapidly.

#### 17.18.2.4 Conflict between MHEG cache policy and MHP cache_priority_descriptors

In some implementations the Object Carousel client used by the MHEG engine may be written to understand cache_ priority_descriptors as specified for the MHP profile Object Carousel. These provide module caching hints to the carousel client.

It should be noted that, under certain circumstances, it is possible for cache_priority_descriptors to cause an MHEG application to operate incorrectly. Application authors should ensure that, if such descriptors are transmitted, they do not conflict with the caching policy required by the application.

The present document does not define the receiver's behaviour when there is such a conflict. It is expected that localized profiles of this Profile will define the appropriate behaviour.

### 17.18.3 Use of memory

#### 17.18.3.1 Forked resident programs

When a resident program is invoked with Call there is only one execution thread. So, the resident program can safely work directly on the variable storage of the MHEG-5 application.

When a resident program is invoked with Fork the resident program executes concurrently with the main MHEG-5 application. As described in clause 11.10.13 the Forked resident program acts on a snapshot (i.e. a copy) of its In and In-Out parameters.

Authors should note that there are memory budget benefits of invoking resident programs with Call rather than Fork. This may be significant where the parameters of the resident programs have large quantities of data, which may be the case with the string manipulation RPs.

#### 17.18.3.2 Original content never goes away

Note that the requirement to support cloning means that each Ingredient must hold a copy of its OriginalContent forever and that the OriginalContent is converted into a run-time form as soon as the object is prepared.

If an Ingredient is cloned it inherits the OriginalContent of the parent object and then manufactures a run-time version of this content. So, each clone has a redundant copy of the OriginalContent of its parent.

More efficient use of memory is made if the object to be cloned has minimal OriginalContent and then SetData is used to initialize each instance.

#### 17.18.3.3 Multiple references to the same content

If multiple Ingredients reference the same file normally only one copy of the content is held in receiver memory. However, if the file version changes between preparing objects then multiple versions of the content may be loaded into memory.

Authors should note that as files do not have individual file version information (the version information is on the module) there may be unexpected side-effects where a "static" file is in the same module as a "dynamic" file.

Separating dynamic and static files into different modules will prevent this problem.

### 17.18.4 Encoding of reserved fields

To ensure future compatibility all reserved fields shall be set to 0 unless otherwise specified.

## 17.19 GetEngineSupport feature strings

### 17.19.1 Engine profile

Certain legacy (non-compliant) receivers respond true when N=1 (character code 0x31) or N=2 (character code 0x32). Receivers that respond true to N=1 should not return true for any value of manufacturer-specific string.

### 17.19.2 Engine identification

The present document provides two mechanisms for applications to obtain information about the receiver it is running on. Normally, applications will use the UniversalEngineProfile(N) GetEngineSupport request (see clause 11.4.1) to test for a particular MHEG-5 engine version, or a particular receiver type and version. The WhoAmI resident program (see clause 11.10.12.1) provides a means for an application to find out the set of UniversalEngineProfile(N) feature strings that the receiver will respond to. This would typically be used to characterize a particular receiver during application development, to determine an appropriate GetEngineSupport request to use subsequently.

### 17.19.3 Packages

A receiver may report itself as conformant with this Profile even though it may not implement some packages specified. (See clause 1.1). Therefore an application that uses functionality offered by these packages should first check for their availability by using the appropriate GetEngineSupport string.

# Annex A (informative): The user experience

## A.1 Introduction

This clause looks at the behaviour seen by the user. It is provided to give context to the receiver specifications. However, much of the behaviour shown here results from functionality coded into the broadcast application, rather than the receiver.

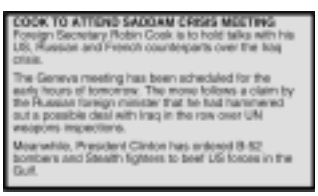
This broadcaster controlled behaviour should be seen as a concept model for how services may appear. As broadcasters develop services, and gain experience from user feed-back, the detail of the behaviour they implement is likely to evolve.

