



MHEG-5 Broadcast Profile

EBU

OPERATING EUROVISION

Reference

RES/JTC-031

Keywords

broadcasting, data, digital, DVB, IP, 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

The present document can be downloaded from:
<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (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
<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:
<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2016.

© European Broadcasting Union 2016.

All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE™** are Trade Marks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights	20
Foreword.....	20
Modal verbs terminology.....	20
Introduction	20
1 Scope	21
1.0 Introduction	21
1.1 Localizing the present document.....	21
1.1.0 Approach to creating a local implementation	21
1.1.1 Extensions.....	21
1.1.2 Allocation of codes	22
1.1.3 Duplicate services	22
2 References	23
2.1 Normative references	23
2.2 Informative references.....	25
3 Definitions and abbreviations.....	26
3.1 Definitions.....	26
3.2 Abbreviations	28
4 Conventions.....	30
5 Basic architecture	30
6 Transport protocols.....	32
7 Content formats	32
7.1 Static formats.....	32
7.1.1 Bitmap image formats.....	32
7.1.1.1 PNG.....	32
7.1.1.2 MPEG-2 I-frames.....	32
7.1.2 Monomedia formats for audio clips	32
7.1.3 Monomedia formats for text	32
7.2 Broadcast streaming formats	32
7.3 Resident fonts	32
7.4 Colour representation	32
8 Application model	33
8.0 Introduction	33
8.1 Application lifecycle	33
8.1.1 Launching and terminating MHEG-5 applications	33
8.1.2 Preparing for launch.....	33
8.1.3 Auto boot broadcast application	34
8.1.3.0 When to launch an auto-boot application.....	34
8.1.3.1 How an auto-boot application is done.....	34
8.1.4 Auto kill application	34
8.1.5 Application context.....	34
8.1.5.0 Identification of a file reference source.....	34
8.1.5.1 Initial carousel.....	34
8.1.5.2 Current carousel	34
8.1.5.3 Current source	35
8.1.5.4 Authentication Source.....	35
8.1.6 Accessible file systems	35
8.1.6.0 List of available file systems	35
8.1.6.1 Broadcast applications	35
8.1.6.2 CI introduced applications	35
8.1.6.3 Interaction Channel	36
8.1.7 Keeping an application running across service tunes.....	36
8.1.7.0 Required behaviour for non-destructive tune	36

8.1.7.1	Broadcast file system requests	36
8.1.7.2	Timers	36
8.1.7.3	Carousel Identity	36
8.1.7.4	Broadcast file system	37
8.1.7.5	Network and service boot info	37
8.1.7.6	Behaviour of stream decoders	37
8.1.7.7	Content management.....	38
8.1.7.8	Receiver keys	38
8.1.7.9	Interaction with resident programs.....	38
8.1.7.10	Interaction channel security	38
8.1.7.11	Autoboot behaviour.....	38
8.1.7.12	True persistent storage	38
8.2	Application stacking.....	38
9	Signalling	39
9.1	Introduction to application lifecycle signalling	39
9.1.0	Overview	39
9.1.1	Application-level signalling.....	39
9.1.2	Service-level signalling.....	39
9.1.3	Network-level signalling.....	40
9.1.4	Scope	40
9.2	AIT Signalling Extension	40
9.2.0	AIT profile	40
9.2.1	Coexistence of Classical and AIT signalling	41
9.2.2	Life-cycle signalling in AIT and PMT.....	42
9.2.2.0	Life-cycle information in the PMT	42
9.2.2.1	carousel_id_descriptor	42
9.2.2.2	data_broadcast_id_descriptor.....	42
9.2.2.3	MHEG Non-destructive tune support with AIT signalling	43
9.2.2.3.0	AIT signalling.....	43
9.2.2.3.1	Network and service boot info.....	43
9.2.2.4	Definition of "well formed" for MHEG Applications	43
9.3	PMT and ServiceGateway signalling extension	43
9.3.1	Introduction.....	43
9.3.2	Identification of auto-boot application.....	44
9.3.2.0	Identifying the boot-PID	44
9.3.2.1	data_broadcast_id_descriptor	44
9.3.2.2	Network boot info sub-descriptor	45
9.3.2.3	Service boot info sub-descriptor	45
9.3.3	Acquisition of the ServiceGateway object.....	46
9.3.3.0	Approach to acquisition of the ServiceGateway object.....	46
9.3.3.1	carousel_id_descriptor	46
9.3.4	Acquisition of the auto-boot object.....	46
9.3.4.0	Approach to acquisition of the auto-boot object	46
9.3.4.1	ServiceContextList.....	46
9.3.4.2	Locating the initial object.....	48
9.3.4.2.1	Explicit Initial Object Identified.....	48
9.3.4.2.2	No Explicit Initial Object Identified	48
9.3.4.2.3	Initial File System.....	48
9.3.4.2.4	Example	48
9.3.5	Example of steps required for auto-boot.....	49
9.3.6	Service-level application lifecycle signalling	49
9.3.7	Network-level application lifecycle signalling	50
9.3.7.0	PMT monitoring.....	50
9.3.7.1	Auto mount broadcast file system.....	50
9.3.7.2	network_boot_info	50
9.3.7.3	data_broadcast_id.....	50
9.3.7.4	carousel_id	50
9.3.7.5	Carousels moving components.....	50
9.3.7.6	Removal of service.....	51
10	Security.....	51

11	MHEG-5 engine profile.....	51
11.0	Introduction	51
11.1	Basic specification.....	51
11.2	Object interchange format	51
11.3	Set of classes	52
11.4	Set of features.....	53
11.4.0	MHEG-5 optional features.....	53
11.4.1	GetEngineSupport "feature" strings.....	53
11.4.1.0	Set of mandatory GetEngineSupport "feature" strings.....	53
11.4.1.1	VideoDecodeOffset.....	56
11.4.1.2	BitmapDecodeOffset.....	56
11.4.1.3	Engine identification strings.....	56
11.4.1.4	Audio stream decoders.....	57
11.5	Content data encoding.....	57
11.5.0	Coding attributes and hook values.....	57
11.5.1	Use of negative hook values	58
11.5.2	Bitmap objects	58
11.5.2.1	Scaling.....	58
11.5.2.2	Tiling.....	58
11.5.2.3	Transparency.....	58
11.5.3	Stream "memory" formats	58
11.5.3.0	Scope of StreamComponent applicability.....	58
11.5.3.1	Audio.....	58
11.5.4	Non-linear stream formats	59
11.5.4.0	Profile for IP-delivered Transport Stream.....	59
11.5.4.1	Video.....	59
11.5.4.2	Audio.....	59
11.5.4.3	Subtitles.....	59
11.5.4.4	Encrypted non-linear streams.....	59
11.6	User input	60
11.6.1	Base remote control functions	60
11.6.1.0	Overview of remote control function groups	60
11.6.1.1	Receiver group	60
11.6.1.2	Register 3 group (see table 11.8).....	61
11.6.1.3	Register 4 group (see table 11.8).....	61
11.6.1.4	Register 5 group (see table 11.8).....	61
11.6.1.5	Register 6 group (see table 11.8).....	61
11.6.2	Extended remote control functions	61
11.6.2.0	Outline of extended function groups.....	61
11.6.2.1	Register 6 group (see table 11.9).....	61
11.6.3	UserInput registers.....	61
11.6.3.1	Base UserInput registers	61
11.6.3.2	Extended UserInput registers	62
11.6.4	Implementation of this interaction model	63
11.6.5	Interaction with broadcast-triggered native applications	63
11.7	Semantic constraints on MHEG-5 applications.....	63
11.8	EngineEvents.....	63
11.8.0	List of required engine events.....	63
11.8.1	Object retrieval errors	65
11.8.2	Object retrieval errors - Interaction Channel	65
11.9	Protocol mapping and external interaction	65
11.10	ResidentPrograms.....	66
11.10.0	List of ResidentPrograms.....	66
11.10.1	Typical use.....	67
11.10.2	Program names	67
11.10.3	Encoding of resident program names.....	68
11.10.4	Date and time functions	68
11.10.4.1	Day, date and time functions.....	68
11.10.4.2	GetCurrentDate	68
11.10.4.3	FormatDate.....	68
11.10.4.4	GetDayOfWeek.....	69
11.10.5	Random number function	69

11.10.5.1	Random	69
11.10.6	Type conversion functions	69
11.10.6.1	CastToContentRef	69
11.10.6.2	CastToObjectRef	70
11.10.6.3	CastToStringInt	70
11.10.7	String manipulation functions	70
11.10.7.0	Use of string manipulation functions	70
11.10.7.1	Range of string index values	70
11.10.7.2	GetStringLength	71
11.10.7.3	GetSubString	71
11.10.7.4	SearchSubString	71
11.10.7.5	SearchAndExtractSubString	72
11.10.8	Service selection	72
11.10.8.0	Use of service selection functions	72
11.10.8.1	SI_GetServiceIndex	72
11.10.8.2	SI_TuneIndex	73
11.10.8.3	SI_GetBasicSI	73
11.10.8.4	SI_TuneIndexInfo	74
11.10.8.4.0	Usage	74
11.10.8.4.1	Destructive service tune	75
11.10.8.4.2	Non-destructive service tune	76
11.10.9	Checking references	77
11.10.9.0	Use of reference checking functions	77
11.10.9.1	CheckContentRef	77
11.10.9.2	CheckGroupIDRef	78
11.10.10	Presentation information	78
11.10.10.1	VideoToGraphics	78
11.10.10.2	SetWidescreenAlignment	79
11.10.10.3	GetDisplayAspectRatio	80
11.10.10.4	SetSubtitleMode	80
11.10.10.5	SetBroadcasterInterruptions	80
11.10.10.6	GetAudioDescPref	81
11.10.10.7	GetSubtitlePref	82
11.10.11	Common Interface	82
11.10.11.1	CI_SendMessage	82
11.10.12	Interaction channel	82
11.10.12.0	Introduction to interaction channel resident programs	82
11.10.12.1	GetICStatus	82
11.10.12.2	ReturnData	83
11.10.12.3	MeasureStreamPerformance	84
11.10.12.4	PromptForGuidance	84
11.10.12.5	PersistentStorageInfo	85
11.10.12.6	SetCookie	85
11.10.12.7	GetCookie	86
11.10.12.8	GetPINSupport	86
11.10.13	Hybrid file system	86
11.10.13.0	Introduction to hybrid file system resident programs	86
11.10.13.1	SetHybridFileSystem	86
11.10.14	Developer utilities	88
11.10.14.1	WhoAmI	88
11.10.14.2	Debug	88
11.10.15	Access to application lifecycle signalling	89
11.10.15.1	GetBootInfo	89
11.10.16	Data exchange with ResidentPrograms	89
11.10.16.0	Scope of behaviour	89
11.10.16.1	Memory spaces	89
11.10.16.2	On invocation	89
11.10.16.3	CallSucceeded/ForkSucceeded Values	89
11.10.16.4	During execution	89
11.10.16.5	On completion	90
11.10.17	Duration of effect of ResidentPrograms	90
11.11	Limitations on standard data-types	90

11.11.1	BooleanVariable	90
11.11.2	IntegerVariable	90
11.11.3	OctetString	90
11.11.4	ObjectNumber.....	90
11.11.5	GroupIdentifier and ContentReference	91
11.12	Extensions to the MHEG-5 language specification.....	91
11.12.1	Preamble	91
11.12.2	Changes to the Group class.....	91
11.12.2.0	Overview of changes.....	91
11.12.2.1	Changes to "Own internal attributes"	91
11.12.2.2	Changes to "Events"	91
11.12.2.3	Changes to "Effect of MHEG-5 actions"	91
11.12.3	Changes to the Application class	91
11.12.3.1	Changes to "Own exchanged attributes"	91
11.12.3.2	Changes to "Own internal attributes"	92
11.12.3.3	Changes to "Effect of MHEG-5 actions"	92
11.12.4	Changes to the Scene class	93
11.12.4.0	Overview of changes.....	93
11.12.4.1	Changes to "Own exchanged attributes"	93
11.12.4.2	Changes to "Own internal attributes"	93
11.12.4.3	Changes to "Events"	93
11.12.4.4	Changes to "Effect of MHEG-5 actions"	93
11.12.5	Changes to the TokenGroup class.....	94
11.12.5.1	Changes to "Effect of MHEG-5 actions"	94
11.12.6	Changes to the ListGroup class.....	94
11.12.6.1	Changes to "Own exchanged attributes"	94
11.12.6.2	Changes to "Own internal attributes"	94
11.12.6.3	Changes to "Effect of MHEG-5 actions"	95
11.12.7	Changes to the Bitmap class	95
11.12.7.1	Changes to "Own internal attributes"	95
11.12.7.2	Changes to "Effect of MHEG-5 actions"	95
11.12.8	Changes to the Text class.....	97
11.12.8.1	Changes to "Own exchanged attributes"	97
11.12.8.2	Changes to "Own internal attributes"	97
11.12.8.3	Changes to "Effect of MHEG-5 actions"	98
11.12.9	Changes to the Stream class.....	99
11.12.9.0	Overview of changes.....	99
11.12.9.1	Changes to "Own exchanged attributes"	99
11.12.9.2	Changes to "Own internal attributes"	100
11.12.9.3	Changes to "Internal behaviours"	100
11.12.9.4	Changes to "Effect of MHEG-5 actions"	100
11.12.10	Changes to the Video class	101
11.12.10.1	Changes to "Own internal attributes"	101
11.12.10.2	Changes to "Effect of MHEG-5 actions"	101
11.12.11	Changes to the Slider class	105
11.12.11.1	Changes to "Own exchanged attributes"	105
11.12.11.2	Changes to "Own internal attributes"	105
11.12.11.3	Changes to "Events"	105
11.12.11.4	Changes to "Internal behaviour"	106
11.12.11.5	Changes to "Effect of MHEG-5 actions"	106
11.12.12	Changes to the HyperText class.....	107
11.12.12.1	Changes to "Own internal attributes"	107
11.12.12.2	Changes to "Events"	108
11.12.12.3	Changes to "Internal behaviours"	108
11.12.12.4	Changes to "Effect of MHEG-5 actions"	109
11.12.13	Changes to the LineArt class	109
11.12.13.0	Overview of changes.....	109
11.12.13.1	Changes to "Own exchanged attributes"	110
11.12.13.2	Changes to "Effect of MHEG-5 actions"	110
11.13	Clarifications, restrictions and amendments.....	110
11.13.1	Additional semantics for the SetTimer action.....	110
11.13.2	CounterPosition attribute	110

11.13.2.0	Relationship of CounterPosition to NPT	110
11.13.2.1	Broadcast delivered streams	110
11.13.2.2	IP delivered streams	110
11.13.3	Synchronous event processing	111
11.13.3.0	Interpretations of synchronous event processing behaviour	111
11.13.3.1	Preferred interpretation	111
11.13.3.2	Alternative interpretation	111
11.13.3.3	Explanation	112
11.13.4	Actions that generate more than one synchronous event	112
11.13.5	TransitionTo deactivation of shared=FALSE ingredients	112
11.13.6	Interactibles	112
11.13.7	Clarification of StreamPlaying and StreamStopped events	113
11.13.8	Use of NextScenes to preload content	113
11.13.9	Application defaults	113
11.13.10	Effect of SetData on Internal Attributes	114
11.13.11	Clarification of TransitionTo, Launch and Spawn behaviour	114
11.13.12	References to shared=FALSE ingredients	114
11.13.13	Restrictions on Link EventSource	114
11.13.14	Video Termination attribute	114
11.13.15	Clarification of Root object destruction behaviour	114
11.13.16	Illegal parameter handling in :SetVariable	114
11.13.17	Referencing Group objects with an InternalReference	115
11.14	Service Information extension	115
11.14.0	Introduction	115
11.14.1	Service Information resident programs	115
11.14.1.0	List of Service Information resident programs	115
11.14.1.1	SI_GetServiceInfo	116
11.14.1.2	SI_GetEventInfo	116
11.15	PVR extensions	118
11.15.0	Introduction	118
11.15.1	PVR Implementation	118
11.15.2	CRID format	118
11.15.3	PVR extension resident programs	118
11.15.3.0	List of PVR extension resident programs	118
11.15.3.1	PVR_MakeBooking	119
11.15.3.2	PVR_CancelBooking	119
11.15.3.3	PVR_ListBooking	120
11.16	InputMaskExtension	120
11.16.0	Introduction	120
11.16.1	Operation	120
11.16.1.0	Outline of operation	120
11.16.1.1	Engine Events	121
11.16.2	Changes to the MHEG specification	121
11.16.2.1	Input Register semantics	121
11.16.2.2	New attributes and ElementaryActions	121
11.16.2.3	TestInputMask New Resident Program	122
11.16.2.4	Key values table	122
11.17	File System Acceleration Extension	123
11.17.0	Introduction	123
11.17.1	File Groups	124
11.17.2	Scope of File Groups	124
11.17.3	Stored groups descriptor	124
11.17.4	Group Location Descriptor	126
11.17.5	Group Manifest	126
11.17.6	Group Manifest management	128
11.17.7	Stored File Access	128
11.17.7.1	Read access	128
11.17.7.2	Write access	128
11.17.8	Stored Group Deletion	129
11.18	Application Launch Extension	129
11.18.1	GetEngineSupport 'feature' strings	129
11.18.2	Resident Programs	129

11.18.2.0	List of resident programs	129
11.18.2.1	ApplicationLaunch	129
11.18.2.2	GetLaunchArguments	130
11.18.3	Referencing MHEG Applications from other Presentation Technologies	131
12	MHEG-5 engine graphics model.....	131
12.1	The graphics plane.....	131
12.2	The colour palette.....	132
12.2.0	Minimum colour support	132
12.2.1	Reservation for MHEG-5 applications	132
12.2.1.0	Reserved locations	132
12.2.1.1	Fidelity of reproduction.....	132
12.2.1.2	Palette definition	132
12.2.2	Reservation for DVB subtitles	133
12.2.3	Subtitle priority for transparency	133
12.2.4	Reservation for manufacturer use	133
12.3	Colour representation	133
12.3.1	Colour space	133
12.3.2	Gamma.....	135
12.3.3	Direct/absolute colours	135
12.3.4	Approximation of transparency	135
12.3.5	PNG modes	135
12.4	Overlapping visibles.....	135
12.4.1	Transparency and overlapping visibles.....	135
12.4.1.1	Overlaying visibles.....	135
12.4.1.2	Rendering performance	136
12.5	LineArt and DynamicLineArt	136
12.5.1	Clarifications.....	136
12.5.1.1	Lineart borders	136
12.5.1.2	"Fat" lines.....	136
12.5.1.2.1	"Fat" lines are centred.....	136
12.5.1.2.2	Clipping at box edge.....	136
12.5.1.3	Line ends	137
12.5.1.4	Bordered bounding box.....	137
12.5.1.5	DrawSector.....	137
12.5.1.6	Effect of pixel transparency	137
12.5.1.7	Co-ordinate system.....	137
12.5.2	Limitations	138
12.6	Text, EntryFields and HyperText	138
12.7	PNG bitmaps	138
12.7.1	Specification conformance.....	138
12.7.2	Colour encoding.....	139
12.7.3	Aspect ratio signalling	139
12.8	MPEG-2 stills	140
12.8.1	File format	140
12.8.2	Semantics.....	140
12.8.3	Presentation.....	140
12.9	MPEG video.....	140
12.10	Appearance of Visible objects during content retrieval.....	140
12.11	High definition graphics model	140
12.11.0	Requirements for support.....	140
12.11.1	Resolution.....	140
12.11.1.0	Resolution support requirements.....	140
12.11.1.1	HD resolution graphics plane	141
12.11.1.1.0	Scope	141
12.11.1.1.1	Resolution.....	141
12.11.1.1.2	Colour range	141
12.11.1.1.3	Direct/absolute colours	141
12.11.1.1.4	Text rendering	141
12.11.1.1.5	Bitmap format and resolution	141
12.11.2	Mapping the MHEG application co-ordinate system to the graphics plane.....	141
12.11.3	Intelligent rendering.....	142

12.11.3.0	Scope.....	142
12.11.3.1	Introduction.....	142
12.11.3.1.0	Overview of process.....	142
12.11.3.1.1	Co-ordinate transformation.....	142
12.11.3.2	Bounding box transformation.....	142
12.11.3.3	Visual appearance.....	143
12.11.3.3.1	Text.....	143
12.11.3.3.2	Images.....	144
12.11.3.3.3	Line Art.....	144
12.11.3.3.4	DynamicLineArt.....	144
12.12	JPEG bitmaps.....	145
12.13	H.264/AVC stills.....	145
12.13.1	File format.....	145
12.13.2	Semantics.....	145
12.13.3	Presentation.....	146
13	Text and interactibles.....	146
13.1	Text rendering overview.....	146
13.1.0	Application of text rendering rules.....	146
13.1.1	Non-presented text.....	146
13.2	Character encoding.....	146
13.2.0	Character encoding format.....	146
13.2.1	UTF-8.....	146
13.2.2	Null characters.....	147
13.2.3	CharacterSet attribute.....	147
13.3	Fonts.....	147
13.3.1	Downloading.....	147
13.3.1.0	Application of downloadable fonts.....	147
13.3.1.1	OpenType fonts.....	147
13.3.1.1.0	Application of OpenType fonts.....	147
13.3.1.1.1	Profile of OpenType.....	147
13.3.1.1.2	Font parameters.....	148
13.3.1.1.3	Text Styles.....	148
13.3.1.2	Presentation.....	148
13.3.1.3	Defensive response.....	148
13.3.1.4	Font resource model.....	148
13.3.2	Embedded font.....	148
13.3.2.1	The DTG/RNIB font characteristics (informative).....	148
13.3.2.2	Font version.....	149
13.3.2.3	Required sizes and styles.....	149
13.3.3	Invoking the font.....	149
13.4	Text object attributes.....	150
13.4.1	FontAttributes.....	150
13.4.1.0	FontAttribute formats.....	150
13.4.1.1	Textual form.....	150
13.4.1.2	Short form.....	150
13.4.2	Control of text flow.....	151
13.4.2.1	Required flow modes.....	151
13.5	Text rendering.....	151
13.5.1	Philosophy.....	151
13.5.2	Font definition.....	152
13.5.2.0	Introduction.....	152
13.5.2.1	Font bounds.....	152
13.5.2.2	"Physical" font data.....	153
13.5.3	Converting font metrics to display pixels.....	153
13.5.3.0	Approach.....	153
13.5.3.1	Vertical resolution.....	153
13.5.3.2	Horizontal resolution.....	153
13.5.4	Rendering within the limits of the Text object.....	154
13.5.4.0	Approach.....	154
13.5.4.1	Vertical limits.....	155
13.5.4.2	Horizontal limits.....	155

13.5.5	"logical" text width rules	155
13.5.5.0	Approach	155
13.5.5.1	Computing "logical" text width	156
13.5.5.1.0	Parameters used	156
13.5.5.1.1	Font sizes	156
13.5.5.1.2	Character widths	156
13.5.5.1.3	Kerning	156
13.5.5.1.4	Letter spacing	156
13.5.5.2	Logical text width	157
13.5.6	Line breaking	157
13.5.6.1	TextWrapping false	157
13.5.6.2	TextWrapping true	157
13.5.7	Positioning lines of text vertically within the Text object	158
13.5.7.1	Truncation	158
13.5.7.2	Positioning	158
13.5.7.3	Void	160
13.5.8	Rendering lines of text horizontally	160
13.5.8.1	Truncation	160
13.5.8.2	Placement	160
13.5.8.3	Examples	160
13.5.8.4	Scaling for HD resolution graphics planes	160
13.5.9	Tabulation	160
13.5.10	Placing runs of characters and words	161
13.6	Text mark-up	162
13.6.1	White space characters	162
13.6.2	Marker characters	162
13.6.3	Non-printing characters	162
13.6.4	Format control mark-up	163
13.6.5	Future compatibility	163
13.7	EntryFields	163
13.7.1	Supported characters	163
13.7.2	Appearance	164
13.7.2.1	Receivers that do not implement InteractionChannelExtension	164
13.7.2.2	Receivers that implement InteractionChannelExtension	165
13.7.3	Behaviour	165
13.7.3.1	Character encoding	165
13.7.3.2	Semantics of EntryFieldFull and MaxLength	165
13.7.3.3	EntryPoint	165
13.7.3.4	Successive character entry	165
13.7.3.5	Only SetData when inactive	166
13.7.3.6	User input	166
13.7.3.7	Numerics of the EntryField	166
13.7.4	Non-numeric input	166
13.7.4.0	Scope	166
13.7.4.1	Introduction	167
13.7.4.2	Minimum requirements	167
13.7.4.3	SMS entry method	167
13.7.4.3.0	Scope	167
13.7.4.3.1	Basic Method	167
13.7.4.3.2	Timeout Period	167
13.7.4.3.3	Appearance during input	167
13.7.4.3.4	Character to key mappings	167
13.7.4.3.5	Character subsets	168
13.8	HyperText	168
13.8.0	HyperText markup format	168
13.8.1	HyperText anchors	169
13.8.2	Appearance	170
13.8.2.1	Visual appearance of anchors	170
13.8.2.2	Default anchor colours	170
13.8.2.3	Highlight	170
13.8.3	Behaviour	170
13.8.3.0	Basic behaviour	170

13.8.3.1	Anchor identification	170
13.8.3.2	Behaviour	170
13.8.3.3	Special behaviour at boundaries	170
13.9	Slider	171
13.9.1	Appearance	171
13.9.2	Behaviour	172
13.10	Text rendering example (informative)	172
14	MHEG receiver requirements	174
14.1	Introduction	174
14.2	Management of stream decoders	174
14.2.1	Application killed by receiver	174
14.2.1.0	Default behaviour	174
14.2.1.1	On change of service	174
14.2.2	Effect of lockscreen	174
14.2.3	Stream inheritance on Application object activation	174
14.2.4	Synchronizing stream decoder state	175
14.2.4.0	Point of synchronization	175
14.2.4.1	Graphics Plane	175
14.2.4.2	Stream component selection	175
14.2.5	Stream continuance on Application object deactivation	175
14.2.6	Locating components carried in Transport Streams	176
14.2.6.0	Identifying a component	176
14.2.6.1	Multiplex references	176
14.2.6.1.1	DSM-CC Stream object	176
14.2.6.1.2	URL explicit format	176
14.2.6.1.3	URL inheritance formats	176
14.2.6.1.4	Interaction Channel format	177
14.2.6.1.5	Services in other transport streams	177
14.2.6.1.6	StreamEvent events	177
14.2.6.1.7	CounterTrigger events	177
14.2.6.1.8	Content management	177
14.2.6.2	Component references	177
14.2.7	Locating components carried in an Elementary Stream	177
14.2.8	Stream presentation errors	178
14.2.9	IC Stream buffering	178
14.2.9.0	Introduction	178
14.2.9.1	Buffer reference model	178
14.2.9.2	Application control of the IC stream buffer	178
14.2.9.3	Restrictions	179
14.3	Application interaction with user control of linear content decoders	180
14.3.0	Introduction and undefined behaviour	180
14.3.1	Video decoder	180
14.3.2	Audio decoder	180
14.3.3	Subtitle decoder	181
14.3.4	Selection of subtitle and audio description components	181
14.4	Application impact on stream decoder specification	182
14.4.1	DVB subtitles	182
14.4.1.1	Flexibility of control	182
14.4.1.1.0	Scope	182
14.4.1.1.1	Subtitles are a facet of full-screen video	182
14.4.1.1.2	Subtitles have priority if enabled	182
14.4.2	Video decoder performance	182
14.4.3	Trick modes	182
14.4.3.0	Standard behaviour	182
14.4.3.1	Pause behaviour	182
14.4.4	MPEG presentation	183
14.4.4.0	Introduction	183
14.4.4.1	MPEG scaling reference model	183
14.4.4.2	Transparency of MPEG encoding	183
14.4.4.3	Quarter-screen MPEG	184
14.4.4.4	BoxSize for MPEG images	184

14.4.4.5	Video/I-frame object placement.....	184
14.4.4.5.0	Behaviour	184
14.4.4.5.1	Restricted capability	185
14.4.5	Content management of IC streams.....	185
14.4.6	Multiple stream objects.....	185
14.4.6.0	Behaviour.....	185
14.4.6.1	Simultaneous Demultiplexing.....	185
14.5	Application control of aspect ratio	185
14.5.0	Introduction.....	185
14.5.1	No active video object	186
14.5.2	I-frames.....	186
14.5.3	Quarter-screen video.....	186
14.5.4	Video greater than quarter-screen.....	187
14.5.5	Decision trees.....	187
14.6	Persistent storage.....	188
14.7	True persistent storage.....	189
14.7.0	Behaviour.....	189
14.7.1	Management of true persistent storage	190
14.8	Receiver resource model	190
14.8.1	Memory	190
14.8.2	Numbers of objects	191
14.8.2.0	Minimum number of active objects using stream decoders	191
14.8.2.1	Single PCR.....	191
14.8.3	Link recursion behaviour.....	191
14.8.4	Timer count and granularity.....	191
14.8.5	Timer duration	191
14.8.6	HD graphics bitmap requirements	191
14.9	Receiver process priority.....	192
14.9.1	OSD arbitration.....	192
14.9.2	Event handling whilst de-prioritized.....	193
14.9.2.1	Transparently	193
14.9.2.2	Non-transparently.....	193
14.10	Interaction with DVB Common Interface module system	193
14.10.1	Overview	193
14.10.2	Introduction of CI sourced file system.....	193
14.10.3	Guidelines for using Application MMI resource	193
14.10.3.0	Introduction.....	193
14.10.3.1	Resource contention.....	193
14.10.3.2	RequestStart	194
14.10.3.2.0	Behaviour	194
14.10.3.2.1	Application Domain Identifier.....	194
14.10.3.2.2	Initial object.....	194
14.10.3.3	RequestStartAck.....	195
14.10.3.4	FileRequest.....	195
14.10.3.5	FileAcknowledge	195
14.10.3.6	AppAbortRequest.....	196
14.10.3.7	AppAbortAck.....	196
14.10.3.8	Asynchronous events	196
14.10.4	Application Info Resource "Enter_Menu"	196
15	File system profile	197
15.1	Introduction	197
15.1.1	Broadcast file system.....	197
15.1.2	Interaction channel.....	197
15.2	Object carousel profile	198
15.2.1	DSM-CC sections	198
15.2.1.0	DSM-CC section format	198
15.2.1.1	Sections per TS packet.....	198
15.2.2	Data carousel	198
15.2.2.1	General	198
15.2.2.2	DownloadInfoIndication	198
15.2.2.3	DownloadServerInitiate	199

15.2.2.4	DownloadDataBlock	199
15.2.2.5	ModuleInfo	199
15.2.2.6	ServiceGatewayInfo	200
15.2.2.7	Download Cancel	201
15.2.3	The object carousel	201
15.2.3.1	BIOP Generic Object Message	201
15.2.3.2	CORBA strings	202
15.2.3.3	BIOP FileMessage	202
15.2.3.4	BIOP DirectoryMessage	203
15.2.3.5	BIOP ServiceGateway message	205
15.2.4	Streams and stream events	205
15.2.4.0	Selection of message type	205
15.2.4.1	BIOP StreamMessage	205
15.2.4.2	BIOP StreamEventMessage	207
15.2.4.2.0	Restrictions and syntax	207
15.2.4.2.1	Stream event names and event ids	208
15.2.4.2.2	Generating MHEG-5 StreamEvents	209
15.2.4.2.3	Tap longevity	209
15.2.4.2.4	Stream event subscription longevity	209
15.2.4.3	Identifying services using StreamMessages and StreamEventMessages	209
15.2.4.3.1	BIOP_PROGRAM_USE tap	209
15.2.4.4	DSM-CC sections carrying stream descriptors	209
15.2.4.4.1	"do it now" events	209
15.2.4.4.2	Section number	209
15.2.4.4.3	Current_next_indicator	209
15.2.4.4.4	Stream event life time	209
15.2.4.4.5	Encoding of table id extension	209
15.2.4.4.6	Resources to monitor stream events "do it now" events	210
15.2.4.5	Stream descriptors	210
15.2.4.5.1	Stream event descriptor	210
15.2.4.5.2	NPT Reference descriptor	210
15.2.4.5.3	NPT Endpoint descriptor	210
15.2.4.5.4	Stream Mode descriptor	210
15.2.4.6	Mapping stream descriptors into the MHEG-5 domain	211
15.2.5	BIOP Interoperable Object References	211
15.2.5.0	Restrictions and syntax	211
15.2.5.1	BIOPProfileBody	212
15.2.5.2	LiteOptionsProfileBody	213
15.2.6	Assignment and use of transactionId values	215
15.2.6.1	Background (informative)	215
15.2.6.2	Use in the present document	215
15.2.7	Mapping of objects to modules	216
15.2.8	Compression of modules	216
15.3	AssociationTag mapping	217
15.3.1	Association tags in "taps"	217
15.3.2	Different uses of "taps"	217
15.3.3	Using an AssociationTag to reference a service	217
15.3.3.1	BIOP_PROGRAM_USE tap	217
15.3.3.2	deferred_association_tags_descriptor	217
15.3.3.2.1	Resolving a service	217
15.3.3.2.2	Default behaviour	217
15.3.3.2.3	Transport_stream_id field	217
15.3.3.3	Service association tag mapping decision tree	218
15.3.4	Using an association tag to reference an elementary stream	218
15.3.4.1	MHEG-5 ComponentTags to DSM-CC association tags	218
15.3.4.1.0	Selecting an elementary stream from a ComponentTag	218
15.3.4.1.1	Tag vales for default components	218
15.3.4.1.2	Explicit component references	219
15.3.4.1.3	Mapping Errors	219
15.3.4.2	Mapping DSM-CC association_tags to DVB component_tags	219
15.3.4.2.0	Selecting a component_tag from an association_tag	219
15.3.4.2.1	stream_identifier_descriptor	219

15.3.4.2.2	association_tag descriptors	219
15.3.4.2.3	Elementary stream matching using the deferred_association_tags_descriptor	220
15.3.4.2.4	PMT changes	220
15.3.4.3	Elementary stream mapping pseudo code and decision tree	221
15.4	Caching	222
15.4.1	Transparent cache model	222
15.4.2	Determining file version	222
15.4.2.0	Use of moduleVersion for determining file version	222
15.4.2.1	Module acquisition	222
15.4.3	Content cache priority	223
15.4.4	Group cache priority	223
15.4.5	Cache validity	223
15.4.6	Dynamic carousel structure	224
15.5	Receiver demultiplexer resources	224
15.6	IC file system	224
15.7	MHEG profile of HTTP	224
15.7.0	Profile of HTTP	224
15.7.1	Protocol parameters	224
15.7.2	Methods	225
15.7.3	Response status codes	225
15.7.4	Header fields	225
15.7.4.1	Request header fields	225
15.7.4.1.0	Header fields that need not be sent	225
15.7.4.1.1	User agent string	226
15.7.4.2	Response header fields	226
15.7.4.3	General header fields	226
15.7.4.4	Entity header fields	227
15.7.5	Cookie support	227
15.8	Connection setup	227
15.9	Multiple connections	228
15.10	HTTP caching	228
15.11	Timeouts	228
15.12	Hybrid file system	229
15.12.1	Introduction	229
15.12.2	Resolution of References	229
15.12.3	Resolution Example	230
15.12.4	Hybrid file system caching	230
15.12.5	Synchronous file loading	230
15.12.6	Size of mapping table	230
15.12.7	Configure mapping table	230
15.12.8	Interaction with security (informative)	230
15.13	Authentication of applications	231
15.13.1	Overview (informative)	231
15.13.2	Authentication levels	232
15.13.3	Hash files	232
15.13.4	Signatures	233
15.13.5	Application signing certificates	233
15.13.6	Authentication process	234
15.13.6.0	Introduction	234
15.13.6.1	Hash file verification	234
15.13.6.2	Signature verification	234
15.13.6.3	Optimizations	235
15.13.6.4	Certificate caching	235
15.13.6.5	Handling race conditions	235
15.13.6.6	Redirection	235
15.13.7	Authentication file formats	235
15.13.7.1	Hash file	235
15.13.7.1.0	Hash file structure	235
15.13.7.1.1	Hash file pathname matching	237
15.13.7.1.2	Hash file location and naming conventions	237
15.13.7.1.3	Digest value computation rules	237
15.13.7.1.4	Special authentication rules	238

15.13.7.2	Signature file	238
15.13.7.2.1	Description	238
15.13.7.2.2	Signature file location and naming conventions.....	239
15.13.7.2.3	Supported algorithms.....	239
15.13.7.3	Certificate file	239
15.13.7.3.1	Description	239
15.13.7.3.2	ASN.1 encoding	240
15.13.7.3.3	Supported algorithms.....	240
15.13.7.3.4	Name matching.....	240
15.13.7.3.5	Certificate file location and naming conventions	240
15.13.7.3.6	Authentication rules.....	241
15.13.8	Example of creating an application that can be authenticated	241
15.13.8.1	Scenario (informative)	241
15.13.8.2	Hash and signature computations.....	241
15.13.8.2.1	Computation of the hashes of the app1 directory	241
15.13.8.2.2	Computation of the hashes of the app1/data1 directory.....	242
15.13.8.2.3	Computation of the hashes of the app1/data2 directory.....	242
15.13.8.2.4	Computation of the signatures	242
15.14	Controlling access to Internet servers.....	243
15.14.1	Overview	243
15.14.2	Restricting access to Internet servers	243
15.14.2.0	Overview.....	243
15.14.2.1	Server list file.....	243
15.14.2.1.1	Location.....	243
15.14.2.2	File format.....	244
15.14.2.3	Example	244
15.15	Transport Layer Security.....	244
15.15.1	Overview	244
15.15.2	TLS features that need not be implemented.....	244
15.15.3	TLS cipher suites	245
15.15.4	Server authentication for TLS.....	245
15.15.4.0	Overview.....	245
15.15.4.1	Certificate files.....	245
15.15.4.1.1	Location.....	245
15.15.4.1.2	File format	246
15.15.4.1.3	Certificate caching.....	246
15.16	HTTP profile for delivering encrypted streams.....	246
15.16.0	Scope	246
15.16.1	Additional response headers	246
15.16.1.0	Usage of additional response headers	246
15.16.1.1	X-Keys	246
15.16.1.2	X-KeyLocation.....	246
15.16.2	Redirects	246
15.16.3	Encrypted content	247
15.16.3.0	Introduction.....	247
15.16.3.1	Encryption scheme	247
15.16.3.2	Format of key file or X-Keys header.....	247
16	Name mapping	248
16.1	Names within the receiver	248
16.2	MHEG-5 component tags.....	248
16.3	Namespace mapping.....	248
16.3.0	Introduction.....	248
16.3.1	MHEG-5 object references	249
16.3.1.0	Mapping rules	249
16.3.1.1	MHEG-5 content references.....	249
16.3.1.2	DSMCC Stream objects	249
16.3.2	Mapping rules for GroupIdentifier and ContentReference	249
16.3.2.1	Void.....	249
16.3.2.2	Case sensitivity	249
16.3.2.3	Structure of file references	250
16.3.2.3.0	Syntax.....	250

16.3.2.3.1	Resolution of file references	251
16.3.2.3.2	Notes on resolving "../"	251
16.3.2.3.3	URI Syntax	251
16.3.2.3.4	Non-equivalence of URIs with different sources	251
16.3.2.3.5	Maximum reference length.....	251
16.3.2.3.6	Character codes	252
16.3.2.4	Shorthand notation	252
16.3.2.5	Reserved characters in HTTP URIs	252
16.3.3	URL formats for access to broadcast services	252
16.3.3.1	Numerical format	252
16.3.3.2	Logical Channel Number format.....	253
16.3.3.3	Handling duplicate services	253
17	MHEG-5 authoring rules and guidelines.....	253
17.1	Introduction	253
17.2	Avoiding confusion with navigator functions	253
17.2.1	Use of user inputs (mandatory).....	253
17.2.1.1	Requirement	253
17.2.1.2	Selecting and entering	254
17.2.1.3	Leaving	254
17.2.2	Broadcast-triggered native applications.....	254
17.3	Use of the "Text" and "Cancel" functions	254
17.3.0	Historical meaning of specific input functions	254
17.3.1	The traditional "teletext" key	254
17.3.1.0	Use of the "Text" key	254
17.3.1.1	Entering.....	254
17.3.1.2	Leaving	255
17.3.2	Accessing additional programme information.....	255
17.3.3	"Text" has no effect	256
17.3.4	On-screen prompts.....	256
17.4	Use of stream decoders.....	256
17.4.1	Number of decoders.....	256
17.4.2	Visible area	256
17.4.3	Conflicts between subtitles and MHEG-5 graphics	257
17.4.4	Accuracy of video positioning.....	257
17.4.5	Defensive behaviour (mandatory).....	257
17.5	Aspect ratio	258
17.5.1	Inheritance of video (mandatory).....	258
17.5.2	MHEG-5 only services	258
17.5.3	I-frames.....	258
17.5.4	Presentation of 16:9 AspectRatio Scenes on 4:3 displays.....	258
17.5.5	Presentation of 4:3 AspectRatio Scenes over HDMI.....	258
17.6	PNG bitmaps	258
17.6.1	Interlaced formats	258
17.7	Missed events	258
17.8	File naming.....	258
17.8.1	Name length (mandatory)	258
17.8.2	Name character coding	258
17.8.3	Case sensitive file names	258
17.8.4	File names in persistent storage (mandatory)	259
17.8.5	File names in true persistent storage (mandatory)	259
17.9	Text encoding.....	259
17.9.1	Mark-up	259
17.9.2	Text flow control	259
17.9.3	Width of row of characters	260
17.10	Reference checking	260
17.10.1	Application design issues when checking references	260
17.10.1.0	Introduction.....	260
17.10.1.1	Preloading is not mandatory.....	260
17.10.1.2	Stopping a reference check	260
17.10.2	Code example	260
17.11	Dynamically updated content	262

17.12	Stream events	262
17.13	User input events	262
17.13.1	Obtaining user input from an application with no Scene (mandatory)	262
17.13.2	Use of user input related engine events	262
17.14	Undefined behaviour (mandatory)	263
17.14.0	Avoid use of features with undefined behaviour	263
17.14.1	Synchronous event processing	263
17.14.2	Order of parallel links firing	263
17.15	Use of Call and Fork with ResidentPrograms	263
17.16	Catching failure of TransitionTo	263
17.16.1	Background	263
17.17	Elements of the object carousel	264
17.17.1	DIIs and elementary streams	264
17.17.2	Directory structure (mandatory)	264
17.17.3	Timeouts (mandatory)	264
17.17.4	Examples of object carousels	265
17.18	Possible uses of persistent storage (informative)	266
17.18.1	Start-up scene reference	266
17.18.2	Return application reference	267
17.18.3	Return application start-up scene reference	267
17.18.4	Persistent store size	267
17.18.5	Interaction channel	267
17.19	Use of true persistent storage	267
17.20	Hints and tips for good authoring	267
17.20.0	Introduction	267
17.20.1	Structure of the file system	267
17.20.1.1	Directories	267
17.20.1.2	File names	267
17.20.2	Structure of the object carousel	268
17.20.2.1	Placing associated objects in a module	268
17.20.2.2	Cache priority and modules	268
17.20.2.3	Object ordering	268
17.20.2.4	Conflict between MHEG cache policy and MHP cache_priority_descriptors	268
17.20.2.5	Carriage of content for HD receivers in DSMCC carousels	268
17.20.2.6	Carriage of "auth" files	269
17.20.3	Use of memory	269
17.20.3.1	Forked resident programs	269
17.20.3.2	Original content never goes away	269
17.20.3.3	Multiple references to the same content	269
17.20.3.4	Simultaneous file requests	269
17.20.4	Encoding of reserved fields	269
17.20.5	Presentation of short audio clips over HDMI	269
17.21	GetEngineSupport feature strings	270
17.21.1	Engine profile	270
17.21.2	Engine identification	270
17.21.3	Extensions	270
17.22	Appearance of Application Visible with no Scene	270
17.23	Colour representation	270
17.24	Text Width	270
17.25	Interaction Channel	270
17.25.1	Example of SetHybridFileSystem resident program	270
17.25.2	Example of ReturnData resident program	271
17.25.3	GroupIdentifiers in the hybrid file system	272
17.25.4	Cache priority in the IC file system	272
17.25.5	Use of Spawn with the hybrid file system	273
17.25.6	Interaction channel engine events	273
17.25.7	Presenting restricted content	274
17.25.8	Use of remote control keys to control A/V streams	274
17.25.9	Use of './' in the Hybrid File System mapping table	275
17.25.10	Use of SetCounterPosition on IC streams	275
17.25.11	CounterEndPosition of IC Streams	275
17.25.12	Cookie support	275

17.26	HD intelligent rendering.....	275
17.26.1	Sizing images for HD presentation	275
17.26.2	Appearance of objects on HD resolution graphics planes.....	275
17.26.3	PNG bitmap resolution	275
17.27	Non-destructive service tunes.....	276
17.27.1	Non-availability of broadcast file system during tune	276
17.27.2	Carousel structure	276
17.27.3	Behaviour of Spawn during tune	276
17.28	Signalling MHEG Applications for all types of Terminal.....	276
17.29	Avoid moving carousel components	276
17.30	Value of LastAnchorFired for boundary events	276
17.31	Case sensitivity in server names.....	277

Annex A (informative): The user experience278

A.1	Introduction	278
A.1.0	Scope	278
A.1.1	Visual appearance.....	278
A.1.1.1	Balance of AV and MHEG-5.....	278
A.1.1.2	Time and space	279
A.2	User navigation	280
A.2.1	Channel change	280
A.2.1.0	Introduction.....	280
A.2.1.1	The "Text" button	281
A.3	Channel selections within an information service.....	282
A.4	Use of video within an information service	284

Annex B (normative): Allocation of codes for use with the present document285

B.1	Application type codes	285
B.2	UniversalEngineProfile(N).....	285
History	286

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 (<https://ipr.etsi.org>).

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

Foreword

This final draft ETSI Standard (ES) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECTrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI), and is now submitted for the ETSI standards Membership Approval Procedure.

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

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

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](#) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

The present document provides a Profile of the International MHEG-5 Standard specification ISO/IEC 13522-5 [14] 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification 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

1.0 Introduction

The present document describes a complete system 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.

The present document contains a number of clarifications related to streaming content and stream event handling and various other changes to increase interoperability of implementations of the specification.

1.1 Localizing the present document

1.1.0 Approach to creating a local implementation

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

In addition, there is functionality which should be considered in a local implementation of the Profile. This information should be put into a localized profile of the ETSI-MHEG Profile. The following issues are addressed when determining what information is necessary in a localized profile:

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

1.1.1 Extensions

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

For a particular implementation, a localized profile specifies exactly which extensions are mandatory or optional.

- Signalling extension:
 - Functionality for identifying and booting an application is provided by the PMT and ServiceGateway extension (see clause 9.3).
- Service Information extension:
 - Functionality for retrieving DVB Service Information (SI) is provided by the SI extension (see clause 11.14). A localized profile states whether this extension is mandatory or optional.
- HDGraphicsPlaneExtension (see clause 12.11.1):
 - This extends the MHEG-5 engine to allow support for an HD graphics plane.
- HDVideoExtension (see clause 12.11):
 - This extends the MHEG-5 engine to allow support for HD video and audio coding and presentation.
- InteractionChannelExtension (see clause 15.1.2 and table 11.14):
 - This extends the MHEG-5 engine to allow support for static content retrieval over an always-on IP connection.

- ICStreamingExtension (see clause 11.5.4 and table 11.15):
 - This extends the MHEG-5 engine to allow support for static and streaming content retrieval over an always-on IP connection. Receivers that implement ICStreamingExtension also implements InteractionChannelExtension.
- LifecycleExtension (see clause 8.1.7):
 - This extends the MHEG-5 engine to allow MHEG-5 applications to continue running across service tunes.
- NativeApplicationExtension (see clause 17.2.2 and table 11.16):
 - This extends the MHEG-5 engine to allow the co-existence of MHEG-5 applications and broadcast-triggered native applications.
- DownloadableFontExtension (see clause 13.3.1):
 - This extends the MHEG-5 engine to allow support for a range of typefaces, styles and sizes for text presentation.
- PVRExtension (see clause 11.15):
 - This extends the MHEG-5 engine to allow support for booking events on a PVR from an application.
- InputMaskExtension (see clause 11.14.2):
 - This extends the MHEG-5 engine to allow fine grained control of user input.
- File System Acceleration Extension (see clause 11.16):
 - This extends the MHEG-5 engine to allow receivers to persistently store a preselected part of a File System in order to improve the speed of subsequent file access.