### A.1.1 Visual appearance

#### A.1.1.1 Balance of AV and MHEG-5

Table A.1 illustrates the range of different visual appearances the viewer might experience. Each "screen" shows a different balance between "conventional TV" AV content and information delivered via MHEG-5.

**Table A.1: Typical range of programme types perceived by viewers**

| Visual appearance | Description |
|---|---|
|  | 1) Conventional TV. |
|  | 2) TV with visual prompt of available information. |
|  | 3) TV with information overlaid. |
|  | 4) Information with video or picture inset. |
|  | 5) Just information. |

### A.1.1.2 Time and space

Figure A.1 illustrates how the user may see a change in appearance either when they change channel or as a service changes through time.

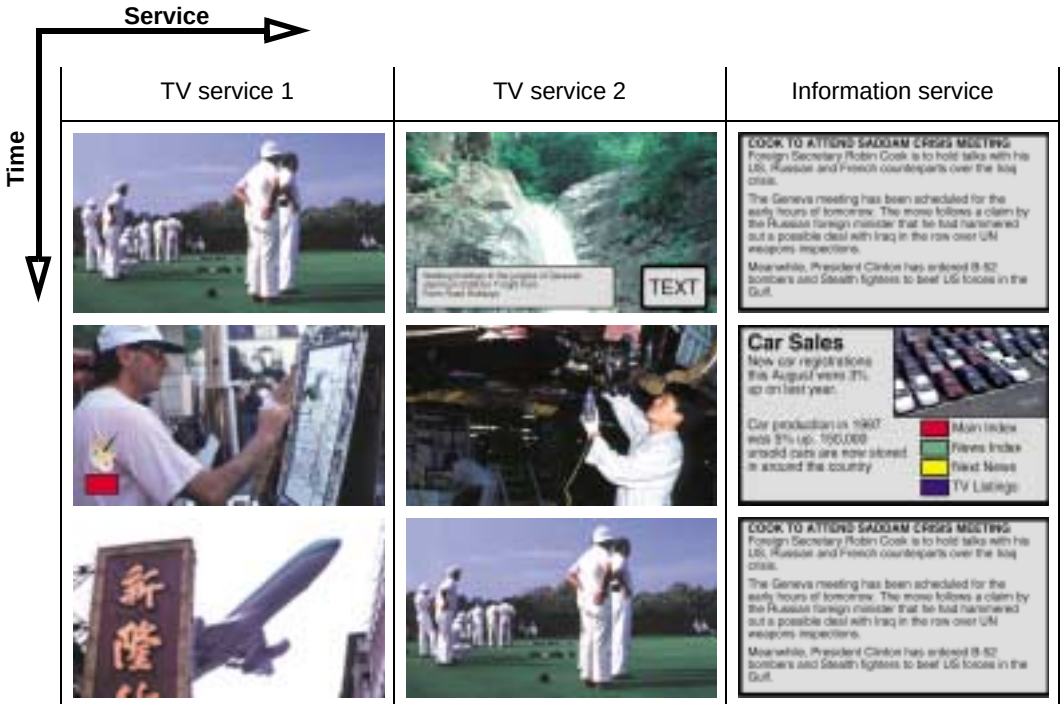



**Figure A.1: What might be seen across channels and through time**

## A.2 User navigation

### A.2.1 Channel change

The user can use any of the navigation methods to change channel:

- "surfing" using "Programme up" and "Programme down";
- direct selection by pressing a favourite channel button;
- selection from options within the "Info" or "Guide" screens.

These apply equally to TV and information services. There may also be cases where the user changes channel from within an information service.

When a service has been selected, regardless of the service type, the user can again change channel in the usual way.

See figure A.2.

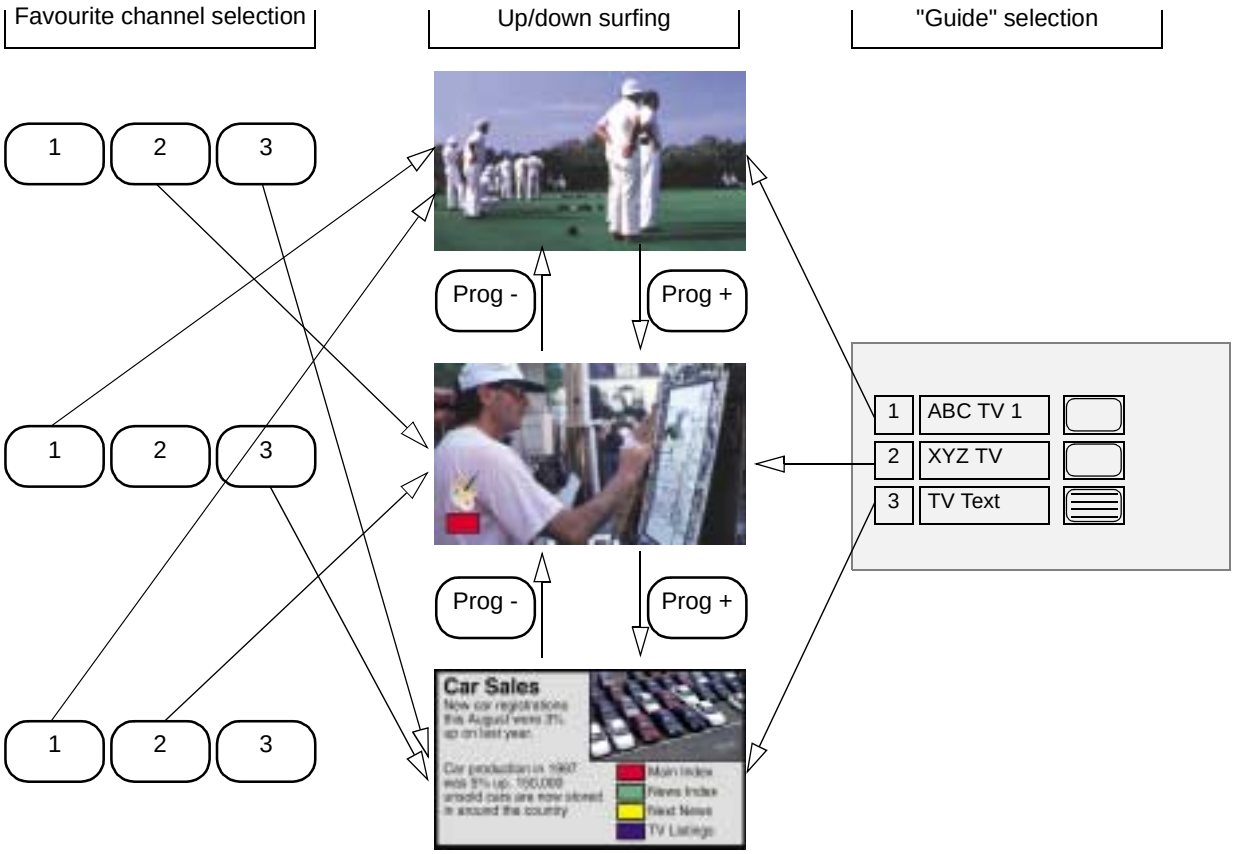



**Figure A.2: Changing channels**

### A.2.1.1 The "Text" button

Figure A.3 illustrates how the effect of the "Text" button may vary slightly from programme to programme.