NOTE: Extensions are not partially implemented.

1.1.2 Allocation of codes

The present document provides for an "application_type_code" (see clause 9.3.2.1) 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 is specified in the localized profile for that implementation.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

- [1] ETSI EN 300 468 (V1.5.1): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".
- [2] ETSI EN 301 192 (V1.2.1): "Digital Video Broadcasting (DVB); DVB specification for data broadcasting".
- [3] ETSI TS 102 809 (V1.2.1): "Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments".
- [4] Void.
- [5] CENELEC EN 50221: "Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications".
- [6] Recommendation ITU-T X.680: "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

NOTE: Also published as ISO/IEC 8824-1.

- [7] Recommendation ITU-T X.690: "Information technology - ASN.1 Encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

NOTE: Also published as ISO/IEC 8825-1.

- [8] 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: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 1: Systems".
- [10] ISO/IEC 13818-2: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 2: Video".
- [11] ISO/IEC 13818-3: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 3: Audio".
- [12] ISO/IEC 13818-6: "Information technology -- Generic coding of moving pictures and associated audio information -- Part 6: Extensions for DSM-CC (Digital Storage Media Command and Control)".
- [13] ISO/IEC 10646: "Information technology -- Universal Coded Character Set (UCS)".
- [14] ISO/IEC 13522-5: "Information technology -- Coding of multimedia and hypermedia information -- Part 5: Support for base-level interactive applications".
- [15] Void.
- [16] DAVIC 1.4.1 Specification Part 9: "Information Representation".

- [17] W3C Recommendation (10 November 2003): "Information technology - Computer graphics and image processing - Portable Network Graphics (PNG): Functional specification", ISO/IEC 15948:2003 (E).
- NOTE: Available at <http://www.w3.org/TR/PNG/>.
- [18] ETSI TS 101 699 (V1.1.1): "Digital Video Broadcasting (DVB); Extensions to the Common Interface Specification".
- [19] Void.
- [20] Void.
- [21] ISO/IEC 13522-5:1997/Cor.1:1999(E): "Information technology - Coding of multimedia and hypermedia information - Part 5: Support for base-level interactive applications. corrigendum 1".
- [22] Recommendation ITU-R BT.470-7 (02/05): "Conventional analogue television systems".
- [23] Void.
- [24] Recommendation ITU-R BT.601: "Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios".
- [25] IETF RFC 1951: "DEFLATE Compressed Data Format Specification version 1.3".
- NOTE: Available at <http://www.rfc-editor.org/rfc/rfc1951.pdf>.
- [26] W3C Recommendation: "HTML 4.01 Specification".
- NOTE: Available at <http://www.w3.org/TR/html401/>.
- [27] IETF RFC 1738: "Uniform Resource Locators (URL)".
- [28] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".
- NOTE: Available at <http://www.rfc-editor.org/rfc/rfc3986.txt>.
- [29] ISO/IEC 9995-8 (2009): "Information technology - Keyboard layouts for text and office systems - Part 8: Allocation of letters to the keys of a numeric keypad".
- [30] IETF RFC 2616: "Hypertext Transfer Protocol - HTTP/1.1".
- [31] IETF RFC 2818: "HTTP Over TLS".
- [32] IETF RFC 6265: "HTTP State Management Mechanism".
- [33] IETF RFC 2459: "Internet X.509 Public Key Infrastructure Certificate and CRL Profile".
- [34] Void.
- [35] Recommendation ITU-T X.509: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".
- [36] ETSI TS 101 812 (V1.3.2): "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3".
- [37] IETF RFC 2246: "The TLS Protocol Version 1.0".
- [38] IETF RFC 3268: "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)".
- [39] Recommendation ITU-T H.264: "Advanced video coding for generic audiovisual services"/ ISO/IEC 14496-10 (2005): "Information Technology - Coding of audio-visual objects - Part 10: Advanced Video Coding".
- [40] ETSI TS 101 154 (V1.8.1): "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream".

- [41] ETSI TS 102 366 (V1.2.1): "Digital Audio Compression (AC-3, Enhanced AC-3) Standard".
- [42] ISO/IEC 14496-3 (2009): "Information technology -- Coding of audio-visual objects -- Part 3: Audio".
- [43] ISO/IEC 14496-22 (2009): "Information technology -- Coding of audio-visual objects -- Part 22: Open Font Format".

NOTE: Available at <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>, and also at <http://www.microsoft.com/typography/otspec/default.htm>.

- [44] ETSI TS 102 822 (parts 1 to 9): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime")".
- [45] ETSI TS 102 323 (V1.3.1): "Digital Video Broadcasting (DVB); Carriage and signalling of TV-Anytime information in DVB transport streams".
- [46] Tiresias™ Screenfont: "Version 8.04 Character Set".

NOTE: Available at <http://catalog.monotype.com/contact>.

- [47] ISO/IEC 10918-1: "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines".
- [48] JPEG File Interchange Format, Version 1.02, September 1st 1992.

NOTE: Available at <http://www.w3.org/Graphics/JPEG/jif3.pdf>.

- [49] FIPS PUB 180-1: "Secure hash standard".
- [50] FIPS PUB 197:2001: "Advanced Encryption Standard (AES)".
- [51] ANSI/SCTE 52:2003: "Data Encryption Standard - Cipher Block Chaining Packet Encryption Specification".
- [52] IEC 62455:2011: "Internet protocol (IP) and transport stream (TS) based service access".
- [53] IETF RFC 3447: "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography" version 2.1.
- [54] Tiresias™ Screenfont: "Version 7.51 Character Set".

NOTE: Available at <http://catalog.monotype.com/contact>.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] IETF RFC 1950: "ZLIB Compressed Data Format Specification version 3.3".
- [i.2] ETSI TR 101 154 (V1.4.1): "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications".
- [i.3] ETSI TR 101 211 (V1.5.1): "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

- [i.4] ETSI TR 101 202 (V1.1.1): "Digital Video Broadcasting (DVB); Implementation guidelines for data broadcasting".
- [i.5] Void.
- [i.6] IETF RFC 1034: "Domain Names - Concepts and Facilities".
- [i.7] IETF RFC 1035: "Domain Names - Implementation and specification".
- [i.8] IETF RFC 1982: "Serial Number Arithmetic".
- [i.9] IETF RFC 2131: "Dynamic Host Configuration Protocol".
- [i.10] IETF RFC 2132: "Dynamic Host Configuration Protocol Options and BOOTP Vendor Extensions".
- [i.11] IETF RFC 2181: "Clarifications to the Domain Name System Specification".

3 Definitions and abbreviations

3.1 Definitions

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

access unit: 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: organization which assembles a sequence of events or programmes to be delivered to the viewer based upon a schedule

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

Content Reference Identifier (CRID): identifier for content that is independent of its location (defined in ETSI TS 102 822 [44])

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

elementary stream: generic term for one of the coded video, coded audio or other coded bit streams in PES packets

NOTE 1: See ISO/IEC 13818-1 [9].

NOTE 2: One elementary stream is carried in a sequence of PES packets with one and only one `stream_id`.

event: 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].

Instantaneous Decoding Refresh (IDR) access unit: Defined in Recommendation ITU-T H.264/ISO/IEC 14496-10 [39].

Instantaneous Decoding Refresh (IDR) picture: Defined in Recommendation ITU-T H.264/ISO/IEC 14496-10 [39].

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 (MHEG): See ISO/IEC 13522-5 [14].

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

network: 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

NOTE: Defined in ETSI EN 300 468 [1].

packet identifier: 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: data structure used to carry elementary stream data: it consists of a PES packet header followed by PES packet payload

NOTE: See ISO/IEC 13818-1 [9].

PES packet header: 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 1: See ISO/IEC 13818-1 [9].

NOTE 2: 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.

Picture Parameter Set (PPS): Defined in Recommendation ITU-T H.264/ISO/IEC 14496-10 [39].

presentation unit: decoded audio access unit or a decoded picture

NOTE: See ISO/IEC 13818-1 [9].

program: 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: 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 are set to "1".

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

Sequence Parameter Set (SPS): Defined in Recommendation ITU-T H.264/ISO/IEC 14496-10 [39].

service: 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

NOTE: 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]) 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".

Supplemental Enhancement Information (SEI): Defined in Recommendation ITU-T H.264/ISO/IEC 14496-10 [39].

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

table_id: identifier of an MPEG table

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

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

transport stream: Defined in ISO/IEC 13818-1 [9].

transport_stream_id: unique identifier of a transport stream within an original network

NOTE: Defined in ETSI EN 300 468 [1].

3.2 Abbreviations

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

AAC	Advanced Audio Coding
AC-3	Audio Coding 3 (aka Dolby Digital 5.1 Channel)
AES	Advanced Encryption Standard
AFD	Active Format Descriptor
AIT	Application Information Table
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN	Abstract Syntax Notation
AV	Audio and Video
AVC	Advanced Video Coding
BAT	Bouquet Association Table

BIOP	Broadcast Inter-Orb Protocol
CA	Conditional Access
CBC	Cipher Block Chaining
CCP	Content Cache Priority
CENELEC	European Committee for Electrotechnical Standardization
CI	Common Interface
CID	Content Identifier Descriptor
CLUT	Colour Look Up Table
CORBA	Common Object Request Broker Architecture
CR	Carriage Return
CRID	Content Reference Identifier
CRL	Certificate Revocation List
CRLF	Carriage Return / Line Feed
DAD	Default Authority Descriptor
DDB	Download Data Block
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DII	Download Info Indication
DSI	Download Server Initiate
DSM	Digital Storage Media
DSM-CC	Digital Storage Media - Command and Control
DSMCC	Digital Storage Media Command and Control
DTG	Digital Television Group
DTT	Digital Terrestrial Television
DVR	Digital Video Recorder
E-AC3	Enhanced Audio Coding 3 (aka Dolby Digital Plus)
EIT	Event Information Table
EPG	Electronic Programme Guide
ES	Elementary Stream
FIPS	Federal Information Processing Standard
H.264/AVC	H.264/Advanced Video Coding
HD	High Definition
HDCP	High-bandwidth Digital Content Protection
HDMI	High Definition Multimedia Interface
HMAC	Hash Message Authentication Code
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HTTPS	HTTP Secure
IC	Interaction Channel
ID	Identifier
IDR	Instantaneous Decoding Refresh
IOR	Interoperable Object Reference
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-R	International Telecommunication Union - Radiocommunication
ITU-T	International Telecommunication Union - Telecommunication
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LCN	Logical Channel Number
LF	Line Feed
LSB	Least Significant Bit
MD5	Message Digest Algorithm 5
MHEG	Multimedia and Hypermedia Experts Group
MHP	Multimedia Home Platform
MIME	Multipurpose Internet Mail Extensions
MMI	Man-Machine Interface
MPEG	Motion Picture Experts Group
NIT	Network Information Table
NPT	Normal Play Time
NSAP	Network Service Access Point
OC	Object Carousel

ORB	Object Request Broker
OSD	On Screen Display
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PFR	Portable Font Resource
PID	Packet IDentifier
PMT	Program Map Table
PNG	Portable Network Graphics
PPS	Picture Parameter Set
PSI	Program Specific Information
PVR	Personal Video Recorder
RAM	Random Access Memory
RGB	Red, Green, Blue
RSA	Rivest Shamir Adleman algorithm (for public-key cryptography)
SD	Standard Definition
SDT	Service Description Table
SEI	Supplemental Enhancement Information
SHA	Secure Hash Algorithm
SI	Service Information
SMS	Short Message Service
SPS	Sequence Parameter Set
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TS	Transport Stream
TV	Television
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator/Location
UTF	Unicode Transformation Format
UTF-8	Universal Character Set Transformation Format, 8-bit format
WSS	Wide-Screen Signalling
XMLAIT	Application Information Table in XML

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 5.1. In figure 5.1, the return channel shown is optional and is not described in the present document.

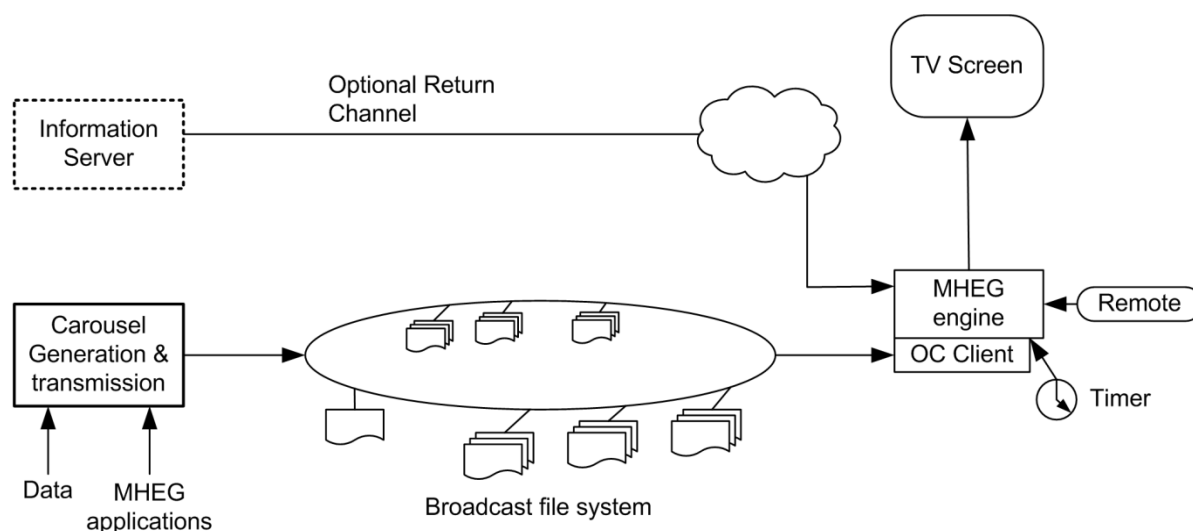


Figure 5.1: Basic Architecture

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 the present document MHEG-5 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 present document, 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 necessary. In the present document 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 5.2 shows the MHEG-5 engine and its interaction with the other parts of the receiver.

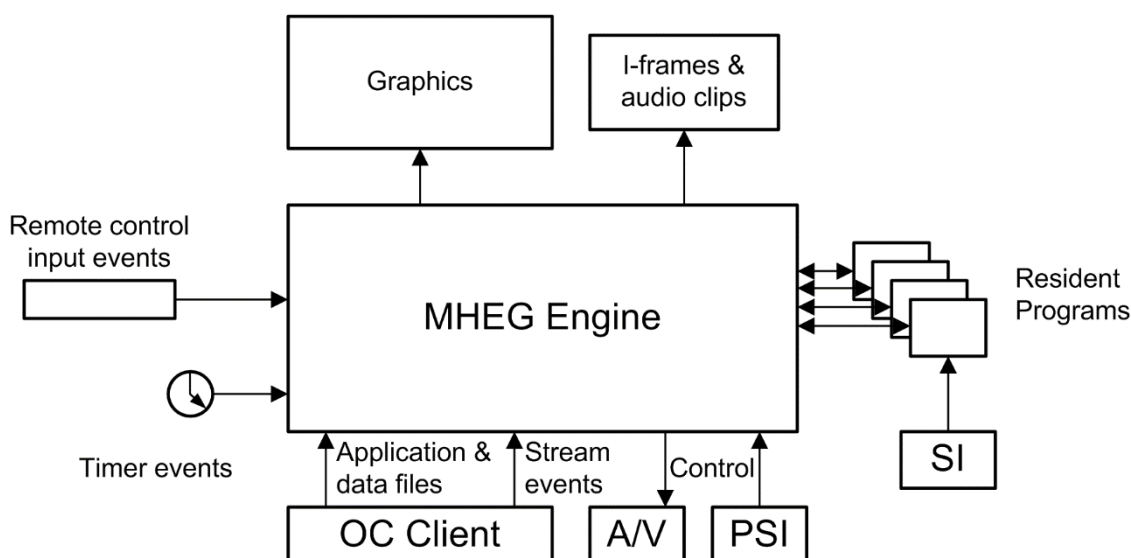


Figure 5.2: MHEG-5 engine and its interaction with a receiver

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 11.7.

7.3 Resident fonts

See clause 13.3.2.

7.4 Colour representation

See clause 12.3.

8 Application model

8.0 Introduction

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 14.10.3.2);
- 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 clauses 14.10.3.2 and 14.10.3.6);
- 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).

Several different file systems are potentially available to an application:

- the broadcast file system delivered by the DSM-CC object carousel (see clause 15.1.1);
- the file system provided by a CI module (see clause 14.10.2);
- for receivers that implement InteractionChannelExtension both the IC file system (see clause 15.1.2) and the hybrid file system.

The conditions that lead to these file systems being available are specified in clause 8.1.6.

8.1.2 Preparing for launch

Before implementing Step 1 of the behaviour of the Launch action described in the ISO/IEC 13522-5 [14] (MHEG-5) specification 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

8.1.3.0 When to launch an auto-boot application

Whenever a service that includes an auto-boot MHEG-5 application as a component is selected, 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.

8.1.3.1 How an auto-boot application is done

The receiver shall look 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 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 shall be launched;
- if an MHEG-5 application Quits the receiver shall attempt to launch a new application. See clauses 8.2 and 9.3.

8.1.4 Auto kill application

Whenever the receiver changes channel, except as a result of a non-destructive tune, any executing MHEG-5 application shall be killed, the application stack shall be cleared and all file systems shall be unmounted.

Therefore if the user interacts with the receivers navigator functions to change channel the current application shall be killed. If the new channel has an application this one shall be launched. If the application invokes a destructive channel change (via a resident program) (see clauses 8.1.7 and 11.10.8.4.1), and the channel change is successful, the current application shall 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

8.1.5.0 Identification of a file reference source

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 shall be defined.

8.1.5.2 Current carousel

At first the Current carousel shall be the same as the Initial carousel (clause 8.1.5.1). 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 receiver shall perform a full autoboot as defined in clause 9.3 in the following cases:

- if the current application quits and the application stack is empty;
- following a RequestStart (clause 14.10.3.2) with a non-empty initial object or AppAbortRequest (clause 14.10.3.6) message from a CI module.

8.1.5.3 Current source

The "Current Source" is the file system from which the currently executing application object was delivered. The "Current Source" is used to resolve file references that do not explicitly include a source.

For example, if the currently executing application object is delivered by the CI file system then a source of "CI:" shall be assumed when resolving file references that do not explicitly specify the source. If the currently executing application object is delivered by a broadcast Object Carousel then the "Current Source" shall be the "Current Carousel" and a source of "DSM:" shall be assumed when resolving file references that do not explicitly specify the source.

If the receiver implements the InteractionChannelExtension then the "Current Source" can be the IC file system or the hybrid file system if the currently executing application object is delivered by these mechanisms. A source of "http:", "https:" or "hybrid:" shall be assumed as appropriate. During the auto-boot process the "Current Source" shall be initialized to "hybrid:".

8.1.5.4 Authentication Source

The "Authentication Source" is the file system from which the Current Application was loaded, unless the Application was loaded from the Interaction Channel. In this case the "Authentication Source" is the file system of the last Application that was not loaded from the interaction channel, i.e. the "Current Carousel" for broadcast applications or "CI:/" for CI applications.

The "Authentication Source" is used to resolve file references to application authentication certificates (see clause 15.13.7.3.5), TLS certificates (see clause 15.14.2.1.1) and server list files (see clause 15.15.4.1.1).

8.1.6 Accessible file systems

8.1.6.0 List of available file systems

The following file systems are potentially available to an application:

- The broadcast file system delivered by the DSM-CC Object Carousel (see clause 15). This, if present, shall be mounted as a consequence of selecting a broadcast service (see clause 9.3.7.1).
- The file system provided by a CI module. This, if present, shall be mounted because of activity by a CI module (see clause 14.10).

Additionally, if the receiver implements InteractionChannelExtension the following file systems are potentially available to an application:

- The IC file system. This may be available as a consequence of an IP connection being available to the receiver (see clause 15.6).
- The hybrid file system, which manages access to other file systems and shall always be available to applications (see clause 15.12).

Any of these file systems may be mounted simultaneously.

8.1.6.1 Broadcast applications

All auto-boot broadcast applications, and applications descended from them, shall have access to files in the "Current carousel" (see clause 8.1.5.2).

A broadcast application will also have access to files provided by the CI module if a CI module has previously sent a RequestStart (see clause 14.10.3.2) message and the session from the module is still open.

8.1.6.2 CI introduced applications

All applications introduced by a CI module, and applications descended from them, shall 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 shall be the Initial carousel (see clause 8.1.5.1) of the service (a side effect of a RequestStart (see clause 14.10.3.2) that introduces the new initial application from the CI module). However, application action can cause the Current carousel (see clause 8.1.5.2) to change.

8.1.6.3 Interaction Channel

If the receiver implements InteractionChannelExtension additional file systems become available.

All applications running on a receiver which has an available IP connection have access to files in the IC file system. To do this, references which are valid http or https URIs are required.

All applications have access to the hybrid file system. Depending on the configuration of the hybrid file system mapping table references may resolve to references to the broadcast, CI or IC file systems.

8.1.7 Keeping an application running across service tunes

8.1.7.0 Required behaviour for non-destructive tune

It is possible to attempt to keep an application running across a service tune (referred to as a "non-destructive tune") by setting the `app_keeprunning_flag` within the `tuneinfo` argument of the `SI_TuneIndexInfo` Resident Program. The steps involved in this are defined in clause 11.10.8.4. Support for this feature is referred to as `LifecycleExtension`.

This clause describes behaviour required for receivers that implement `LifecycleExtension`. The new service may be the same as the currently tuned service; there is no special behaviour in this case and the service tune shall be executed as if the two services were different.

During the period of the tune itself and the subsequent attempt to attach to a carousel in the new service the running application shall continue to operate in the same manner as if no service tune had occurred: in particular there shall be no disruption to the presentation of I-Frames and/or audio from memory at any point during this process. The only exceptions to this are defined in the remainder of this clause (clause 8.1.7.1 to clause 8.1.7.10) as follows.

8.1.7.1 Broadcast file system requests

During the non-destructive tune there will be a period where there is no mounted carousel and so no broadcast file system will be available to the running application. Consequently any outstanding requests for items in the broadcast file system at the start of this period shall be queued and any new requests during this period added to the queue. This shall apply to all requests even if they could potentially be resolved directly from receiver cache, i.e. requests made with a non-zero even value of `ContentCachePriority` (see clause 15.4.3) or `GroupCachePriority` (see clause 15.4.4).

However, this shall not apply to requests made to other file systems such as IP requests. This shall also not apply to requests made to the hybrid file system (see clause 15.12) unless such a request is resolved as a broadcast file system request.

Once the non-destructive service tune has been successfully completed and a carousel in the new service has been attached, all queued requests shall be processed in the context of this new carousel.

Only requests made with a non-zero even value of `ContentCachePriority` or `GroupCachePriority` may be resolved using files obtained and cached from a previous carousel. It is not possible to resolve requests using a transparent cache as version numbers may not be the same between the new and previous object carousels.

8.1.7.2 Timers

The receiver shall maintain any outstanding or new timers. However it is recognized that the accuracy of any active timers may degrade during the tune itself.

8.1.7.3 Carousel Identity

On successful mount of a carousel in the new service (see clause 11.10.8.4) this carousel shall become the `Current Carousel` (see clause 8.1.5). The running application shall be considered to have originated from the mounted `DSM-CC carousel` and as such may be terminated in accordance with `Life Cycle Signalling` as defined in clauses 9.3.6 and 9.3.7.

If an auto-mount broadcast file system is present in the new service (see clause 9.3.7) this shall become the `Initial Carousel` (see clause 8.1.5), otherwise no `Initial Carousel` is defined.

Once the tune to the new service has been completed and the receiver has identified and attached to the requested carousel, the following steps shall be executed:

- 1) Application Stack:
 - The application stack shall be reset, i.e. all applications shall be removed, in line with clause 8.2. The currently running application shall remain on the application stack.
- 2) Indication of successful completion:
 - A "NonDestructiveTuneOK" Engine Event shall be generated (see clause 11.8).

NOTE: Receivers are not required to acquire a DSI from the new carousel before beginning these steps.

8.1.7.4 Broadcast file system

The carousel attached to as part of a non-destructive service tune (the new Current Carousel) need not be the same as the (previous) Current Carousel in the previous service at the point that the application initiated the non-destructive tune. This is true both in terms of the broadcast file system that it delivers and the encoding of the underlying Object and Data Carousel structures. Furthermore, this broadcast file system need not contain the currently running MHEG-5 Application object. See also clause 17.27.

8.1.7.5 Network and service boot info

Following a non-destructive service tune, the receiver shall examine the PMT of the new service to look for a new source of service- and network-level application lifecycle signalling as follows:

- The receiver shall look for a data_broadcast_id_descriptor on whichever component carries the carousel_id_descriptor for the carousel selected by the non-destructive tune process. The receiver shall use the NB_info and service_boot information parsed from the new data_broadcast_id_descriptor if one exists. If this descriptor is not found, the receiver shall continue to use the service_boot information from the previous service but shall discard any NB_info field from the previous service.
- The receiver shall ignore any change in the value of the NB_version field between the last PMT of the previous service and the first received PMT of the new service.
- Regardless of whether the receiver uses information from a service_boot_info sub-descriptor from the new or previous service, it shall resolve the assocTag value using the new PMT.

8.1.7.6 Behaviour of stream decoders

The running application may include active MHEG-5 Stream objects at the point of initiating the non-destructive service tune.

If the Storage attribute for an active MHEG-5 Stream object is set to memory, or if the Storage attribute is set to stream and the reference is not to a broadcast stream, then decoding and presentation of the relevant stream components shall continue without disruption as if no service tune had occurred.

If the Storage attribute for an active MHEG-5 Stream object is set to stream and references a broadcast stream then:

- If the target service for the non-destructive tune is in the same MPEG transport stream as the current service then decoding and presentation of the stream components based on the current context shall continue until the tune has been successful at which point the effect of the MHEG-5 Stream object is re-evaluated (see below).
- If the target service for the non-destructive tune is in a different MPEG transport stream to the current service then decoding and presentation of the stream components based on the current context may cease at any point following initiation of the tune by the application. In this case video presentation shall default to black and audio output shall be muted.

Once the new service has been tuned to:

- 1) The URLs "rec://svc/def" and "rec://svc/cur" shall change to mean the new service for this re-evaluation and for future references.

- 2) Any prepared MHEG-5 Stream objects shall be re-evaluated according to the context of the new service as if a SetData action had occurred. Where the Content internal attribute for a prepared MHEG-5 Stream object contains a URL format reference, re-evaluation can commence immediately, without having to wait for a carousel in the new service to be attached.

NOTE: Where a Stream object uses an explicit reference before a non-destructive tune, this will mean that subsequent use of "rec://svc/cur" will be resolved differently from "rec://svc/def".

The receiver shall not fire a StreamStopped or StreamPlaying event during a successful tune. However, should the receiver fail to re-evaluate a running MHEG-5 Stream object on re-attachment, presentation of the Stream object shall cease and the receiver shall fire a StreamStopped event.

At no point during or after the tune should the receiver present the user with any video, audio or subtitles unless explicitly requested by the application.

8.1.7.7 Content management

Where the content provided by an MHEG-5 Stream object changes following a non-destructive tune, receivers shall ensure that the content management controls that are observed are those that apply to the service containing the newly selected stream components.

8.1.7.8 Receiver keys

Any of the receiver's keys (which is outside the set of keys that generate MHEG-5 UserInput events) that immediately terminates any running MHEG-5 application shall return the viewer directly to the channel associated with the viewer service context (see clause 11.10.8.2).

NOTE: This channel may not be the active channel if an MHEG-5 application has performed a channel tune with tune_quietly_flag set.

8.1.7.9 Interaction with resident programs

If a non-destructive tune occurs while any of the following resident programs has been invoked and the procedural code for that resident program is still running, the effect of the resident program shall be undefined.

- SI_TuneIndex.
- SI_TuneIndexInfo.
- GetBootInfo.

8.1.7.10 Interaction channel security

Following a non-destructive tune, any cached application signing certificates, server list files and TLS certificates shall be invalidated. When an application signing certificate, server list file or TLS certificate is next required, an up-to-date version shall be obtained from the attached carousel in the new service.

8.1.7.11 Autoboot behaviour

When the current application quits and the application stack is empty, the receiver shall perform a full autoboot as defined in clause 9.3.

8.1.7.12 True persistent storage

Following a non-destructive tune, the current service for the purposes of writing to true persistent storage shall be the service associated with the viewer service context (clause 14.7).

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.

The structure and mapping rules that apply to `GroupIdentifiers` and `ContentReferences` are defined in clauses 11.11.5. and 16.3.2. The `LiteOptionsProfileBody` is used for making links to objects carried in other object carousels (see clause 15.2.5.2).

This is in accordance with the ISO/IEC 13522-5 [14] (MHEG-5) specification. Note particularly the corrigenda (ISO/IEC 13522-5:1997/Cor.1:1999(E) [21]) 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.

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

- each time there is a service change;
- if invalid information is found on the stack;
- if an application is introduced by a CI module (using a `RequestStart` (see clause 14.10.3.2) message) displacing an already executing application.

9 Signalling

9.1 Introduction to application lifecycle signalling

9.1.0 Overview

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 necessary 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 Scope

The following application lifecycle specification is designed for operation with broadcast DSM-CC carousels and CI file systems and addresses the interaction channel for receivers that implement InteractionChannelExtension.

9.2 AIT Signalling Extension

9.2.0 AIT profile

Terminals shall support AIT, XMLAIT and metadata signalling of MHEG applications as defined in ETSI TS 102 809 [3] and further profiled in table 9.1.

Table 9.1: MHEG AIT Signalling

ETSI TS 102 809 [3]	M/O/NI (see note)	Notes								
5.2.2 Application types	M	The value of the application type field for the MHEG presentation technology is 0x0008 as defined by DVB for broadcast AIT and "application/vnd.etsi.mheg5" for the XMLAIT.								
5.2.3 Application Identification	M	Application ids shall be in the range of unsigned applications as defined in ETSI TS 102 809 [3].								
5.2.4 Application Control Codes	M	The following control codes shall be supported: <table border="1" data-bbox="762 1055 1321 1173"> <thead> <tr> <th>MPEG-2 Encoding</th> <th>Identifier</th> </tr> </thead> <tbody> <tr> <td>0x01</td> <td>AUTOSTART</td> </tr> <tr> <td>0x02</td> <td>PRESENT</td> </tr> <tr> <td>0x03</td> <td>KILL</td> </tr> </tbody> </table>	MPEG-2 Encoding	Identifier	0x01	AUTOSTART	0x02	PRESENT	0x03	KILL
MPEG-2 Encoding	Identifier									
0x01	AUTOSTART									
0x02	PRESENT									
0x03	KILL									
5.2.5 Platform Profiles	M	The value of the application_profile shall be as defined for application_type_code as in clause B.1. The value of the version fields shall be as follows: <table border="1" data-bbox="762 1256 1321 1375"> <thead> <tr> <th>Field</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>version.major</td> <td>1</td> </tr> <tr> <td>version.minor</td> <td>1</td> </tr> <tr> <td>version.micro</td> <td>1</td> </tr> </tbody> </table>	Field	Value	version.major	1	version.minor	1	version.micro	1
Field	Value									
version.major	1									
version.minor	1									
version.micro	1									
5.2.6 Application Visibility	M	As defined in ETSI TS 102 809 [3]. Shall be set to NOT_VISIBLE_ALL								
5.2.7 Application Priority	M	As defined in ETSI TS 102 809 [3].								
5.2.8 Application Icons	NI									
5.2.9 Graphics Constraints	NI									
5.2.10 Application Usage	NI									
5.2.11 Stored Applications	NI									
5.2.12 Application Description File	NI									
5.3.2 Program Specific Information	M	As defined in ETSI TS 102 809 [3].								
5.3.4 Application Information Table	M	A maximum of one sub-table shall be transmitted per service. The minimum repetition rate for the AIT sub table shall be 1 second. Applications removed from the AIT sub-table which was previously signalled but where the AIT sub-table remains present in the network shall be stopped as if they had been signalled with a KILL control code.								

ETSI TS 102 809 [3]	M/O/NI (see note)	Notes
5.3.4.7 Visibility of AIT	M	If an MHEG-5 application performs a destructive service tune 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, see clause 11.10.6.4. If an MHEG-5 application performs a non-destructive service tune away from the current service then the application behaviour will be as defined by clause 8.1.7. For destructive and non-destructive tunes, the receiver shall ignore the <code>service_bound_flag</code> in the Application descriptor.
5.3.4.9 Access to an MPEG-2 format AIT via a broadband connection	O	AIT via a broadband connection shall reference an MHEG application delivered by a broadcast channel.
5.3.5.1 Application Signalling Descriptor	M	As defined in ETSI TS 102 809 [3].
5.3.5.2 Data Broadcast Id Descriptor	O	As defined in ETSI TS 102 809 [3]. ETSI TS 102 809 [3] allows the the Data Broadcast Id Descriptor to be used as a shortcut to the carousel allowing the terminal to mount the carousel before acquiring the AIT. It may used with wildcard data broadcast id values 0x00f0 and 0x00f1 and an optional list the app types OR with the MHEG data broadcast id value of 0x0106. If the data broadcast id is 0x0106 then the format of descriptor shall be as defined by classical signalling in clause 9.3.2.1. Terminals that support MHEG AIT may optionally interpret both formats of Data Broadcast Id Descriptor as a shortcut to the carousel but shall ignore the id specific data when the data broadcast id is 0x0106.
5.3.5.3 Application Descriptor	M	When a receiver changes channel the value of the <code>service_bound_flag</code> shall not affect application lifecycle (see Visibility of AIT).
5.3.5.4 Application Recording Descriptor	NI	
5.3.5.5 Application Usage Descriptor	NI	
5.3.5.6 User Information Descriptors	NI	
5.3.5.7 External Authorization Descriptor	NI	
5.3.5.8 Graphics Constraints Descriptor	NI	
5.3.6 Transport Protocol Descriptors	M	Terminals shall support the Object Carousel transport protocol as defined in ETSI TS 102 809 [3]. Receivers need not support other transport protocols.
5.3.7 Simple Application Location Descriptor	M	As defined in ETSI TS 102 809 [3].
5.3.8 Simple Application Boundary Descriptor	NI	
5.3.9 Service Information Descriptor	NI	
5.3.10 Stored Applications Descriptor	NI	
NOTE:	M: Mandatory to provide. O: Optional to support. NI: Not Included (may be ignored by Terminals).	

9.2.1 Coexistence of Classical and AIT signalling

Two types of application signalling are available:

- Classical application signalling as described in clause 9.3 (Application identification and boot).
- AIT signalling as described in this clause.

Classical application signalling, AIT signalling, both signalling types or neither may be present for a service.

Receivers supporting AIT signalling extension shall support the application signalling precedence rules defined in this clause.

If an AIT subtable for a service is present, then a receiver shall process the AIT signalling and shall ignore any classical application signalling. If an AIT subtable for a service is not present, then a receiver shall process any available classical application signalling for MHEG.

The "presence of an AIT subtable for a service" is defined as either of the following:

- An `application_signalling_descriptor` without a sub-loop is present in any `ES_loop` of the PMT for that service AND at least one AIT subtable referenced from these ES loop(s) has a supported `application_type`.
- An `application_signalling_descriptor` with a sub-loop containing a supported `application_type` is present in any `ES_loop` of the PMT for that service.

A supported `application_type` is defined such that the receiver has the ability to launch an application of any of the signalled `application_types`.

9.2.2 Life-cycle signalling in AIT and PMT

9.2.2.0 Life-cycle information in the PMT

Existing service-level and network-level signalling is specified using certain descriptors and sub-descriptors in the PMT. When an interactive application uses information from the AIT, any life-cycle related information from the PMT shall be ignored.

This restriction applies to the PMT that contains the `carousel_id_descriptor` for the current object carousel, including the object carousel that contains the auto-boot application and any object carousel that is mounted as a result of `LiteOptionsProfileBody` object references or a non-destructive service tune.

9.2.2.1 `carousel_id_descriptor`

The existence of the `carousel_id_descriptor` in the PMT is mandatory as long as the object carousel is mounted, regardless of AIT signalling.

This means that if the `carousel_id` descriptor identifying the mounted carousel is removed from a service then all MHEG applications that originated from the mounted DSM-CC carousel shall be terminated or removed from the application stack, and the boot process shall be restarted.

9.2.2.2 `data_broadcast_id_descriptor`

When AIT signalling is used, the existence or content of the `data_broadcast_id_descriptor` in the PMT shall be ignored for the purpose of life-cycle signalling.

More specifically, the following restrictions shall apply:

- Removal of the `data_broadcast_id_descriptor` from the PMT shall not cause any application to be terminated or removed from the application stack, or the boot process to be restarted.
- The network boot info sub-descriptor shall be ignored.
- The `NetworkBootInfo EngineEvent` shall not be generated.
- The `GetBootInfo ResidentProgram` shall always return `infoResult = false` and `bootInfo = ""` (empty string).
- Any change to `NB_version` or `NB_action` values shall not cause any application to be terminated or removed from the application stack, or the boot process to be restarted.
- The service boot info sub-descriptor shall be ignored. Furthermore, any `DsmccDescriptorList` section with `table_id_extension` of `0xffff` shall be ignored.

9.2.2.3 MHEG Non-destructive tune support with AIT signalling

9.2.2.3.0 AIT signalling

Application lifecycle signalling may be PMT and Service Gateway or AIT as defined by clause 11.10.8.4. When performing a non-destructive tune the target service shall use the same signalling method as the source service.

When AIT signalling mode is used the target service shall include:

- An application_signalling_descriptor in the elementary stream loop of the PMT identifying the component carrying the AIT with MHEG application type as defined by table 9.1.
- An AIT containing:
 - An application_identifier in the application loop with an organization_id and application_id matching that of the running application.
 - A transport_protocol_descriptor in either the common or application loops with a protocol_id of 0x0001 (Object Carousel) and with selector bytes referencing the component carrying the "appropriate carousel id" as specified by clause 11.10.8.4.
- A carousel_id_descriptor in the elementary stream loop of the PMT identifying the component carrying the Object Carousel DSI and which contains the "appropriate carousel id".

If the carousel id specified by the non-destructive tune is explicit or current, as defined by clause 11.10.8.4, then the appropriate carousel id in the target service signalling shall match this value. If the carousel is specified as initial then the carousel id in the target service is derived from the AIT.

If the appropriate signalling is not present on the target service then the receiver shall terminate the running application and attempt to launch the auto-boot application on the new service as in clause 9.2.

The Terminal shall interpret the AIT signalling on the target service as specified in the present document.

This means that where the control code is KILL the application shall be terminated.

9.2.2.3.1 Network and service boot info

Following a non-destructive tune, the receiver shall re-evaluate service and network level lifecycle signalling. If application signalling is AIT based as defined by clause 9.2 then network boot info and service boot info are not supported and shall not be evaluated.

9.2.2.4 Definition of "well formed" for MHEG Applications

For the purposes of launching from another Presentation Technology via AIT, an MHEG Application is considered well formed if the IOR of the Initial Object has been extracted from the relevant parent Directory (or ServiceGateway) object.

9.3 PMT and ServiceGateway signalling extension

9.3.1 Introduction

This clause covers the identification and boot of an application by a receiver. The term "application" in this clause describes the use of the broadcast stream. Therefore, the application defined in the present document shall 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 the present document.
- 2) Acquisition of the ServiceGateway object.

- 3) Acquisition of the auto-boot object - in the present document an MHEG-5 Application object. This shall be identified explicitly via signalling in the ServiceContextList (see clause 9.3.4.1) of the ServiceGateway object or implicitly by applying the default identification rules.

9.3.2 Identification of auto-boot application

9.3.2.0 Identifying the boot-PID

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` (see clause 9.3.2.1) 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 the present document the use of the `data_broadcast_id_descriptor` (see clause 9.3.2.1) for signalling is mandatory.

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` value of 0x0106, 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 a `data_broadcast_id_descriptor` with this value of `data_broadcast_id` for each elementary stream within a PMT. This single descriptor is capable of identifying multiple application types within that elementary stream if necessary. Note that this does not preclude there being other `data_broadcast_id_descriptor`s in this elementary stream descriptor loop that contain different values of `data_broadcast_id`.

The generic form of the `data_broadcast_id_descriptor` when used with the `data_broadcast_id` value of 0x0106 is shown in table 9.2. 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 9.2: Data broadcast ID descriptor

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

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

data_broadcast_id: The value 0x0106 in this 16-bit field identifies descriptors with this structure and semantics.

application_type_code: The value of this field identifies that an auto-boot application of this application type is carried by this data broadcast. The value is valid within the scope of this `data_broadcast_id`.

A receiver compliant with the present document shall respond to the values specified in clause B.1 with the exception of the null application type code (0x0000).

In the profile defined by the present document there shall be at most one instance of any of these particular `application_type_codes` within the PMT.

If more than one of these `application_type_codes` are 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".

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 choose between them. The hint is provided by the broadcaster, with 0 the lowest preference. No two application types shall have the same boot priority.

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

application_specific_data_byte: In the present document 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 9.3.

Table 9.3: Network boot info sub-descriptor

Syntax	Size (bytes)	Value
<code>network_boot{</code>		
tag	1	0x01 (Local to ETSIProfile1)
length	1	(N+2)
NB_version	1	
NB_action	1	0x00 - No action 0x01 - Initiate autoboot for DSM-CC mounted applications, generate "NetworkBootInfo" (see table 11.10) EngineEvent for CI mounted applications 0x02 - Generate "NetworkBootInfo" (see table 11.10) EngineEvent
NB_info	N	Allocated by Broadcaster
<code>}</code>		

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 clauses 11.8 and 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 9.4).

Table 9.4: Service boot info sub-descriptor

Syntax	Size (bytes)	Value
<code>service_boot{</code>		
tag	1	0x02 (Local to ETSIProfile1)
length	1	2
assocTag	2	+
<code>}</code>		

The value of `assocTag` indicates the association tag (and thus component) which the receiver shall 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 shall be broadcast on the same component.

9.3.3 Acquisition of the ServiceGateway object

9.3.3.0 Approach to acquisition of the ServiceGateway object

For a receiver supporting the profile defined in the present document, 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 shall be acquired before any other object can be downloaded. The presence of an object carousel is indicated by the carousel_id_descriptor (see clause 9.3.3.1). 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 the present document the use of the carousel_id_descriptor (see clause 9.3.3.1) for signalling is mandatory. The consequence is that if a PMT second descriptor loop contains a data_broadcast_id_descriptor (see clause 9.3.2.1) that provides signalling for the profile defined by the present document, it shall also contain a carousel_id_descriptor (see clause 9.3.3.1).

NOTE: A single PID will only contain messages from a single object carousel and so only one carousel_id_descriptor (see clause 9.3.3.1) will be present in any second descriptor loop. However, a single service may contain more than one object carousel. Consequently, the carousel_id_descriptor (see clause 9.3.3.1) may appear more than once in any single PMT.

9.3.3.1 carousel_id_descriptor

This descriptor is MPEG defined and in the present document may be included in the second descriptor loop of a PMT. See table 9.5.

Table 9.5: 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		
}			
}			

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

FormatID: The use of this field and following data bytes is defined in ETSI TR 101 202 [i.4]. It shall not be used in the present document. Receivers compliant with the present document may ignore this field and any subsequent private data.

9.3.4 Acquisition of the auto-boot object

9.3.4.0 Approach to 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 (see clause 9.3.4.1) 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 may be used to provide a potential location (in addition to the data_broadcast_id_descriptor (see clause 9.3.2.1) in the PMT) for the inclusion of application specific data.

The ServiceContextList contains a loop over "ServiceID". In the present document the 4-byte ServiceID is generated by the combination of data_broadcast_id and application_type_code (see clause 9.3.2.1). The inclusion of an entry in this list is 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 (see clause 9.3.2.1), 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 9.6. Receivers shall be able to parse the structure regardless of the value(s) of application_type_code (see clause 9.3.2.1) and shall be able to skip the application_specific_data_byte (see clause 9.3.2.1) values if they do not recognize the application_type_code (see clause 9.3.2.1).

Table 9.6: Service context list

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

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

ServiceID: The value in this 32-bit field is the concatenation of data_broadcast_id and application_type_code (see clause 9.3.2.1) as shown in table 9.6.

A receiver compliant with the present document 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).

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

application_specific_data_byte: In the present document these bytes may contain a set of language specific initial objects (as defined in table 9.7).

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 9.7.

Table 9.7: Application boot service context list

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

number_languages: Specifies the number of different languages for the application type.

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.

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

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

initial_object_byte: In the present document 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 (see clause 9.3.2.1).

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 (see clause 9.3.4.1) 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 (see clause 9.3.4.1) or the initial Application object identified cannot be acquired from the object carousel, then the receiver shall use the default names "a" and "startup", in that order, to locate the initial Application object in the ServiceGateway object.

If the receiver does not implement InteractionChannelExtension these are invariably references to the Object Carousel and so are equivalent to the following references:

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

9.3.4.2.3 Initial File System

If the receiver implements InteractionChannelExtension then the initial object shall be launched via the hybrid file system such that "hybrid:" becomes the "Current Source" and all relative file references shall be made through it.

When the auto-boot process begins, the hybrid mapping table is reset to contain only the default mapping, so that "hybrid:/" is equivalent to "DSM:/" (see clause 15.12).

9.3.4.2.4 Example

For the initial_object bytes "my_app", the receiver shall work down the list until it is able to locate a file or reaches the end of the list:

- "<source>://my_app".
- "<source>://my_app/a".
- "<source>://my_app/startup".
- "<source>://a".
- "<source>://startup".

If the receiver does not implement InteractionChannelExtension then "<source>" shall explicitly be "DSM:". If the receiver does implement InteractionChannelExtension then "<source>" shall be "hybrid:".

9.3.5 Example of steps required for auto-boot

- 1) "Identification of auto-boot application" (see clause 9.3.2):
 - 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 (see clause 9.3.2.1) containing an appropriate application_type_code (see clause 9.3.2.1) 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" (see clause 9.3.3):
 - a) Search through the second descriptor loop corresponding to the boot-PID for a carousel_id_descriptor (see clause 9.3.3.1).
 - 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" (see clause 9.3.4):
 - a) Get the name of the initial File object (in this case containing an MHEG-5 Application object) using the ServiceContextList (see clause 9.3.4.1) 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 2c) to 2e) in a more generic fashion as necessary).
 - 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.

For receivers that implement InteractionChannelExtension step 1a) shall be preceded by initializing the mapping table for the hybrid file system so that it contains only the default mapping ("/" maps to "DSM://"). Additionally, after step 3e) the "Current Source" shall be initialized to "hybrid:" before the file that was retrieved is launched.

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 9.8 shall be supported in the present document. Note that it is not necessary 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 9.8: 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 shall be 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

9.3.7.0 PMT monitoring

To provide network level signalling all platforms shall monitor changes to the current service PMT.

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 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 14.10.3.2).

9.3.7.2 network_boot_info

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.

A CI mounted application shall receive a NetworkBootInfo (see table 11.10) 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 clauses 9.3.2.1 and 11.10.15.1).

9.3.7.3 data_broadcast_id

If a data_broadcast_id descriptor is removed from a service then all applications that originated from the mounted DSMCC 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

If the carousel_id descriptor identifying the mounted carousel 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.

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 should be 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 should 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.

9.3.7.6 Removal of service

Time-exclusive services may be achieved by altering SI/PSI signalling in accordance with ETSI TR 101 211 [i.3].

Table 3 of ETSI TR 101 211 [i.3] defines the set of states that a service may be in. If a service enters any non-transition state other than "service is running and broadcasting" 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 carousel shall then be unmounted.

Action when the service is in a transition state as defined by ETSI TR 101 211 [i.3] is not defined by the present document.

10 Security

Applications conforming to the present document need not support 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 clauses 14.6 and 14.8. Any risk that might lead to the need to securely manage the control of this storage resource is therefore mitigated by these factors.

11 MHEG-5 engine profile

11.0 Introduction

This clause provides the detailed profile of the MHEG-5 engine that shall be supported 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification 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 Recommendation ITU-T X.680 [6]) defined in annex A of the ISO/IEC 13522-5 [14] (MHEG-5) specification 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 Recommendation ITU-T X.690 [7].