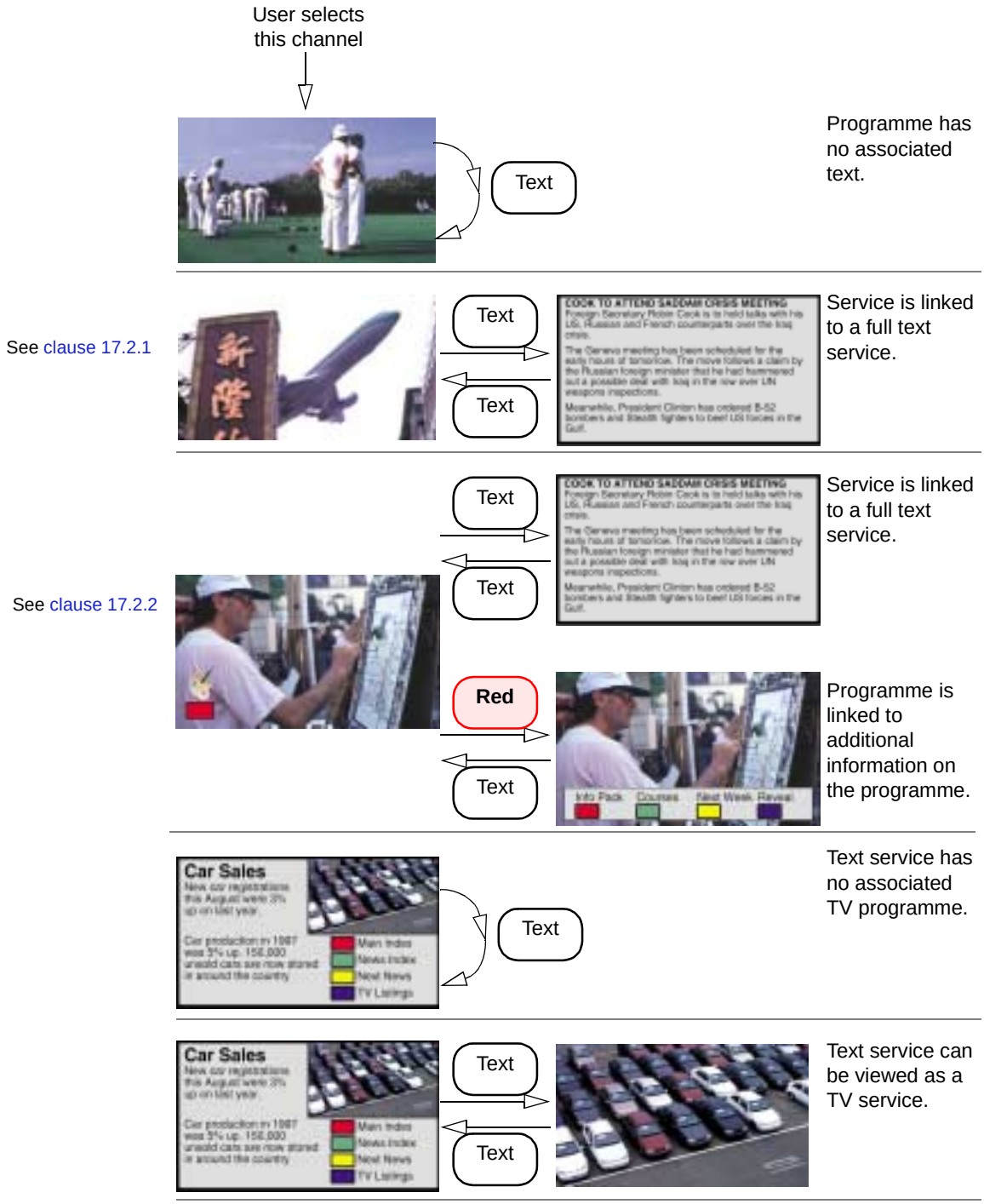



**Figure A.3: Using the text button**

## A.3 Channel selections within an information service

MHEG-5 services can provide facilities to change channel. For example, they may provide listings of current programmes and the ability to select a programme.

The effect of changing channel from an MHEG-5 application is exactly the same as if the user selected a channel from the receiver's "Guide".

See figure A.4.