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

- The engine shall ignore each element of a Group's Items sequence that it does not recognize, as well as valid MHEG-5 objects that are not supported by the present document.
- 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.

In all such cases, there shall be no effect on the rest of the Application or Scene, which shall be interpreted normally.

11.3 Set of classes

Table 11.1 identifies the classes that a receiver implementing the ETSIEngineProfile1 shall implement.

Table 11.1: Classes supported by the present document

Class	Abstract/concrete (see note 1)	Required (Y/N) (see 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	
Palette	C	N	
Font	C	Y/N (see note 4)	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 clauses 11.5.2 and 12.8.
LineArt	C	N	See clause 12.5 (see 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 clauses 11.4.1.4 and 11.5.3.
Video	C	Y	
RTGraphics	C	N	
Interactable	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	

Class	Abstract/concrete (see note 1)	Required (Y/N) (see note 2)	Notes
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 need not 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 the present document shall include LineArt objects. No content encoding is defined for this class in the present document. However, engine implementations may include the LineArt class effectively as an abstract class supporting instantiable classes based on LineArt such as Rectangle and DynamicLineArt.			
NOTE 4: Receivers supporting DownloadableFontExtension shall support the Font class. Receivers not supporting DownloadableFontExtension shall not support the Font class.			

11.4 Set of features

11.4.0 MHEG-5 optional 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 11.2 identifies the ETSIEngineProfile1 requirement with regard to features defined as "optional" within the ISO/IEC 13522-5 [14] (MHEG-5).

Table 11.2: Requirements for MHEG-5 "optional" features

Feature	Required (Y/N) (see note)	Notes
Caching	Y	See clause 15.4.
Ancillary connections	N	Need not be implemented.
Cloning	Y	The receiver shall support cloning at least of the following classes: Text, Bitmap and Rectangle.
Free-moving cursor	N	Need not be implemented.
Bitmap scaling	N	Limited scaling shall be supported 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 8.2.
Trick mode	N	Need not be implemented. Not applicable for any currently defined stream-items' content type. See clause 14.4.3.
NOTE: "Y" = yes, i.e. receivers implementing ETSIEngineProfile1 shall support these features.		

11.4.1 GetEngineSupport "feature" strings

11.4.1.0 Set of mandatory GetEngineSupport "feature" strings

Table 11.3 identifies the GetEngineSupport feature strings that receivers implementing ETSIEngineProfile1 shall support.

Table 11.3: ETSIEngineProfile1 GetEngineSupport behaviour

Standard	String		Constraint
	Standard	Short	
AncillaryConnections		ACo	Shall return "false".
ApplicationStacking		ASt	Shall return "true". See clause 8.2.
Cloning		Clo	Shall return "true".
FreeMovingCursor		FMC	Shall return "false".
MultipleAudioStreams(N)		MAS(N)	Shall return "true" for $N \leq 1$. See clause 11.4.1.4.
MultipleVideoStreams(N)		MVS(N)	Shall return "true" for $N \leq 1$.
OverlappingVisibles(N)		OvV(N)	Shall return "true" for all values of N. See clauses 14.4.2 and 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).

String		Constraint								
Standard	Short									
SceneCoordinateSystem(X,Y)	SCS(X,Y)	Shall return "true" for (X,Y) is (720,576). See clause 12.1.								
TrickModes	TrM	Shall return "false".								
VideoScaling(CHook,X,Y) (see note 1)	VSc(CHook,X,Y) (see note 1)	Shall return "true" for the combinations of CHook, X & Y that are supported by the implementation. The minimum set of combinations that shall be supported is tabulated below. See "MPEG presentation". See clause 14.4.4. <table border="1"> <thead> <tr> <th>Parameter</th> <th>Values</th> </tr> </thead> <tbody> <tr> <td>CHook</td> <td>10, 15</td> </tr> <tr> <td>X</td> <td>1440, 1 080, 720, 540, 360</td> </tr> <tr> <td>Y</td> <td>1152, 576, 288</td> </tr> </tbody> </table>	Parameter	Values	CHook	10, 15	X	1440, 1 080, 720, 540, 360	Y	1152, 576, 288
Parameter	Values									
CHook	10, 15									
X	1440, 1 080, 720, 540, 360									
Y	1152, 576, 288									
BitmapScaling(CHook,X,Y) (see note 2)	BSc(CHook,X,Y) (see note 2)	Shall return "true" for the combinations of CHook, X & Y that are supported by the implementation. The minimum set of combinations that shall be supported is tabulated below. See clause 14.4.4. <table border="1"> <thead> <tr> <th>Parameter</th> <th>Values</th> </tr> </thead> <tbody> <tr> <td>CHook</td> <td>2, 7</td> </tr> <tr> <td>X</td> <td>1440, 1 080, 720, 540, 360</td> </tr> <tr> <td>Y</td> <td>1152, 576, 288</td> </tr> </tbody> </table>	Parameter	Values	CHook	2, 7	X	1440, 1 080, 720, 540, 360	Y	1152, 576, 288
Parameter	Values									
CHook	2, 7									
X	1440, 1 080, 720, 540, 360									
Y	1152, 576, 288									
VideoDecodeOffset(CHook,Level) (see note 1)	VDO(CHook,Level) (see note 1)	Shall return "true" for the combinations of CHook and Level tabulated below that are supported by the implementation. See table 11.4.								
BitmapDecodeOffset(CHook,Level) (see note 2)	BDO(CHook,Level) (see note 2)	Shall return "true" for the combinations of CHook and Level tabulated below that are supported by the implementation. See table 11.5.								
UniversalEngineProfile(N)	UEP(N)	Receivers fully conformant with the present document shall return "true" when N corresponds to the values defined in clause B.2. 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. See also clause 17.21.1.								
ICProfile(N)	ICP(N)	Shall return "true" for N=0 if the receiver supports the InteractionChannelExtension. Shall return "true" for N=1 if the receiver supports ICStreamingExtension but may not implement the stream buffering as described in clause 14.2.9. Shall return "true" for N=2 if the receiver supports ICStreamingExtension including the buffer restrictions as described in clause 14.2.9. Shall return "true" for N=3 if the receiver supports ICStreamingExtension as described in clause 14.2.9 (see note 5). Shall return "true" for N=4 if the receiver supports ICEncryptedStreamExtension (see note 6). Shall return "true" for N=5 if the receiver supports ICStreamingExtension including bit rates up to and including 4 096 Kbps and H.264 HD AVC video as described in clause 11.5.4. Shall return "false" for all other values of N.								

String		Constraint
Standard	Short	
DownloadableFont(CHook)	DLF(CHook)	<p>Shall return "true" for the values of CHook that are supported by the Font class.</p> <p>Shall return "false" for all other values of CHook.</p> <p>See clause 13.3.1.</p>
NonLinearPlaybackKeys	NLK	<p>Shall return "true" if the receiver supports ICStreamingExtension and supports all of the following keys as defined in clause 11.6.3:</p> <ul style="list-style-type: none"> • Stop key. • Play and Pause keys, or a combined Play/Pause key. • Either Skip Forward or Fast Forward or both. • Either Skip Backwards or Rewind or both.
HDEExtension(N)	HDE(N)	<p>Shall return "true" for N=0 if the receiver supports HDVideoExtension as described in clause 12.11.</p> <p>Shall return "true" for N=1 if the receiver supports the HDGraphicsPlaneExtension as described in clause 12.11.1.1.</p> <p>See note 4.</p> <p>Shall return "false" for all other values of N.</p>
BitmapFormat(CHook)	BFo(CHook)	<p>Shall return "true" where CHook is one of the defined values of Bitmap content hook listed in table 11.7 and supported by the receiver.</p> <p>This string is mandatory if the receiver supports HDVideoExtension (see clause 12.11) or HDGraphicsPlaneExtension (see clause 12.11.1.1) otherwise optional.</p>
PVREExtension(N)	PVR(N)	<p>Shall return "true" for N=0 if the receiver supports PVREExtension.</p> <p>Shall return "false" for all other values of N.</p>
SIPackageSupport (see note 3)	SPS	<p>Shall return "true" if the receiver implements the Service Information extension. See clause 11.14.</p>
<p>NOTE 1: This CHook is that for a Stream. See table 11.7.</p> <p>NOTE 2: These CHooks are those for Bitmaps See table 11.7.</p> <p>NOTE 3: An application using the Service Information extension shall check for the availability of the extension using this GetEngineSupport string. The naming of this engine support string is historical.</p> <p>NOTE 4: Receivers supporting both the HDVideoExtension and HDGraphicsPlaneExtension should respond "true" to tests for HDE(0) and for HDE(1).</p> <p>NOTE 5: A receiver which returns "true" to ICP(3) shall also return "true" to ICP(2) and ICP(1).</p> <p>NOTE 6: A receiver which returns "true" to ICP(4) does not mean that ICP(2) or ICP(3) is supported.</p>		

Unrecognized requests shall always result in GetEngineSupport returning false. This implies that future MHEG-5 engine implementations supporting more than these minimum features shall recognize additional strings.

Receivers shall truthfully reflect their capability when interrogated with GetEngineSupport.

11.4.1.1 VideoDecodeOffset

Table 11.4 shows the parameters for VideoDecodeOffset.

Table 11.4: VideoDecodeOffset parameters

CHook	Level	Description
10, 15 (see table 11.7)	0	The receiver can present Video for which the Position and VideoDecodeOffset attributes are 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).
	1	The Position and VideoDecodeOffset attribute for Video can be such that the decoded and scaled video can be partially off screen. At least half of the decoded image height shall remain within the display raster. For horizontal scaling factors of full size and above, at least one quarter of the decoded image width shall remain within the display raster. For horizontal scaling factors below full size, at least half the decoded image width shall remain within the display raster.

See the examples under clause 11.12.10.2.

11.4.1.2 BitmapDecodeOffset

Table 11.5 shows the parameters for BitmapDecodeOffset.

Table 11.5: BitmapDecodeOffset parameters

CHook	Level	Description
2, 7 (see table 11.7)	0	The receiver can present I-frame Bitmaps for which the Position and BitmapDecodeOffset attributes are 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).
	1	The Position and BitmapDecodeOffset attributes for I-frame Bitmaps can be such that the decoded and scaled bitmap can be partially off screen. At least half of the decoded image height shall remain within the display raster. For horizontal scaling factors of full size and above, at least one quarter of the decoded image width shall remain within the display raster. For horizontal scaling factors below full size, at least half the decoded image width shall remain within the display raster.

See clause 11.12.7 and the examples under clause 11.12.10.2.

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 shall be 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) or is delivered via the Interaction Channel.
- 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) (see table 11.3) for $N > 1$ has not been defined.

11.5 Content data encoding

11.5.0 Coding attributes and hook values

Table 11.6 identifies the minimum set of coding of attributes that shall be supported by the engine.

Table 11.6: 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 13.3.
TransitionEffect	Shall not be implemented.

Table 11.7 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 11.7: Encoding table

Type of content	Specification (data types)	Hook values
Font	See clause 13.3.1.	10 (see note)
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
	Reserved.	5
	JPEG bitmap. See clause 12.12. Support for JPEG bitmaps is mandatory for a receiver that supports HDGraphicsPlaneExtension, otherwise optional.	6
	H.264 Intra Frame. See clause 12.13. Support for H.264 Intra Frames is mandatory for a receiver that supports HDVideoExtension, otherwise optional.	7
Reserved.	8	
Text	See clause 13.6.	10
EntryField	See clause 13.2.	10
	See clause 13.7.	

Type of content	Specification (data types)	Hook values
HyperText	See clause 13.8.	10
Stream	A broadcast MPEG program with construction and components that conform to the present document.	10
	A "file" containing a single elementary stream that can be decoded from memory. Data formats are described in clause 11.5.3. In the present document Audio is the only stream component that can be decoded from memory.	11
	An MPEG-2 Single Program Transport Stream, containing a combination of audio, video and DVB subtitle components, restricted as described in clause 11.5.4, and which may be accessed in a non-linear method, see clause 14.4.3. In the present document "http:" or "https:" are the only sources that shall be supported for this content hook. This content hook shall be supported for receivers that implement ICStreamingExtension.	15
LineArt	(None specified).	
CursorShape	(None specified).	
InterchangedProgram	(None specified).	
NOTE: For the Font class, content hook 10 shall be supported by receivers that implement DownloadableFontExtension. For receivers that do not implement DownloadableFontExtension there is no content hook value defined for the Font class.		

11.5.1 Use of negative hook values

Negative hook values shall be 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-2 I-frame and H.264 Intra frame content. See clause 12.8. Scaling shall not be supported for **Bitmap** objects using PNG [17] or JPEG bitmaps.

11.5.2.2 Tiling

Tiling shall be supported for PNG [17] bitmaps and for JPEG bitmaps, where supported.

11.5.2.3 Transparency

Transparency can be encoded within the PNG [17] bitmaps. Object level transparency (i.e. the **Transparency** internal attribute of the **Bitmap** class) shall not be supported for any bitmap type. See clause 12.4.1.

11.5.3 Stream "memory" formats

11.5.3.0 Scope of StreamComponent applicability

In the present document the only **StreamComponent** that accepts a **Storage** specification of memory shall be **Audio**. **Video** shall only be played from **Stream**.

11.5.3.1 Audio

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

The MPEG Audio data in all other respects conforms to the specifications provided in the present document with the constraint that only MPEG-2 audio data is required to be supported.

11.5.4 Non-linear stream formats

11.5.4.0 Profile for IP-delivered Transport Stream

Receivers that implement ICStreamingExtension shall be able to present a Transport Stream delivered by IP at bit rates up to and including 2 048 kbps. Receivers returning true for ICP (5) shall be able to present a Transport Stream delivered by IP at bit rates up to and including 4 096 kbps and H.264 HD AVC. Higher bit rates may be supported.

The audio, video and subtitle components will be encapsulated in a single program MPEG-2 Transport Stream. The transport stream will contain minimal PSI necessary to enable correct interpretation of the stream:

- PAT. There will be only one programme identified in the PAT.
- PMT. This will identify the video, audio and subtitle components of the programme, where present.

SI/PSI tables shall remain unchanged throughout the Transport Stream.

No additional SI/PSI tables can be guaranteed to be included; if they are then they should be ignored.

In the present document Streams with a content hook of 15 shall be subject to the following restrictions.

11.5.4.1 Video

Where the Transport Stream includes a video component it shall be encoded as H.264 AVC according to ISO/IEC 14496-10 [39], constrained according to ETSI TS 101 154 [40]. Receivers shall support H.264 SD AVC according to ISO/IEC 14496-10 [39], constrained according to ETSI TS 101 154 [40]. Receivers returning true for ICP(5) shall also support H.264 HD AVC according to ISO/IEC 14496-10 [39], constrained according to ETSI TS 101 154 [40]. Only the 25 Hz AVC variants described in clause 5 (video) of ETSI TS 101 154 [40] are relevant to the present document.

11.5.4.2 Audio

Where the Transport Stream includes audio components they shall be encoded as either:

- MPEG-4 High Efficiency AAC, according to ISO/IEC 14496-3 [42] (up to HE-AAC Level 4 excluding the use of the Parametric Stereo tool), constrained according to ETSI TS 101 154 [40]; or
- Dolby E-AC3 encoding constrained according to ETSI TS 102 366 [41].

Receivers shall support MPEG-4 High Efficiency AAC. Receivers returning true for ICP(5) shall also support Dolby E-AC3. Multi-channel audio need not be supported.

11.5.4.3 Subtitles

No additional restrictions.

11.5.4.4 Encrypted non-linear streams

Receivers that implement ICEncryptedStreamExtension shall be able to decrypt non-linear streams delivered by IP.

When the content payload of Transport Stream packets is encrypted it shall use the AES-128 encryption algorithm defined in FIPS PUB 197 [50]. This is used in the variant of Cipher-Block-Chaining (CBC) mode as defined in ANSI_SCTE 52 [51].

NOTE: SCTE52 was originally defined for use with DES, which has a 64-bit block size. However the extension of the principle to the 128-bit block size of AES is obvious.

11.6 User input

11.6.1 Base remote control functions

11.6.1.0 Overview of remote control function groups

Figure 11.1 illustrates the base 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 base functions may be quite different from the one illustrated.

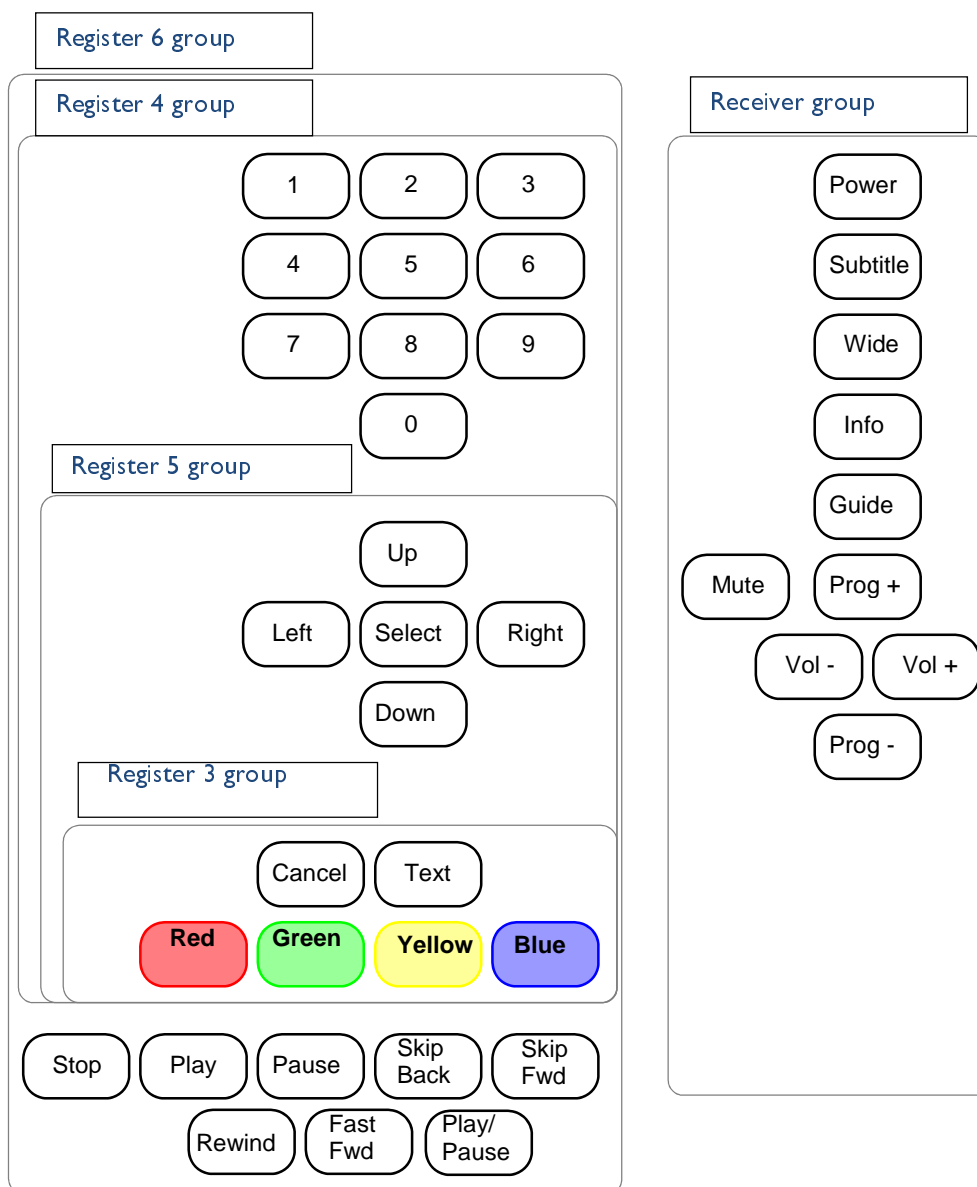


Figure 11.1: Base remote control function group

11.6.1.1 Receiver group

The input provided by this group shall not be 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" shall always have the effect of changing channel.

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

11.6.1.2 Register 3 group (see table 11.8)

The input provided by this group shall always be 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 14.9). This group shall not be used for receiver "channel surfing".

11.6.1.3 Register 4 group (see table 11.8)

The input provided by this group can be available to a running MHEG-5 application.

MHEG-5 applications shall 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 (see table 11.8)

As Register 4 group (see clause 11.6.1.3).

11.6.1.5 Register 6 group (see table 11.8)

As Register 4 group (see clause 11.6.1.3). Receivers that support ICStreamingExtension shall support this register group. However, only those keys that exist on the receiver's remote control shall actually generate events. If no DVR keys are present, this input register is equivalent to register 4. Receivers shall indicate the availability of the optional keys to the MHEG application through the "NonLinearPlaybackKeys" engine feature string (see clause 11.4.1).

Receivers may implement either separate Play and Pause keys or a combined Play/Pause key.

11.6.2 Extended remote control functions

11.6.2.0 Outline of extended function groups

Figure 11.1 also illustrates the extended functions of the register 6 group. Register 13, 14 and 15 groups are not illustrated.

11.6.2.1 Register 6 group (see table 11.9)

As Register 4 group. Receivers that support ICStreamingExtension shall support this register group. However, only those keys that exist on the receiver's remote control shall actually generate events. If no DVR keys are present, this input register shall be equivalent to register 4. Receivers shall indicate the availability of the optional keys to the MHEG application through the "NonLinearPlaybackKeys" engine feature string (see clause 11.4.1).

Receivers shall implement either separate Play and Pause keys or a combined Play/Pause key.

11.6.3 UserInput registers

11.6.3.1 Base UserInput registers

Receivers shall implement input event registers 3, 4 and 5 defined in table 11.8. The DAVIC defined registers 1 and 2 are shown for information only.

Table 11.8: Base InputEvent Registers

UserInput EventData	Function name	Register number				
		1	2	3	4	5
1	Up	✓	✓		✓	✓
2	Down	✓	✓		✓	✓
3	Left	✓	✓		✓	✓
4	Right	✓	✓		✓	✓
5 to 14	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 respectively	✓			✓	
15	Select	✓	✓		✓	✓
16	Cancel	✓	✓	✓	✓	✓

UserInput EventData	Function name	Register number				
		1	2	3	4	5
17	Help	✓	✓			
18 to 99	Reserved by DAVIC					
100	Red			✓	✓	✓
101	Green			✓	✓	✓
102	Yellow			✓	✓	✓
103	Blue			✓	✓	✓
104	Text			✓	✓	✓
105 to 119	Reserved for future specification					
120	Stop					
121	Play					
122	Pause					
123	Skip Forward					
124	Skip Back					
125	Fast Forward					
126	Rewind					
127	Play/Pause					
128 to 256	Reserved for future specification					
257 to 299	Vendor specific					
300	EPG					
≥ 301	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.						

11.6.3.2 Extended UserInput registers

UserInput registers for receivers that support extensions that implement additional optional functionality and keys are defined in table 11.9. Keys marked as 'O' are optional. Where they are supported by the receiver the values defined shall be generated.

Table 11.9: Extended InputEvent Registers

UserInput EventData	Function name	Register number			
		6 (see note 1)	13 (see note 2)	14 (see note 2)	15 (see note 2)
1	Up	✓		✓	✓
2	Down	✓		✓	✓
3	Left	✓		✓	✓
4	Right	✓		✓	✓
5 to 14	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 respectively	✓		✓	
15	Select	✓		✓	✓
16	Cancel	✓	✓	✓	✓
17	Help				
18 to 99	Reserved by DAVIC				
100	Red	✓	✓	✓	✓
101	Green	✓	✓	✓	✓
102	Yellow	✓	✓	✓	✓
103	Blue	✓	✓	✓	✓
104	Text	✓	✓	✓	✓
105 to 119	Reserved for future specification				
120	Stop	O			
121	Play	O			
122	Pause	O			
123	Skip Forward	O			
124	Skip Back	O			
125	Fast Forward	O			
126	Rewind	O			
127	Play/Pause	O			
128 to 256	Reserved for future specification				
257 to 299	Vendor specific				

UserInput EventData	Function name	Register number			
		6 (see note 1)	13 (see note 2)	14 (see note 2)	15 (see note 2)
300	EPG		✓	✓	✓
≥ 301	Vendor specific				
NOTE 1: Input register 6 shall only be supported by receivers that implement ICStreamingExtension. NOTE 2: All three input registers (13, 14 and 15) shall be supported by receivers that implement the EPG key.					

11.6.4 Implementation of this interaction model

The receiver shall be 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 should be addressed through authoring guidelines (see clause 17.2).

11.6.5 Interaction with broadcast-triggered native applications

For receivers supporting NativeApplicationExtension, it shall be possible for keys in the Register 3 group (i.e. Cancel, Text, Red, Green, Yellow, Blue) to be available to a broadcast-triggered native application instead of an MHEG-5 application, irrespective of the input event register, even when clause 11.6.1 would otherwise grant exclusive use of those keys to an MHEG-5 application.

The currently defined native applications require access to the Green key.

11.7 Semantic constraints on MHEG-5 applications

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

11.8 EngineEvents

11.8.0 List of required engine events

Table 11.10 lists the minimum set of engine events that the engine shall support.

Table 11.10: Required engine events

EventData		Notes
Name	Value	
	< 0	Manufacturer specific.
Reserved	0	Reserved.
	1	
GroupIDRefError	2	This event shall be raised if a Launch, Spawn or TransitionTo elementary action attempts to access a file that cannot be provided by the file system as a file. See clause 11.8.1.
ContentRefError	3	This event shall be raised if an application attempts to access referenced content that cannot be provided by the file system. See clause 11.8.1.
TextKeyFunction	4	Generated when the user activates the "Text" function and there is an active scene object. This event shall be raised independently of the InputEvent register selected at the current moment or whether any interactable has the InteractionStatus of True.
		If "Text" function causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first. See clause 11.6.1.
Reserved	5	Reserved.

EventData		Notes
Name	Value	
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. Note that this event shall not be 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 14.9.
SubtitlePrefChanged	8	Generated when the user preferences for the presentation of DVB Subtitles changes. This event is raised when the condition of "Control E3" as illustrated in figure 14.4, clause 14.3.3 changes.
NetworkBootInfo	9	Generated in response to changes in the relevant data_broadcast_id_descriptor (see clause 9.3.2.1) when: <ul style="list-style-type: none"> the network_boot_sub-descriptor either appears or changes (as indicated by the NB_version), and the NB_action is set to 0x02; 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. The application can access the value of the NB_info field at any time via the resident program GetBootInfo (see clause 11.10.15.1).
NonDestructiveTuneOK	10	If the receiver implements LifecycleExtension, generated when the receiver successfully attaches to the requested carousel in a target service after a non-destructive tune (see clauses 11.10.8.4 and 8.1.7). If the receiver does not implement LifecycleExtension, reserved.
Reserved	11 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 shall be raised independently of the InputEvent register selected at the current moment or whether any interactible has the InteractionStatus of True. If a key press causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first. See clause 11.6.1.
Reserved	17 to 19	Reserved.
PVRChangedEvent	20	For receivers that implement the PVRExtension, generated when the list of events to be recorded by the PVR changes. Possible reasons for changes include but are not limited to: <ul style="list-style-type: none"> event recorded; event cancelled due to conflict; event added or removed through the PMB or PCB ResidentPrograms.
Reserved	21 to 99	Reserved.
RedKeyFunction	100	Generated when the user activates the appropriate colour key and there is an active scene object. This event shall be raised independently of the InputEvent register selected at the current moment or whether any interactible has the InteractionStatus of True.
GreenKeyFunction	101	
YellowKeyFunction	102	
BlueKeyFunction	103	If a key press causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first. See clause 11.6.1.
Reserved	104 to 199	Reserved.
The following engine events shall be implemented only in receivers that implement InteractionChannelExtension. Otherwise these values are reserved.		
ICStatusChanged	200	Generated when the state of the receiver's interaction channel connection changes in a way that would alter the return value of the GetICStatus resident program. (See clause 11.10.12.1.)

EventData		Notes
Name	Value	
ICLocalError	201	Generated if an application attempts to access referenced content, an application object or a scene object that cannot be provided by the file system because no interaction channel connection is available.
ICNetwork Error	202	Generated if an application attempts to access referenced content an application object or a scene object that cannot be provided by the file system because a remote server did not respond.
ICRemoteError	203	Generated if an application attempts to access referenced content, an application object or a scene object that cannot be provided by the file system because a remote server indicates that the requested file is unavailable.
The following engine events shall be implemented only in receivers that implement ICStreamingExtension. Otherwise these values are reserved.		
StreamRefError	204	This event is raised if an application fails to access a media stream. The event shall not be raised for broadcast services.
StreamUnderflow	205	This event is raised when a media stream stops being presented due to buffer underflow. This event shall not be raised when playback reaches the end of the stream. The event shall not be raised for broadcast services.
Reserved	206	Reserved.
AudioDescPrefChanged	207	Generated when the viewer preference for the presentation of Audio Description changes.
StreamUnderflowResume	208	This event is raised when a media stream resumes being presented following buffer underflow. The event shall not be raised for broadcast services.
Reserved	209	Reserved
KeyFileError	210	This event is raised when a key file associated with a media stream cannot be obtained, or is not valid.
Reserved	211-299	Reserved.
The following engine event shall be implemented only in receivers that implement the EPG key. Otherwise this value is reserved.		
EPGKeyfunction	300	Generated when the user activates the EPG 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. If a key press causes both the EngineEvent and the UserInput event then the EngineEvent shall be raised first.
Reserved	All other values	Reserved.

11.8.1 Object retrieval errors

The engine event GroupIDRefError (see table 11.10) shall be raised if an Action directly attempts to access an object that cannot be provided by the file system as a file. Similarly, the engine event ContentRefError (see table 11.10) shall be raised if an Action directly attempts to access some content that cannot be provided by the file system as a file, Stream or StreamEvent. The events shall not be generated if the content of the file is otherwise invalid. For clarification a retrieved empty file shall not generate either of these events. Neither event shall be raised when a URL reference to a broadcast service cannot be resolved. See also clause 14.2.8.

The above does not define the behaviour of file access errors that are the consequence of file accesses from resident programs. In the present document the only ResidentPrograms that attempt to access files are CheckContentRef and CheckGroupIDRef (see table 11.12). These shall not raise any engine events (the return parameters from the program include an indication of file availability).

11.8.2 Object retrieval errors - Interaction Channel

This clause applies only to receivers that implement InteractionChannelExtension.

If an Action results in one of ICLocalError, ICNetworkError or ICRemoteError being raised in addition to one of the GroupIDRefError or ContentRefError, the engine event relating to the interaction channel shall be raised first.

11.9 Protocol mapping and external interaction

Table 11.11 shows the protocol mapping and external internal interaction.

Table 11.11: Protocol mapping

MHEG-5 entity	Mapping needed	Semantics
OpenConnection, CloseConnection	Mapping to connection management	Shall not be implemented
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
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 ResidentPrograms

11.10.0 List of ResidentPrograms

Table 11.12 lists the ResidentPrograms that a ETSIEngineProfile1 receiver shall implement.

Table 11.12: ETSIEngineProfile1 mandatory resident programs

ResidentProgram		Invocation			Reference
Description	Name	Typical use		Never Fork	
		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 (see note 1)	CTC	✓			Clause 11.10.6.1.
CastToObjectRef (see note 1)	CTO	✓			Clause 11.10.6.2.
CastToStringInt (see note 1)	CSI	✓			Clause 11.10.6.3.
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	✓		✓ (see 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 11.10.15.1.
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 11.10.14.1.

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

Table 11.13 lists further optional ResidentPrograms.

Table 11.13: Optional resident programs

ResidentProgram		Invocation			Reference
Description	Name	Typical use		Never Fork	
		Call	Fork		
Debug	DBG	✓			Clause 11.10.14.2.

Table 11.14 lists the ResidentPrograms that receivers implementing InteractionChannelExtension shall implement.

Table 11.14: Mandatory Resident Programs for receivers that implement InteractionChannelExtension

ResidentProgram		Invocation			Reference
Description	Name	Typical use		Never Fork	
		Call	Fork		
GetICStatus	GIS	✓			Clause 11.10.12.1.
ReturnData	RDa	✓			Clause 11.10.12.2.
SetHybridFileSystem	SHF	✓		✓	Clause 11.10.13.1.
PersistentStorageInfo	PST	✓			Clause 11.10.12.1.
SetCookie	SCk	✓			Clause 11.10.12.1.
GetCookie	GCK	✓			Clause 11.10.12.1.

Table 11.15 lists the ResidentPrograms that receivers implementing ICStreamingExtension shall implement.

Table 11.15: Mandatory Resident Programs for receivers that implement ICStreamingExtension

ResidentProgram		Invocation			Reference
Description	Name	Typical use		Never Fork	
		Call	Fork		
MeasureStreamPerformance	MSP		✓		Clause 11.10.12.3.
PromptForGuidance	PFG	✓		✓	Clause 11.10.12.4.
GetAudioDescPref	GAP	✓			Clause 11.10.10.6.
GetSubtitlePref	GSP	✓			Clause 11.10.10.7.
GetPINSupport	GPS	✓			Clause 11.10.12.8.

Table 11.16 lists the ResidentPrograms that receivers implementing NativeApplicationExtension shall implement.

Table 11.16: Mandatory Resident Programs for receivers that implement NativeApplicationExtension

ResidentProgram		Invocation			Reference
Description	Name	Typical use		Never Fork	
		Call	Fork		
SetBroadcasterInterruptions	SBI	✓			Clause 11.10.10.5.

11.10.1 Typical use

Table 11.12 through to table 11.16 provide an indication of the method by which the resident program should be 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 17.15.

11.10.2 Program names

The three character "Name" indicated in table 11.12 through to table 11.16 shall be used to invoke the ResidentProgram.

11.10.3 Encoding of resident program names

The names of resident programs shall be conveyed in OctetStrings. The character encoding for names of resident programs uses ISO/IEC 8859-1 [8]. All of the characters used in table 11.12 lie in the code range 0x00 to 0x7E.

The names of resident programs shall be 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.

11.10.4.2 GetCurrentDate

Synopsis: Retrieves the current local date and time.

Arguments: GCD(date, time). See table 11.17.

Table 11.17: 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

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

Arguments: FDa(dateFormat, date, time, datestring). See table 11.18.

Table 11.18: 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	Values in the range 0 to 86399 describe the time encoded as the number of seconds since midnight. The meaning of other values are not defined.
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	dateString	The resultant formatted date and time.

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

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

Synopsis: Returns the day of the week.

Arguments: GDW(date, dayOfWeek). See table 11.19.

Table 11.19: 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

Synopsis: Returns a random number.

Arguments: Rnd(num, random). See table 11.20.

Table 11.20: 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. The returned value is undefined if the num parameter < 1.

11.10.6 Type conversion functions

11.10.6.1 CastToContentRef

Synopsis: Casts an OctetString to a ContentReference.

Arguments: CTC(string, contentRef). See table 11.21.

Table 11.21: CastToContentRef parameters

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

Description: Casts the OctetString variable string to the ContentReference variable contentRef.

11.10.6.2 CastToObjectRef

Synopsis: Casts an OctetString and Object Identifier to an Object Reference.

Arguments: CTO(string, objectId, objectRef). See table 11.22.

Table 11.22: 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	

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).

11.10.6.3 CastToStringInt

Synopsis: Casts an ObjectReference to its GroupIdentifier and ObjectNumber.

Arguments: CSI(objectRef, groupId, objectNumber). See table 11.23.

Table 11.23: CastToStringInt parameters

In/out/in-out	Type	Name	Comment
input	GenericObjectReference	objectRef	Object Reference to be cast.
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	groupId	String to store the groupId.
output	GenericObjectInteger (shall provide an IndirectReference to an IntegerVariable)	objectNumber	Integer to store the object number.

Description: Casts the ObjectReference to an OctetString variable and an Integer variable. The OctetString variable shall contain the object reference's GroupIdentifier, and the Integer variable shall contain the object reference's ObjectNumber. If the ObjectReference does not have an encoded GroupIdentifier then the resident program shall return the default value of GroupIdentifier. The format of the GroupIdentifier returned (i.e. if it uses a fully resolved or shorthand notation) is not defined.

11.10.7 String manipulation functions

11.10.7.0 Use of 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 shall be 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

Synopsis: Returns the number of octets within the string.

Arguments: GSL(string, length). See table 11.24.

Table 11.24: 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	

Description: Returns in the output "length" the number of octets within the input "string".

11.10.7.3 GetSubString

Synopsis: Extracts a sub-string from a string.

Arguments: GSS(string, beginExtract, endExtract, stringResult). See table 11.25.

Table 11.25: 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#"	

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

Synopsis: Searches for a sub-string within a string.

Arguments: SSS(string, startIndex, searchString, stringPosition). See table 11.26.

Table 11.26: 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 <i>string</i> or -1 if not found.

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 shall be 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 shall be returned.

11.10.7.5 SearchAndExtractSubString

Synopsis: Searches and extracts a sub-string within a string.

Arguments: SES(string, startIndex, searchString, stringResult, stringPosition). See table 11.27.

Table 11.27: 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.

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 shall be 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" shall return the string from "startIndex" up to, but not including, "searchString". For example, if "Bar" is sought in "Foo##Bar" then the returned values shall be "Foo##" and 9.

11.10.8 Service selection

11.10.8.0 Use of service selection functions

These resident programs provide a means to select (i.e. tune to) a given service. They also include the SI_GetBasicSI resident program, which assists in dealing with the regional variants of a service. The behaviour of these resident programs shall be unaffected by the use of any defined service_attribute_descriptor.

11.10.8.1 SI_GetServiceIndex

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

Arguments: GSI(serviceReference, serviceIndex). See table 11.28.

Table 11.28: 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 (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 (see clause 11.10.8.2). -1 is returned if the service is not available.

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 16.1) formats.

Service availability shall be 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 does not guarantee that the service is running.

11.10.8.2 SI_TuneIndex

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

Arguments: TIn(serviceIndex). See table 11.29.

Table 11.29: 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.

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

Synopsis: Returns basic SI information about the service identified in the input serviceIndex.

Arguments: BSI(serviceIndex, networkId, origNetworkId, transportStreamId, serviceId). See table 11.30.

Table 11.30: 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	

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 shall be undefined if an invalid serviceIndex is used.

11.10.8.4 SI_TuneIndexInfo

11.10.8.4.0 Usage

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

Arguments: TII(tuneinfo). See table 11.31.

Table 11.31: SI_TuneIndexInfo parameters

In/out/in-out	Type	Name	Comment
input	GenericInteger	tuneinfo	This input parameter specifies how the receiver shall execute any subsequent application initiated service tune.

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 (tuneinfo) 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 as if tuneinfo has been set to 0, resulting in a destructive service tune, executed "normally" (see below).

The 4-byte tuneinfo argument shall be treated as a 32-bit bit-field encoded as follows:

The value of bit 0 (the least significant bit of tuneinfo) represents the tune_quietly_flag, which shall be interpreted as follows:

- 0 (unset) Perform any subsequent application initiated service tune normally.
- 1 (set) Perform any subsequent application initiated service tune quietly.

If the receiver does not support LifecycleExtension, bits 1, 2 and 3 of tuneinfo shall be ignored and the receiver shall behave as if they were unset i.e. indicating a destructive service tune, as described below.

The value of bit 1 of tuneinfo represents the `app_keeprunning_flag`, which shall be interpreted as follows:

- | | |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (unset) | Perform any subsequent application initiated service tune destructively by terminating the current application. |
| 1 (set) | Perform any subsequent application initiated service tune non-destructively by attempting to keep the current application running both during and afterwards. |

The value of bit 2 of tuneinfo represents the `explicit_carousel_id_flag`, which shall be interpreted as follows:

- | | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (unset) | Value of <code>carousel_id</code> to be used in a non-destructive tune not encoded but instead shall be obtained based on the setting of the <code>carousel_id_mapping_flag</code> (see below).

Bits 8-15 of tuneinfo are reserved in this case and shall be ignored. |
| 1 (set) | Value of <code>carousel_id</code> to be used in a non-destructive tune encoded by bits 8 to 15 of tuneinfo, where bit 8 is the lsb of the <code>carousel_id</code> . |

These 8 bits contain the complete encoding of the `carousel_id`, not just the first 8 bits. Carousels that need to be referenced in this way shall have their `carousel_id` set to be in the range 0 to 255. The value of bit 3 of tuneinfo represents the `carousel_id_mapping_flag`, which shall be interpreted as follows:

- | | |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (unset) | Value of <code>carousel_id</code> to be used in a non-destructive tune shall be that for the Current Carousel (see clause 8.1.5). |
| 1 (set) | Value of <code>carousel_id</code> to be used in a non-destructive tune shall effectively be that of the Initial Carousel for the target service to be tuned to (see clause 8.1.5). |

If the receiver does not support `NativeApplicationExtension`, bit 4 of tuneinfo shall be ignored and the receiver shall behave as if it were unset.

The value of bit 4 of tuneinfo represents the `broadcaster_interrupt_flag`, which shall be interpreted as follows:

- | | |
|-----------|----------------------------------------------------------------------------------------------------------|
| 0 (unset) | Broadcaster interruption status flag shall follow the normal rules specified in clause 11.10.17. |
| 1 (set) | Set broadcaster interruption status flag to disabled following the service tune (see clause 11.10.10.5). |

Summary of extensions:

- Bit 0 of tuneinfo shall be interpreted by all receivers.
- Bits 1 to 3 of tuneinfo shall be interpreted by receivers that support `LifecycleExtension` and shall otherwise be ignored.
- Bit 4 of tuneinfo shall be interpreted by receivers that support `NativeApplicationExtension` and shall otherwise be ignored.
- Bits 5 to 7 and all other bits of tuneinfo are reserved and shall be ignored.

11.10.8.4.1 Destructive service tune

If the `app_keeprunning_flag` is unset, or if the receiver does not support `LifecycleExtension`, then any subsequent application initiated service tune shall be executed destructively. This means that the current application will be terminated as part of the service tune. In this scenario:

- If the `tune_quietly_flag` is unset then the receiver shall execute the service tune normally. This means that the viewer experience shall be exactly as if they had initiated a standard service tune through selection of the new service using any of the receiver's inherent service navigation mechanisms, e.g. using the Ch+ or Ch- keys or numeric entry. This shall be reflected in (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.

- If the `tune_quietly_flag` is set then the receiver shall execute the service tune quietly. This means that the receiver shall suppress the presentation of all information, e.g. channel number, service name, and now/next, etc. usually presented during a standard 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 that 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+ or Ch-, shall be performed.
- The `explicit_carousel_id_flag`, `carousel_id_mapping_flag` and any encoded value of `carousel_id` shall be ignored.

11.10.8.4.2 Non-destructive service tune

If the `app_keeprunning_flag` is set and the receiver supports `LifecycleExtension` then any subsequent application initiated service tune shall be attempted non-destructively. This means that the receiver shall attempt to keep the current (calling) application running both during and after the service tune.

The first step of a non-destructive service tune is the "tune" itself. In this scenario:

- If the `tune_quietly_flag` is unset (i.e. a normal tune) then the receiver shall nonetheless execute the transitional part of the service tune as if the `tune_quietly_flag` were set, suppressing the presentation of all information, e.g. channel number, service name, and now/next, etc. usually presented during a standard service tune. However, the receiver state at the end of the service tune shall be exactly as if the viewer had initiated a standard service tune through selection of the new service using any of the receiver's inherent service navigation mechanisms, e.g. using the Ch+ or Ch- keys or numeric entry. This should be reflected in (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+ or Ch-, shall be performed.
- If the `tune_quietly_flag` is set then the receiver shall execute the service tune quietly exactly as defined for the scenario when the `app_keeprunning_flag` is unset, as described previously.

The receiver shall keep the application running during the tune, although some functions, such as the servicing of carousel file requests, will be suspended during this period (see clause 8.1.7).

The second step of a non-destructive tune occurs after the actual "tune" has been successfully completed. This involves re-evaluating the presentation of broadcast streams by any active MHEG-5 Stream objects and attempting to attach to a carousel in the new service. In this scenario:

- If the `explicit_carousel_id_flag` is unset then the receiver shall use the `carousel_id_mapping_flag` to determine the carousel to attempt to attach to.
- If the `carousel_id_mapping_flag` is unset then the receiver shall attempt to attach to a carousel in the new service that has the same `carousel_id` as that of the Current Carousel (in the previous service) at the point that the application initiated the non-destructive tune.
- If the `carousel_id_mapping_flag` is set then the receiver shall attempt to attach to the Initial Carousel for the new service (see clause 8.1.5).
- If the `explicit_carousel_id_flag` is set then the receiver shall attempt to attach to a carousel in the new service that has the value of `carousel_id` identified by bits 8 to 15 of `tuneinfo`, i.e. in the range 0 to 255.
- The `carousel_id_mapping_flag` shall be ignored.

If the identified method for attempting to attach to a carousel in the new service fails, e.g. no carousel is present in the new service with the required value of `carousel_id`, then the receiver shall not attempt to employ an alternative method as a fallback. Instead the receiver shall terminate the running application and attempt to launch the auto-boot application for the new service (see clause 8.1.3).

11.10.9 Checking references

11.10.9.0 Use of reference checking functions

This set of resident programs can be used by applications to determine if objects (from a file system) 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. References to be checked using these Resident Programs may be to any file system supported by the receiver, including any optional file systems that are implemented.

When the object being referenced is to be retrieved from a DSM-CC object carousel then, 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.

If the receiver implements InteractionChannelExtension then when the object being referenced is to be retrieved over HTTP, receivers shall make a HEAD request to discover whether the object is available. Object availability shall be determined by the reception of an HTTP response code indicating success (see clause 15.7.3).

If the receiver implements a cache for the IC file system then depending on the server's response to the HEAD request, the receiver may subsequently make a GET request so that the object can be loaded into the receiver's memory as a side effect of the Resident Program.

The access controls as defined in clause 15.14 shall be considered when determining the availability of a file. A HEAD request shall be validated against the server list file but needs no further authentication. However if a GET request is optionally performed then this shall follow the requirements of the authentication rules. An object referenced in CheckGroupIDRef shall be treated as a file containing code for the purposes of availability. Therefore the availability of an object may be different when requested with CheckContentRef than with CheckGroupIDRef.

An authoring example is provided under clause 17.10, which illustrates the expected use of these resident programs.

11.10.9.1 CheckContentRef

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

Arguments: CCR(ref-to-check, ref-valid-var, ref-checked-var). See table 11.32.

Table 11.32: 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.

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

The ref-valid-var result shall be 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 need not start decoding the content.

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

11.10.9.2 CheckGroupIDRef

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

Arguments: CGR(ref-to-check, ref-valid-var, ref-checked-var). See table 11.33.

Table 11.33: 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.

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

The ref-valid-var result shall be 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 need not start decoding the content.

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

11.10.10 Presentation information

11.10.10.1 VideoToGraphics

Synopsis: 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.

Arguments: VTG(videoX, videoY, graphicsX, graphicsY). See table 11.34.

Table 11.34: 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) shall be 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.

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

$$\begin{bmatrix} \text{graphicsX} \\ \text{graphicsY} \\ 1 \end{bmatrix} = \begin{bmatrix} \text{scaleX} & 0 & (\text{posX} + \text{offsetX}) \\ 0 & \text{scaleY} & (\text{posY} + \text{offsetY}) \\ 0 & 0 & 1 \end{bmatrix} \times [\text{DecFC}] \times \begin{bmatrix} \text{videoX} \\ \text{videoY} \\ 1 \end{bmatrix} \quad (1)$$

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;
- **scaleX** and **scaleY** are the current Video scale factor (e.g. 0.5; 1.0; etc.);
- **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:

$$\text{DecFC}_{\text{none}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{DecFC}_{\text{cco}} = \begin{bmatrix} 16/12 & 0 & -120 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{DecFC}_{\text{pillarbox}} = \begin{bmatrix} 12/16 & 0 & 90 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{DecFC}_{\text{letterbox}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 12/16 & 72 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{DecFC}_{14\text{by}9\text{letterbox}} = \begin{bmatrix} 16/14 & 0 & -720/14 \\ 0 & 12/14 & 576/14 \\ 0 & 0 & 1 \end{bmatrix}$$

The resident program in practice uses the true transformation in use rather than any theoretical one if the two differ.

Examples are given in table 11.35.