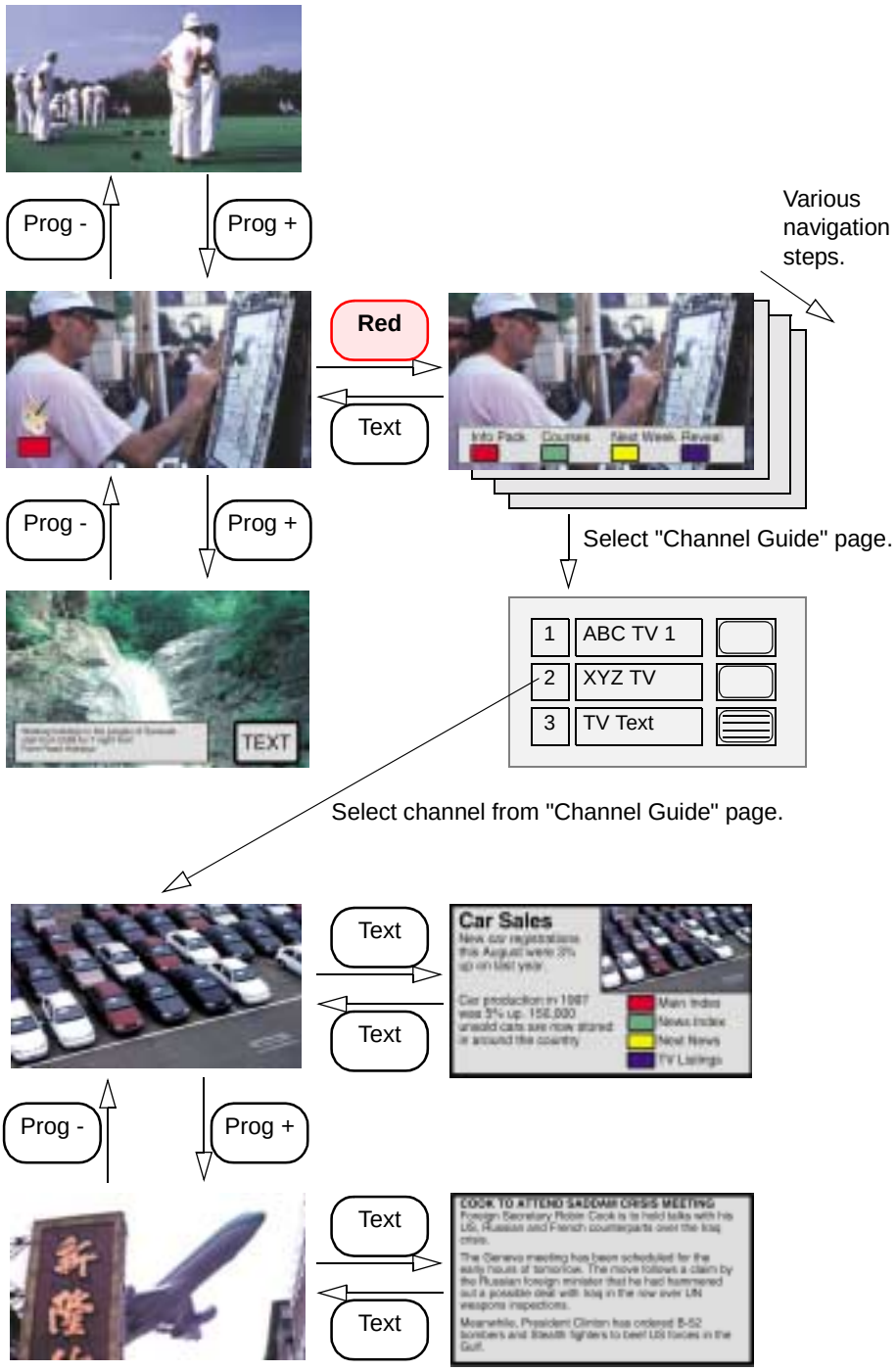



**Figure A.4: Using a TV listings page to change channel**

## A.4 Use of video within an information service

MHEG-5 services can use video inset into pages. For example, this could provide a video "preview" of the services on a multiplex. This might result in a channel change, but if the viewer "backs out" they will return to the service they started from.

See figure A.5.