Table 11.35: VideoToGraphics parameters

Scenario	Input points	Output points
Broadcast is a 16:9 coded frame of size 720 x 576. The display is 16:9 and no DecFC is being applied.	(40, 60) (-500, -500)	(40, 60) (-500, -500)
Broadcast is a 16:9 coded frame of size 720 x 576. The display is 4:3 and the DecFC is centre-cut-out.	(0, 0) (360, 100)	(-120, 0) (360, 100)
Broadcast is a 16:9 coded frame of size 352 x 288. The display is 4:3 and the video is being shown in a 16:9 letterbox.	(0, 0) (720, 576)	(0, 72) (720, 504)
Broadcast is a 4:3 coded frame of size 720 x 576. The display is 4:3 and the video has been scaled to 360 x 288 and positioned at (200, 200).	(0, 0) (720, 576)	(200, 200) (560, 488)
Broadcast is a 16:9 coded frame of size 1 920 x 1 080. The Scene co-ordinate system is 720 x 576. The display is 16:9 and no DecFC is being applied.	(40, 60) (-500, -500)	(40, 60) (-500, -500)

11.10.10.2 SetWidescreenAlignment

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

Arguments: SWA(mode). See table 11.36.

Table 11.36: SetWidescreenAlignment parameters

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

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 11.10.17.

11.10.10.3 GetDisplayAspectRatio

Synopsis: Returns the aspect ratio of the display.

Arguments: GDA(aspectratio). See table 11.37.

Table 11.37: GetDisplayAspectRatio parameters

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

11.10.10.4 SetSubtitleMode

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

Arguments: SSM(on). See table 11.38.

Table 11.38: 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. True = enable.

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 clauses 11.10.17 and 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.

11.10.10.5 SetBroadcasterInterruptions

This resident program shall be implemented only in receivers that support NativeApplicationExtension.

Synopsis: Provides a means to define whether the MHEG application can be interrupted by a broadcast-triggered native application.

Arguments: SBI(status). See table 11.39.

Table 11.39: SetBroadcasterInterruptions parameters

In/out/in-out	Type	Name	Comment
input	GenericBoolean	status	Enables or disables the receiver's ability to respond to other broadcaster signalling, e.g. promotional linking which may interrupt or overlay the current application. True = Enabled False = Disabled

Description: This resident program defines how broadcast-triggered native applications shall co-exist with a running MHEG-5 application. It enables a running MHEG-5 application to permit interruption from other broadcast-triggered native applications at certain times, e.g. when running in the background, and prohibit interruption when that would create an undesirable user experience, e.g. while the user is interacting with the running MHEG-5 application.

If the status is enabled, the receiver may interrupt or suspend the current MHEG-5 application e.g. to display a broadcast-signalled message.

The MHEG-5 application shall run until a broadcast-triggered native application is signalled by SI and the receiver is not capable of simultaneous presentation of the broadcast application and the MHEG-5 application.

If the receiver is capable of simultaneous presentation of the broadcast application and the MHEG-5 application, the MHEG-5 application may continue to run but may not receive some UserInput events. See clause 11.6.5. This resident program shall not affect the receiver's response to keys other than those in the Register 3 group.

If the status is disabled, broadcast-triggered native application (e.g. promotional link events) shall not interrupt (or overlay) the current MHEG-5 application.

User-initiated actions can always pause, suspend or kill the MHEG-5 application, see clause 14.9.

The effect of changing the status shall be immediate irrespective of whether a broadcast-triggered native application has already been signalled. If the receiver is currently showing a broadcast-triggered native application with which the user has not interacted when the status is set to disabled, that broadcast-triggered native application shall immediately be removed and any keys in use by the application shall be available to the MHEG-5 application.

The default status shall be enabled, unless following a service tune where modified by bit 4 of a call to SI_TuneIndexInfo (see clause 11.10.8.4).

The status is a receiver global value shared by an auto-boot Application and any subsequent Applications invoked by a Launch or Spawn. The status shall be reset to the default value when the last Application in the Application stack is terminated.

11.10.10.6 GetAudioDescPref

Synopsis: Returns the viewer preference for audio description.

Arguments: GAP(viewerPref).

Table 11.40: GetAudioDescPref parameters

In/out/in-out	Type	Name	Comment
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	viewerPref	Returns true if the current viewer preference is for the receiver to use the audio description soundtrack.

Description: This resident program allows an MHEG application to determine whether audio description is currently enabled. This may be used to allow a content provider to make available streams with and without audio description to avoid streaming unused data.

11.10.10.7 GetSubtitlePref

Synopsis: Returns the viewer preference for subtitles.

Arguments: GSP(viewerPref).

Table 11.41: GetSubtitlePref parameters

In/out/in-out	Type	Name	Comment
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	viewerPref	Returns true if the current viewer preference is for the receiver to display DVB subtitles.

Description: This resident program allows an MHEG application to determine whether subtitles are currently enabled. This may be used to allow a content provider to make available streams with and without subtitles to avoid streaming unused data.

11.10.11 Common Interface

11.10.11.1 CI_SendMessage

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

Arguments: CIS(Message, Response). See table 11.42.

Table 11.42: CI_SendMessage parameters

In/out/in-out	Type	Name
input	GenericOctetString	Message
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	Response

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 14.10.3.4). The FileRequest RequestType shall 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 14.10.3.5).

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

11.10.12 Interaction channel

11.10.12.0 Introduction to interaction channel resident programs

The resident programs defined in this clause shall be implemented only in receivers that implement InteractionChannelExtension and optionally ICStreamingExtension.

11.10.12.1 GetICStatus

Synopsis: Returns the availability of the interaction channel.

Arguments: GIS(ICStatus) See table 11.43.

Table 11.43: GetICStatus parameters

In/out/in-out	Type	Name	Comment
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	ICstatus	0 = Active 1 = Inactive 2 = Disabled

Description: The return value of this resident program provides dynamic information to indicate the status of the interaction channel connection available to the receiver.

"Active" means that the interaction channel has not been disabled by the viewer and that the network interface is physically connected and fully configured.

NOTE: The network interface is considered to be fully configured if the necessary parameters have either been provided by the viewer or have been obtained automatically.

"Inactive" means that the interaction channel has not been disabled by the viewer and that the network interface is not physically connected or is not fully configured.

"Disabled" means that the interaction channel has been disabled through the action of the viewer.

If the receiver has more than one connection, it should respond with the status of the "best connection". A reasonable interpretation of "best" would be the connection that would retrieve content in the shortest time.

11.10.12.2 ReturnData

Synopsis: Sends data to a remote server using the HTTP POST method.

Arguments: RDa(url, [name, value]..., responseCode, responseData). See table 11.44.

Table 11.44: ReturnData parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString	url	This input parameter specifies the URL to which data will be sent. See clause 16.3.2.5.
input	GenericOctetString	name	A list of name/value pairs to be sent to the server.
input	GenericBoolean , or GenericInteger , or GenericOctetString , or GenericObjectReference , or GenericContentReference	value	
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	responseCode	The response code from the server. Zero if no response code was returned.
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	responseData	The data returned by the server, if any.

Description: This resident program takes a variable number of arguments. One or more name and value pairs may be present. The name and value arguments shall be used to construct a data set of content type application/x-www-form-urlencoded as specified by section 17.13.4 of HTML 4.01 [26] except that references to IETF RFC 1738 [27] shall be taken as references to IETF RFC 3986 [28], which updates it. Given "name1%" = "value1" and "name2" = "contains spaces", this will produce a data set of the form name1%25=value1&name2=contains+spaces, where each of the names and values has been percent-encoded, except that space characters are replaced with "+".

The data set may contain characters that are not represented in the US-ASCII character set; consequently the percent-encoding shall be carried out as specified for characters from the Universal Character Set in section 2.5 of IETF RFC 3986 [28]. Characters shall be assumed to be encoded as UTF-8.

EXAMPLE: The character Latin Capital Letter 'A With Grave' is represented in UTF-8 by the octets 0xC380. In the text representation of MHEG, this character is written as '=C3=80'; after percent-encoding this becomes "%C3%80".

This data set shall form the Entity-Body of an HTTP POST request. The server's response code and any data returned by the server shall be returned to the MHEG application. The response code shall be interpreted in accordance with clause 15.7.3. If no valid response code is returned by the server, the responseCode parameter shall be set to zero.

GenericOctetString and GenericContentReference arguments shall be treated directly as strings. GenericInteger arguments shall be converted to strings as decimal integers with no leading zeros. GenericBoolean arguments shall be converted to the string "true" if true and to the string "false" if false. GenericObjectReference arguments shall be converted to a string consisting of the GroupIdentifier (if any) followed by ",", (0x2C) followed by the ObjectNumber as a decimal integer with no leading zeros.

In any case where an invalid set of arguments is supplied (such as a missing value argument) the resident program call shall fail in accordance with clause 11.10.16.

See also clause 17.25.2.

11.10.12.3 MeasureStreamPerformance

Synopsis: Measures the IP-sourced stream presentation performance.

Arguments: MSP(url, maxBytes, speed). See table 11.45.

Table 11.45: MeasureStreamPerformance parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString	url	This input parameter specifies an optional URL that can be retrieved for the purposes of determining the throughput. See clause 16.3.2.5.
input	GenericInteger	maxBytes	The maximum number of bytes to retrieve. If set to 0 or a negative value the entire file may be retrieved.
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	speed	The average speed capability, in bytes/second or -1 if speed cannot be determined.

Description: This resident program allows an MHEG application to determine the maximum rate of stream delivery from its IP connection that can be expected to be successfully presented. The speed returned should consider all limitations on the stream presentation in addition to the performance of the IP connection, e.g. throughput of the HTTP stack.

The receiver shall use the URL provided to determine the time taken to retrieve and decode a representative item of content. The content shall not be made available to the MHEG application. The receiver shall not source the data from any internal cache. From the size of the content and the time taken to load it completely to memory, along with adjustment for any known internal restrictions, the receiver shall calculate the speed as bytes per second.

If maxBytes is a positive number then the receiver shall make a partial request from the server, using the following header:

- Range: 0-<maxBytes>

If the HTTP server does not support the Range header, the receiver shall close the connection after a maximum of maxBytes bytes have been transferred.

If for any reason the speed cannot be determined the program shall return a speed value of -1.

- Range: 0- (i.e. no upper bound in the range request)

If this form of Range Request is used, then the receiver shall close the connection after a maximum of maxBytes bytes have been transferred.

11.10.12.4 PromptForGuidance

Synopsis: Allows the receiver to verify that content is acceptable for presentation.

Arguments: PFG(restriction, result). See table 11.46.

Table 11.46: PromptForGuidance parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString	restriction	This input parameter provides a string that can be presented to the user to indicate why verification is required.
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	result	Indicates if the restricted content may be presented.

Description: This resident program allows an application to confirm that an item of content can be presented to the viewer, based on the restriction advice provided. The method of confirming that the content may be presented is not defined but shall not cause the MHEG application to be terminated. As an example if the receiver implements a "parental control" PIN it could display a dialogue to indicate that the content is restricted, using the restriction string to indicate why it is restricted, and ask the viewer to enter the PIN to allow presentation. If the PIN is not provided then the resident program shall return "false". For a receiver that does not implement any parental control mechanism the resident program shall set the presentable return flag to "true".

The restriction parameter contains a string that may be presented to the viewer. The string shall use UTF-8 representation of character codes. Any unrecognized bytes in the string may be ignored and the string shall be limited to 50 characters.

11.10.12.5 PersistentStorageInfo

Synopsis: Returns the status of true persistent storage.

Arguments: PST (enabled).

Table 11.47: PersistentStorageInfo parameters

In/ out/in-out	Type	Name	Comment
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	enabled	True = enabled False = disabled

Description: The resident program returns whether the user has enabled access to true persistent storage (see clause 14.7).

11.10.12.6 SetCookie

Synopsis: Sets a cookie for use by the receiver when making HTTP requests.

Arguments: SCK(identity, expiryDate, value, secure).

Table 11.48: SetCookie parameters

In/ out/in-out	Type	Name	Comment
input	GenericOctetString	identity	The name of the cookie to write. The string is formatted in the form: name, domain/path.
input	GenericInteger	expiryDate	The expiry date of the cookie, expressed as a Modified Julian date or zero if the cookie is transient.
input	GenericOctetString	value	The value to set for the cookie.
input	GenericBoolean	secure	Specifies the secure attribute (see clause 15.7.5).

Description: The resident program sets a cookie that may be used when making HTTP requests. The identity parameter defines the name of the cookie and the domain and path for which it applies.

EXAMPLE: The identity `userid,serverdomain.com/mheg` creates a cookie named `userid` which is to be sent with requests to servers in the `.serverdomain.com` domain for resources within the `/mheg` directory.

11.10.12.7 GetCookie

Synopsis: Retrieves a cookie set by the SetCookie resident program or previously received in an HTTP response.

Arguments: GCK(identity, found, expiryDate, value, secure).

Table 11.49: GetCookie parameters

In/ out/ in-out	Type	Name	Comment
input	GenericOctetString	identity	The name of the cookie to read. The string is formatted in the form: name, domain/path.
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	found	True if the requested cookie was found.
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	expiryDate	The expiry date of the cookie, expressed as a Modified Julian date or zero if the cookie is transient.
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	value	The value of the cookie.
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	secure	True if the secure attribute was specified (see clause 15.7.5).

Description: The resident program reads a cookie set by the SetCookie resident program or previously received in an HTTP response. The identity parameter defines the name of the cookie and the domain and path for which it applies.

EXAMPLE: The identity userid,serverdomain.com/mheg reads a cookie named userid which is to be sent with requests to servers in the .serverdomain.com domain for resources within the /mheg directory.

11.10.12.8 GetPINSupport

Synopsis: Returns the PIN support in the receiver.

Arguments: GPS(support).

Table 11.50: GetPINSupport parameters

In/ out/ in-out	Type	Name	Comment
output	GenericInteger (shall provide an IndirectReference to a IntegerVariable)	Support	Indicates the PIN support in the receiver -1 = PIN is not supported 0 = PIN is supported and disabled 1 = PIN is supported and enabled

Description: This resident program allows an MHEG application to determine whether a parental control mechanism using a PIN is supported in the receiver and whether it is currently enabled or not.

11.10.13 Hybrid file system

11.10.13.0 Introduction to hybrid file system resident programs

The resident programs defined in this clause shall be implemented in receivers that implement InteractionChannelExtension.

11.10.13.1 SetHybridFileSystem

Synopsis: Defines or modifies the mappings used by the hybrid file system.

Arguments: SHF(pathName, mappingList). See table 11.51.

Table 11.51: SetHybridFileSystem parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString	pathName	This input parameter specifies a pathname which is to be mapped to a list of locations.
input	GenericOctetString	mappingList	This input parameter specifies a list of locations to which the pathname is to be mapped.

Description: This sets up a mapping from a pathname within the hybrid file space to a list of locations. A pathname is composed of the following terms in sequence:

- Path Origin.
- zero or more instances of a Path.
- zero or one instances of a Filename.

where those terms have the meanings defined in table 16.2.

A location is composed of the following terms in sequence:

- Source.
- Path Origin.
- zero or more instances of a Path.
- zero or one instances of a Filename.

where those terms have the meanings defined in table 16.2, save that the Source shall be that of any file system available to the receiver other than the hybrid file system and persistent storage (including true persistent storage).

Considering the hybrid and other file systems as tree structures, each pathname or location shall therefore be a reference to a node of such a tree. A pathname or location shall end with "/" (0x2F) unless it refers to a terminal node (a file as opposed to a directory).

The mappingList argument shall consist of one or more locations; if there is more than one location, each shall be separated by a space character (0x20).

If the pathName argument ends with "/", each location within the mappingList argument shall end with "/", otherwise each location shall not end with "/".

The only permitted exceptions are as follows:

- If mappingList is an empty string (and pathName is not an empty string), the mapping for that pathName (if any) shall be removed. Receivers shall ignore any request to remove the default mapping in this way.
- If pathName is an empty string, all mappings shall be removed and the default mapping restored.

If the arguments to the resident program are not in accordance with the requirements stated above, the resident program shall not alter the mapping table.

If a mapping is made for a pathname for which there is an existing mapping, the new mapping shall replace the existing one. The default mapping cannot be deleted but can be replaced by this mechanism.

Changes made to the state of the mapping table shall not affect any requests for content that are in progress. Requests for content made through the hybrid file system shall proceed according to the state of the mapping table at the time the request was made.

11.10.14 Developer utilities

11.10.14.1 WhoAml

Synopsis: Returns all engine identification strings recognized by the engine.

Arguments: WAI(ident). See table 11.52.

Table 11.52: WhoAml 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.

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) (see table 11.3) GetEngineSupport request (see clause 11.4.1).

EXAMPLE: A receiver might return the string "FEG001103 MHGFEG056 DSMFEG017".

11.10.14.2 Debug

Support for this resident program is optional. If implemented, it 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 computer TV card may open a text window.

Synopsis: Allows output of debug messages.

Arguments: DBG([argument] ...). See table 11.53.

Table 11.53: 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.

Description: This resident program outputs a list of zero or more input arguments. The exact output representation for each variabletype is implementation dependent, table 11.54 provides examples of suitable output.

Table 11.54: 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.

11.10.15 Access to application lifecycle signalling

11.10.15.1 GetBootInfo

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

Arguments: GBI(infoResult, bootInfo). See table 11.55.

Table 11.55: 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 is present in the data_broadcast_id descriptor on the component from which the initial carousel was mounted. 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".

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).

11.10.16 Data exchange with ResidentPrograms

11.10.16.0 Scope of behaviour

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

11.10.16.1 Memory spaces

There are two distinct memory spaces to consider:

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

11.10.16.2 On invocation

When a ResidentProgram is invoked (with Call or Fork) the behaviour shall be 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.

11.10.16.3 CallSucceeded/ForkSucceeded Values

Clause 14.4 of the ISO/IEC 13522-5 [14] (MHEG-5) specification 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 shall be deemed to have succeeded even if, for example, the input parameters are formatted with incorrect values such as null strings or negative integers.

11.10.16.4 During execution

While the procedural code executes there shall be 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.

11.10.16.5 On completion

If the procedural code completes normally then its results (any in-out, output parameters and the succeeded parameter) shall be transferred back to the MHEG-5 application memory. All of the results of the ResidentProgram shall be 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.

11.10.17 Duration of effect of ResidentPrograms

Certain ResidentPrograms affect the state of the receiver. 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). The state shall be reset before any links in the new application can fire.

Receivers that implement InteractionChannelExtension shall vary the above solely for the state of the hybrid mapping table (see clause 15.12). This shall persist until the auto-boot process begins (see clause 9.3.4.2) such that Applications inherit the state of the mapping table from previously running Applications. When an Application is spawned, the state of the hybrid mapping table shall be saved; when a spawned Application quits, the state of the hybrid mapping table shall be restored before the previous Application is restarted.

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 11.56.

Table 11.56: BooleanVariable values

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

11.11.2 IntegerVariable

See table 11.57.

Table 11.57: 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 shall be restricted to be not longer than 2 147 483 647 octets long, thus the only practical restriction should be that of available memory (see clause 14.8).

A null OctetString value shall be encoded as "".

11.11.4 ObjectNumber

The ObjectNumber data type shall be restricted as shown in table 11.58.

Table 11.58: ObjectNumber values

Type of value	Size	Min. value	Max. value
signed integer	32 bits	1 (see note)	2 147 483 647
NOTE: Group objects shall use the reserved value of 0.			

11.11.5 GroupIdentifier and ContentReference

GroupIdentifiers shall be at least 1 byte in length. ContentReferences may be 0 bytes length. In both cases the maximum length has the same restrictions as an OctetString.

See clause 16.3.2.

11.12 Extensions to the MHEG-5 language specification

11.12.1 Preamble

The present document specifies a number of changes to the MHEG-5 language that shall 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

11.12.2.0 Overview of changes

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 the ISO/IEC 13522-5 [14] (MHEG-5) specification.) 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. <ul style="list-style-type: none"> 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification) is added. All references in the text to Scene shall be replaced by Group. The text form of the syntax description shall be modified as follows:
---------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Text form:

```
SetTimer ::= ":SetTimer" "(" Target TimerID [NewTimer] ")".
NewTimer ::= TimerValue [AbsoluteTime].
```

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.

- OriginalDesktopColour** Colour to be used for the Desktop (i.e. the bottom of the display stack).
- Optional Integer or OctetString. An Integer shall be interpreted as an index in a Palette; an OctetString shall be interpreted as a direct colour value.
 - Default value: Opaque black.

ASN.1 form:

A.3 Application Class

```
ApplicationClass ::= SET
{
    COMPONENTS OF GroupClass,
    on-spawn-close-down [35] ActionClass OPTIONAL,
    on-restart [36] ActionClass OPTIONAL,
    default-attributes [37] SEQUENCE SIZE (1..MAX) OF DefaultAttribute
OPTIONAL,
    original-desktop-colour [249] Colour OPTIONAL
}
```

Text Form:

```
ApplicationClass ::= "{:Application" Group [OnSpawnCloseDown] [OnRestart]
[DefaultAttributes] [OriginalDesktopColour] }".
OriginalDesktopColour ::= ":OrigDesktopColour" Colour.
```

11.12.3.2 Changes to "Own internal attributes"

- DesktopColour** Colour to be used for the Desktop (i.e. the bottom of the display stack).
- Integer or OctetString value. An Integer shall be interpreted as an index in a Palette; an OctetString shall be interpreted as a direct colour value.
 - Initial value: Value of the OriginalDesktopColour attribute.

11.12.3.3 Changes to "Effect of MHEG-5 actions"

The wording of the :Launch and :Quit elementary actions of the Application class appear to render the :OnCloseDown attribute useless. The wording is modified so as to resolve this problem and bring the behaviour for the Application class in line with that of the Scene class and its :TransitionTo elementary action.

- Launch** Steps 1 and 2 of the sequence of actions are modified to read:
- 1) Apply the Deactivation and Destruction behaviours of the currently active Scene object, if any.
 - 2) Apply the Deactivation and Destruction behaviours of the currently active Application object, if any.
- Quit** Steps 1 and 2 of the sequence of actions are modified to read:
- 1) Apply the Deactivation and Destruction behaviours of the currently active Scene object, if any.
 - 2) Apply the Deactivation and Destruction behaviours of the target Application object.

SetDesktopColour
(NewDesktopColour) Set the DesktopColour attribute of the target Application object.

ASN.1 form:

```
set-desktop-colour [250] SetDesktopColour
SetDesktopColour ::= SEQUENCE
{
    Target GenericObjectReference
    New-desktop-colour NewColour
}
```

Text form:

```
":SetDesktopColour" "(" Target NewColour ")"
```

11.12.4 Changes to the Scene class

11.12.4.0 Overview of changes

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"

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 attributes are 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).

BitmapScale Value of the bitmap scaling in pixels to be used when presenting the Bitmap object.

- Pair of Integers (XBitmapScale, YBitmapScale).
- Initial value: this part of the ISO/IEC 13522-5 [14] (MHEG-5) specification does not define the initial values for the BitmapScale attribute; these values may be specified in the application domain.

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.
```


ScaleBitmap
(NewXScale,
NewYScale)

If the MHEG-5 engine implements the **Scaling** option, the effect of this action is described below. Engines that do not support the **Scaling** option shall ignore this action.

NOTE: This action does not affect the **BoxSize** internal attribute of the **Bitmap** object; in other words, the **Bitmap** is scaled, but its bounding box remains the same.

Execute the following sequence of actions:

- 1) Set the **BitmapScale** attribute according to the values of **NewXScale** and **NewYScale**.
- 2) Adapt the rendering of the **Bitmap** so that it fits to the **XBitmapScale** and **YBitmapScale** dimensions. The **XBitmapScale** and **YBitmapScale** attributes represent the final dimensions of the **Bitmap** in pixel numbers. Thus, the graphical representation of the **Bitmap** may not keep its original aspect ratio.

Provisions of use:

- The Target object shall be an available **Bitmap** object.
- **NewXScale** and **NewYScale** shall be positive Integers.

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.

	<ul style="list-style-type: none"> • Initial value: OriginalTextColour.
BackgroundColour	<p>Colour to use for the background area when representing the Text object.</p> <p>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.</p> <ul style="list-style-type: none"> • 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	<p>This attribute is used to set the specific Font attributes such as style, character size, text colour and background colour.</p> <p>The exact encoding format of the FontAttribute attribute is related to the value of the type of Font object mentioned by the Font attribute.</p> <ul style="list-style-type: none"> • 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

11.12.9.0 Overview of changes

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]
"]".
```

11.12.9.2 Changes to "Own internal attributes"

The following internal attribute is added:

CounterMaxPosition	Position of the last frame of the Stream.
	This attribute is expressed in StreamCounterUnits.
	<ul style="list-style-type: none"> • Integer. • Initial value: -1, meaning EndOfStream.