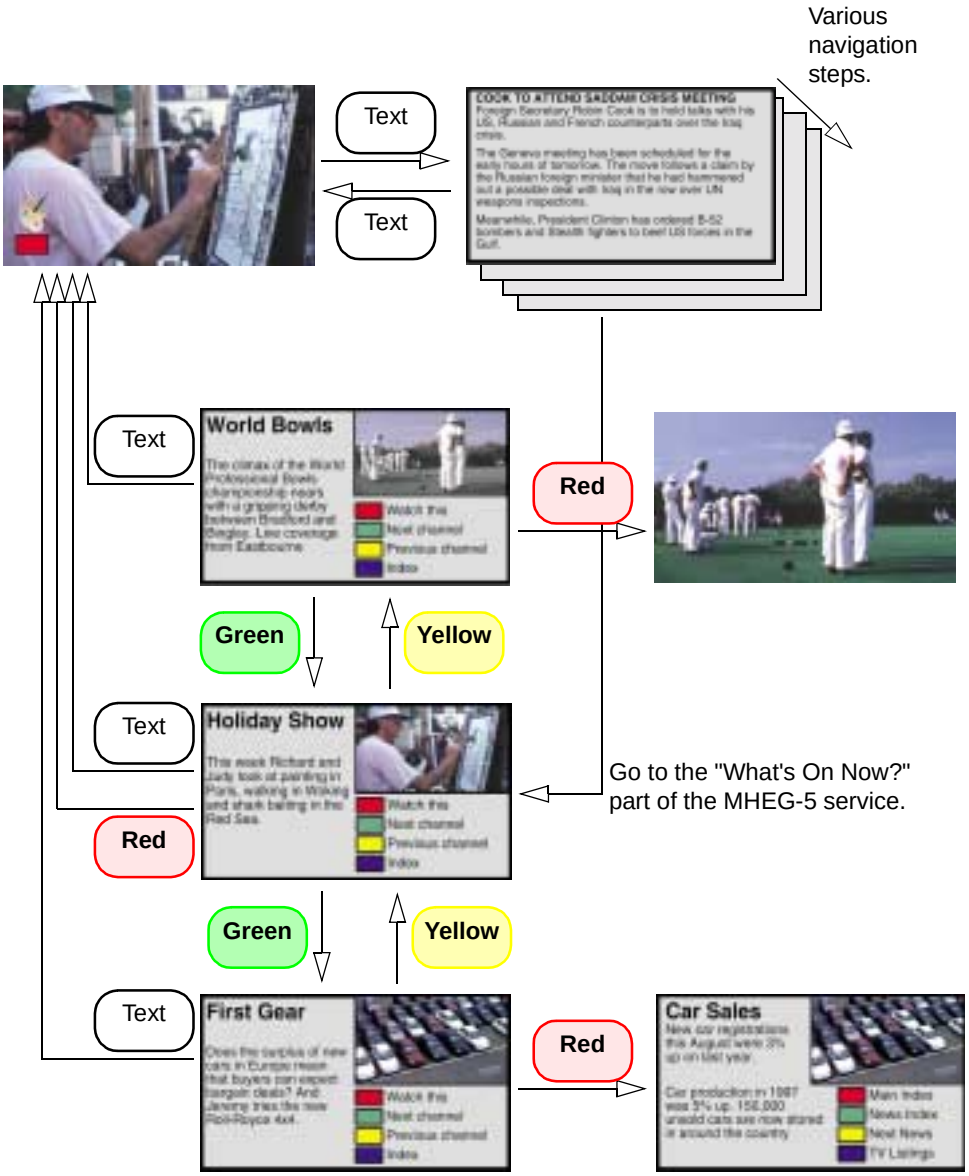



**Figure A.5: A possible "What's on now?" TV browser**

# History

| Document history | | |
|---|---|---|
| V1.1.1 | September 2004 | Membership Approval Procedure MV 20041105: 2004-09-07 to 2004-11-05 |
| | | |
| | | |
| | | |
| | | |