11.12.9.3 Changes to "Internal behaviours"

The following internal behaviour is changed:

ContentPreparation	[Existing behaviour]
	<ol style="list-style-type: none"> 1) Attempt to set the CounterMaxPosition internal attribute to the position of the last frame of the Stream, expressed in StreamCounterUnits.

11.12.9.4 Changes to "Effect of MHEG-5 actions"

The following actions are added:

GetCounterPosition (CounterPositionVar)	Get the current position within a stream.
	Execute the following action:
	<ol style="list-style-type: none"> 1) Set the value of CounterPositionVar to the value of the CounterPosition attribute of the target Stream.
	Provisions of use:
	<ul style="list-style-type: none"> • The Target object shall be an available Stream object. • CounterPositionVar shall refer to an active IntegerVariable object.

ASN.1 form:

```
get-counter-position [251] GetCounterPosition
GetCounterPosition ::= SEQUENCE
{
    target GenericObjectReference,
    counter-position-var ObjectReference
}
```

Text form:

```
":GetCounterPosition" "(" Target CounterPositionVar ")"
CounterPositionVar ::= ObjectReference
```

GetCounterMaxPosition (CounterMaxPositionVar)	Gets the total size of a stream.
	Execute the following action:
	<ol style="list-style-type: none"> 1) Set the value of CounterMaxPositionVar to the value of the CounterMaxPosition attribute of the target Stream.
	Provisions of use:
	<ul style="list-style-type: none"> • The Target object shall be an available Stream object. • CounterMaxPositionVar shall refer to an active IntegerVariable object.

ASN.1 form:

```

get-counter-max-position [252] GetCounterMaxPosition
GetCounterMaxPosition ::= SEQUENCE
{
    target GenericObjectReference,
    counter-position-var ObjectReference
}

```

Text form:

```

":GetCounterMaxPosition" "(" Target CounterPositionVar ")"
CounterPositionVar ::= ObjectReference

```

11.12.10 Changes to the Video class

11.12.10.1 Changes to "Own internal attributes"

The following internal attributes are added:

VideoDecodeOffset	<p>Position of the top left corner of the decoded and scaled video with respect to the top left corner of the Video object.</p> <ul style="list-style-type: none"> • Pair of Integers (XOffset, YOffset). • Initial value: (0, 0).
VideoScale	<p>Value of the video scaling in pixels to be used when presenting the Video object.</p> <ul style="list-style-type: none"> • Pair of Integers (XVideoScale, YVideoScale). • Initial value: this part of the ISO/IEC 13522-5 [14] (MHEG-5) specification does not define the initial values for the VideoScale attribute; these values may be specified in the application domain.

11.12.10.2 Changes to "Effect of MHEG-5 actions"

The following actions are added:

SetVideoDecodeOffset (NewXOffset, NewYOffset)	<p>Change the location of the decoded and scaled video with respect to the target Video object. The offset parameters may be negative.</p> <p>Execute the following sequence of actions:</p> <ol style="list-style-type: none"> 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 11.2 illustrates quarter-screen video with masking to prevent display of pixels normally hidden at fullscreen size due to overscan.

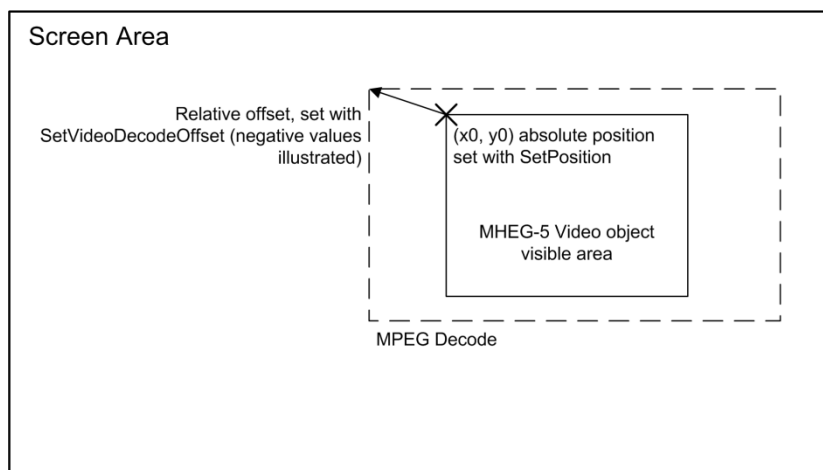


Figure 11.2: Quarter screen video (overscan)

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 11.3 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.

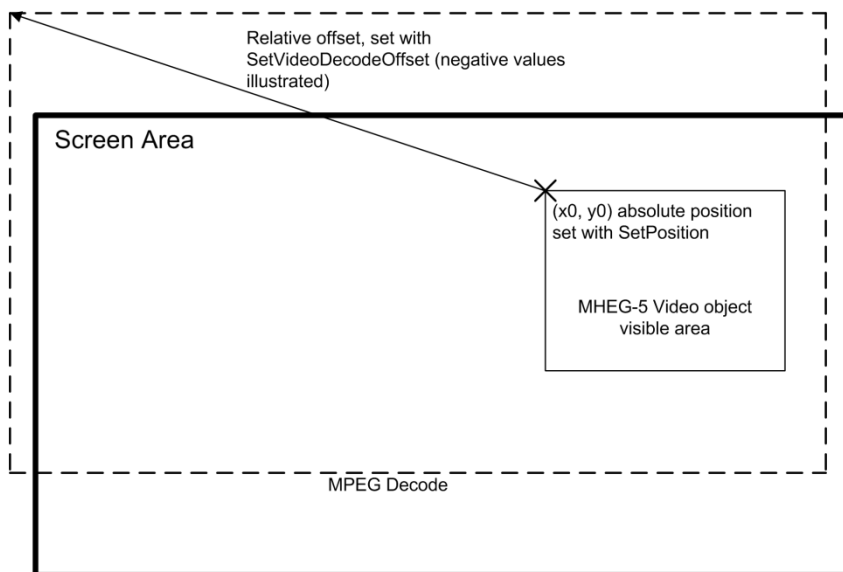


Figure 11.3: Quarter screen video presentation

Example code:

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

EXAMPLE 3

Figure 11.4 illustrates double size video masked to show a quarter of the video image fullscreen.

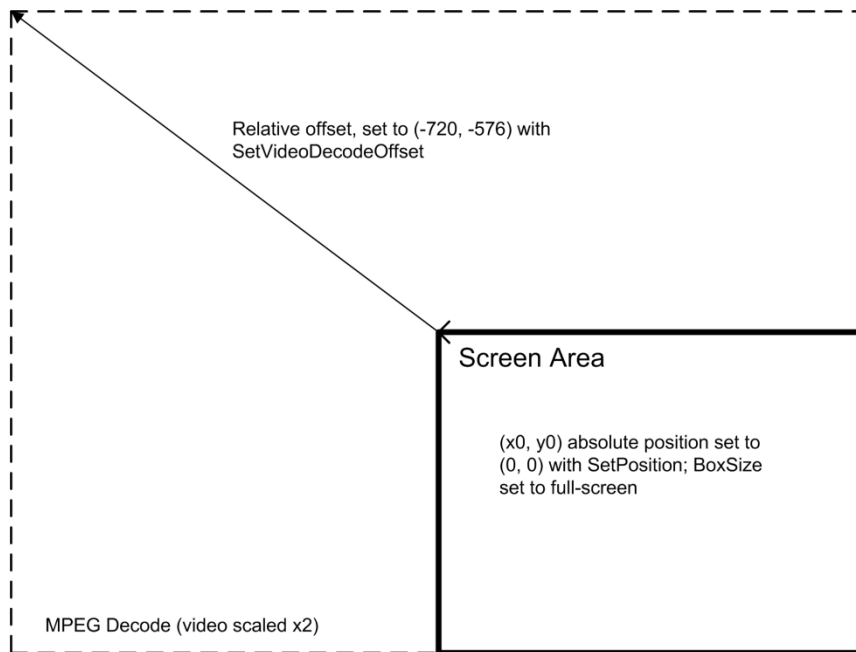


Figure 11.4: Quarter screen video (fullscreen)

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.
```

The following action is changed to the following description, to replace that in clause 39.4 of the ISO/IEC 13522-5 [14] (MHEG-5) specification:

ScaleVideo (NewXScale, NewYScale) If the MHEG-5 engine implements the **Scaling** option, the effect of this action is described below. Engines that do not support the **Scaling** option shall ignore this action.

NOTE: This action does not affect the **BoxSize** internal attribute of the Video object.

Execute the following sequence of actions:

- 1) Set the **VideoScale** attribute according to the values of **NewXScale** and **NewYScale**.
- 2) Adapt the rendering of the Video so that it fits to the **XVideoScale** and **YVideoScale** dimensions. The **XVideoScale** and **YVideoScale** attributes represent the final dimensions of the Video in pixel numbers. Thus, the graphical representation of the Video may not keep its original aspect ratio.

Provisions of use:

- The Target object shall be an available Video object.
- **NewXScale** and **NewYScale** shall be positive Integers.

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. <ul style="list-style-type: none"> Integer value. Initial value: Value of the OriginalMinValue attribute.
MaxValue	Greatest value that the SliderValue attribute may be set to. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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 of the ISO/IEC 13522-5 [14] (MHEG-5) specification before the final closing brace:

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

Text form:

Add the following to the EventTypeEnum in clause B.4.6 of the ISO/IEC 13522-5 [14] (MHEG-5) specification 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: <ol style="list-style-type: none"> 1) Apply the Interaction behaviour as defined in the Interactable 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification. However, the smallest marker displacement shall be proportional to the value of the StepSize attribute. Each time the marker is moved: <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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:
 - $\text{NewMinValue} < \text{NewMaxValue}$.
 - $N \times \text{NewStepSize} = (\text{NewMaxValue} - \text{NewMinValue})$, where N is some positive integer.
 - $\text{NewMaxValue} - \text{NewMinValue} \geq \text{Portion}$.

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	<p>Signals that the value of the <code>FocusPosition</code> attribute has been updated. This event will be generated when either the user moves the anchor highlight, or the <code>SetFocusPosition</code> ElementaryAction is invoked.</p> <ul style="list-style-type: none"> • Associated data: the new value of <code>FocusPosition</code> - Integer. • Event type: Asynchronous.
------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ASN.1 form:

Add the following to the `EventType` enumeration in clause A.6 of the ISO/IEC 13522-5 [14] (MHEG-5) specification before the final closing brace:

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

Text form:

Add the following to the `EventTypeEnum` in clause B.4.6 of the ISO/IEC 13522-5 [14] (MHEG-5) specification before the final full stop:

```
| "FocusMoved"
```

11.12.12.3 Changes to "Internal behaviours"

Interaction step 2	<p>The following text replaces the definition of the <code>HyperText</code> Interaction behaviour Step 2 in the ISO/IEC 13522-5 [14] (MHEG-5) specification:</p> <p>"Allow the user to move the focus through the set of anchors in the <code>Hypertext</code> object and to select the focused anchor. Each time the focus moves a <code>FocusMoved</code> event is generated.</p> <p>Each time an anchor is selected an <code>AnchorFired</code> event is generated."</p>
ContentPreparation	<p>The <code>ContentPreparation</code> internal behaviour semantics have changed from this object's base class as follows:</p> <ul style="list-style-type: none"> • Apply Steps 1 to 3 of the <code>ContentPreparation</code> behaviour of the base class synchronously. <p>The following steps are asynchronous and occur when the content of the object has been fully retrieved:</p> <ul style="list-style-type: none"> • Apply Steps 4 and 5 of the <code>ContentPreparation</code> behaviour of the base class. • Step 6: if there are no anchors then set the <code>FocusPosition</code> internal attribute to zero otherwise set it to one. If the <code>FocusPosition</code> internal attribute changes then generate a <code>FocusMoved</code> event. • Step 7: generate a <code>ContentAvailable</code> 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.
```

11.12.13 Changes to the LineArt class

11.12.13.0 Overview of changes

To remove the inconsistency in the requirements of OriginalLineWidth and NewLineWidth, zero values shall be permitted in both cases.

11.12.13.1 Changes to "Own exchanged attributes"

The following exchanged attribute is changed:

OriginalLineWidth	The second paragraph of definition for this attribute is modified to read: The OriginalLineWidth attribute is expressed in pixels in the scene coordinate space. It is specified in pixel height for horizontal lines and in pixel width for vertical lines. It shall not be negative.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

11.12.13.2 Changes to "Effect of MHEG-5 actions"

The following action is changed:

SetLineWidth (NewLineWidth)	The second paragraph of definition for this attribute is modified to read: The second provision of use is modified to read: NewLineWidth shall be set or refer to a non-negative integer value.
--------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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.

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

11.13.2.0 Relationship of CounterPosition to NPT

CounterPosition is an internal attribute that is set by the SetCounterPosition Elementary Action and updated as the Stream content is presented.

In the present document there is no support for managing NPT so there is no mapping between the MHEG-5 internal attribute CounterPosition of the Stream class and NPT.

11.13.2.1 Broadcast delivered streams

In the case where the Stream class references a broadcast stream, the Stream class CounterPosition attribute shall always remain in an undefined state, and any MHEG-5 actions which depend on this attribute (i.e. Stream class SetCounterTrigger, SetCounterPosition, GetCounterPosition, SetCounterEndPosition and GetCounterMaxPosition) shall be ignored. The associated internal attributes CounterEndPosition, CounterMaxPosition and CounterTriggers are also undefined.

11.13.2.2 IP delivered streams

In the case where the Stream class references a stream obtained from the IP connection the CounterPosition shall indicate an offset in the stream playback in units of 188 bytes.

When the content starts to be presented to the decoder the value of CounterPosition will be derived from the stream content position. The tolerance of the CounterPosition returned from GetCounterPosition shall be no more than +/-5 seconds.

Receivers are not required to wait until the new stream content has been loaded before returning from a SetCounterPosition call.

As a result of a **SetData ElementaryAction** the **CounterPosition** attribute is reset to 0 and the **CounterEndPosition** and **CounterMaxPosition** attributes are reset to -1. See clause 11.12.9.2.

As a result of **SetCounterPosition** the receiver may be required to refill its internal buffers. This shall not cause a **StreamUnderflow** event to be generated.

After a **SetCounterPosition ElementaryAction**, the presentation of video and audio is implementation dependent until the receiver is able to present the first frame from the new **CounterPosition**.

Where the **CounterPosition** is set to a valid value greater than **CounterEndPosition**, e.g. using the **SetCounterPosition ElementaryAction**, it shall indicate the end of stream. Note that the **CounterMaxPosition** attribute has a value of -1 until the size of the stream can be determined. For broadcast streams and IP-delivered streams that do not terminate the value will remain equal to -1.

Where the **CounterEndPosition** is set to a value greater than -1 and less than or equal to the **CounterPosition**, e.g. using the **SetCounterEndPosition ElementaryAction**, then the effect is implementation dependent for the current loop and for subsequent loops.

For IP-delivered streams the size of the stream should be calculated from the **Content-Length** header provided by a server. If a **Content-Range** header is present, this should be used in preference to the **Content-Length** header so that **CounterMaxPosition** will represent the size of the stream rather than the amount of data transferred over HTTP. Where neither **Content-Length** nor **Content-Range** headers are present **CounterMaxPosition** shall remain -1. Where **CounterMaxPosition** is -1 the effect of **ElementaryActions** that change the **CounterPosition** or **CounterEndPosition** is undefined.

11.13.3 Synchronous event processing

11.13.3.0 Interpretations of synchronous event processing behaviour

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 the ISO/IEC 13522-5 [14] (MHEG-5) specification are allowed in the present document 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification. 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification 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 shall be 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 17.14.1.

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 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 shall 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 shall be applied to all active **Ingredient** objects of the currently active **Application** object that have the **Shared** parameter set to **False**. This shall happen regardless of whether or not there is a **Scene** active before the **TransitionTo** action is executed.

11.13.6 Interactibles

In accordance with the ISO/IEC 13522-5 [14] (MHEG-5) specification, MHEG-5 objects that belong to the class **Interactable** 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 **Interactable** 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

Table 11.59 defines the terminology used in ISO/IEC 13522-5 [14] for each stream type.

Table 11.59: MHEG stream type terminology

	Broadcast	IP Delivered and Memory
Stopped	A Stream shall be considered to be " Stopped " either if it has never raised a StreamPlaying event or it has subsequently raised a StreamStopped event corresponding to the last StreamPlaying event.	
Playing	A Stream shall be considered as " Playing " if it has raised a StreamPlaying event and has not yet subsequently raised a StreamStopped event.	
Started Playing	A " Stopped " Stream has " Started Playing " when its state is changed such that it has a valid ContentReference and the Stream object is running.	A " Stopped " stream has " Started Playing " when the first byte of any of its component data, corresponding to the CounterPosition from which the Stream is being started, is presented to the decoder, regardless of the decoder's ability to decode the stream.
Stopped Playing	A " Playing " Stream has " Stopped Playing " when its state is changed such that it has an invalid ContentReference or the Stream object is not running.	A stream shall be considered as " Stopped Playing " when the last piece of a " Playing " Stream's component data is presented to the user. This could be as a result of reaching the CounterEndPosition at the end of the last playout loop or as a result of an Elementary Action, or if decoding failed.

A Stop action on a "Stopped" Stream shall not raise a StreamStopped event.

A SetData action with a valid ContentReference invoked on a "Playing" Stream object shall not cause StreamStopped and StreamPlaying events to be generated. A SetData action with an invalid ContentReference shall cause a StreamStopped event to be generated. A valid ContentReference is one that can be resolved to a stream, which may not necessarily include presentable audio or video components. See also clause 14.2.8.

Where a stream has "Stopped Playing", the stream shall continue playing as a result of a SetCounterPosition with a new position within the stream. There is no need for Stop and Run elementary actions to restart playback.

The CounterPosition attribute is reset to zero when a SetData actions targets a stream. It shall not be reset of zero when the stream has "Stopped Playing".

See also clause 14.2.5.

11.13.8 Use of NextScenes to preload content

In the present document 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 11.60.

Table 11.60: Application defaults

Attribute	Default value	
Font	rec://font/uk1 (see table 16.1)	
FontAttribute	Attribute	Value
	Size	24 pt
	Line Spacing	28 pt
	Letterspace	0
Style	plain	
TextColour	'=FF=FF=FF=00'	
HighlightRefColour	'=FF=FF=FF=00'	
SliderRefColour	'=FF=FF=FF=00'	
CharacterSet	10	
TextContentHook (see note 2)	10	
BitmapContentHook	4	
StreamContentHook	10	

Attribute	Default value
DesktopColour (see note 1)	'=00=00=00=00'
NOTE 1: I.e. the colour of the bottom of the display stack.	
NOTE 2: Applies also to subclasses of Text.	

11.13.10 Effect of SetData on Internal Attributes

In the present document, **SetData** actions targeted to **Stream** and **Bitmap** objects shall 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].

Additionally, the **SetData** action targeted to a **Stream** object does not reset the **Speed** attribute. Triggers are not reset when **SetData** action is applied.

11.13.11 Clarification of TransitionTo, Launch and Spawn behaviour

The ISO/IEC 13522-5 [14] (MHEG-5) specification defines the sequences of actions that take place in response to **TransitionTo**, **Launch** and **Spawn ElementaryActions**. In the present document 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 shall be ignored. See also clause 17.16.

11.13.12 References to shared=FALSE ingredients

In the present document, item I.B. of clause 51 of the ISO/IEC 13522-5 [14] (MHEG-5) specification shall be read as if the word "shared" were omitted, i.e. "Reference to an Ingredient of the active Application:".

11.13.13 Restrictions on Link EventSource

Clause 51 of the ISO/IEC 13522-5 [14] (MHEG-5) specification places restrictions on the types of **ObjectReference** that can be dereferenced within the context of an **Application**. In the present document, the use of **ObjectReferences** within **Application** objects is further restricted for the specific case of the **LinkCondition** attribute of **Links** contained within an **Application**. Such **Links** shall not specify an **EventSource** that references other **Applications** or **Scenes**. Receiver behaviour is not defined if such references are made.

11.13.14 Video Termination attribute

In the present document, all streams containing video shall be considered to be continuous and do not come to an end. The **Termination** attribute of the **Video** class shall therefore be ignored for broadcast streams.

Streams obtained from the **Interaction Channel** can come to an end; receivers shall observe the semantics of the **Termination** attribute when presenting such streams.

When a stream obtained from the **Interaction Channel** comes to an end and the **Stream** object has its **Termination** attribute set to **Freeze**, the video decoder output shall freeze showing the final complete frame of the stream. If the **SetData** action is targeted to such a **Stream** that has ended in this way, the presentation shall remain frozen until new content becomes available.

If the MHEG engine is de-prioritized when the stream is terminated then the state of video presentation when it is reprioritized shall be undefined.

11.13.15 Clarification of Root object destruction behaviour

When an object dynamically created using **Clone** is destroyed, the MHEG-5 engine shall free all resources (including internal resources) allocated to the object. This means that a destroyed object created with **Clone** cannot be re-prepared. In this case, the **Root** class **Destruction** behaviour shall be carried out as if the **GroupCachePriority** of the group containing the object were 0, regardless of the assigned value. This means that applications should be able repeatedly to **Clone** an object and **Unload** the clone without leaking memory.

11.13.16 Illegal parameter handling in :SetVariable

If the parameter passed to the **:SetVariable** **ElementaryAction** cannot be parsed because it is incorrectly formed (for instance it cannot be parsed as an **Integer**) then the effect on the target variable is implementation specific.

11.13.17 Referencing Group objects with an InternalReference

As an extension to ISO/IEC 13522-5 [14], the definition of internal-reference is changed to allow an object-number of 0.

```
ObjectReference ::= CHOICE
{
  external-reference ExternalReference
  internal-reference INTEGER (0..MAX)
}
```

This does not affect the external-reference format; Groupobject definitions shall still include a group-identifier.

11.14 Service Information extension

11.14.0 Introduction

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

The string format supported by DVB Event sections is not compatible with the present document. Therefore the information strings obtained from these sections shall be filtered to make them presentable.

All strings provided by the ResidentPrograms of the Service Information extension shall be encoded in UTF-8. Furthermore the control codes (0x80-0x9F and 0xE080-0xE09F) shall not be present. Where the CR control code (0x8A, 0xE08A) is present in the section it shall be replaced with the value 0x0D.

NOTE: This may result in the string containing characters that are not available in the selected character repertoire.

11.14.1 Service Information resident programs

11.14.1.0 List of Service Information resident programs

Table 11.61: Service Information extension resident programs

Resident program		Invocation			Reference
		Typical use		Never Fork	
Description	Name	Call	Fork		
SI_GetServiceInfo	SeI	✓			Clause 11.14.1.1
SI_GetEventInfo	GEI	✓			Clause 11.14.1.2

11.14.1.1 SI_GetServiceInfo

Support for this resident program is optional.

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

Arguments: SeI(serviceIndex, serviceName, serviceProvider, serviceType). See table 11.62.

Table 11.62: 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).

In/out/in-out	Type	Name	Comment
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	serviceType	Returns the service type (derived from the service's SI service descriptor).

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).

11.14.1.2 SI_GetEventInfo

Support for this resident program is optional.

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

Arguments: GEI(serviceIndex, porf, eventName, shortDescription, parentalRating, startDate, startTime, duration, category, freeNotCA). See table 11.63.

Table 11.63: 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 is 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.
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).

Description: The resident program returns 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").

11.15 PVR extensions

11.15.0 Introduction

This clause specifies extensions that are used for managing the booking of events on Personal Video Recorders (PVR); identified as the PVRExtension.

11.15.1 PVR Implementation

Referencing programme events by EIT event_id is not supported by this extension. Instead all events in EIT shall include a Content Identifier Descriptor (CID) that, in conjunction with a Default Authority Descriptor (DAD), provides a CRID for the event [45].

11.15.2 CRID format

CRIDs carried in SI may be defined in such a way that the Scheme and Authority parts are carried once only if they are common to a group of CRIDs. However the format for CRIDs passed across the MHEG PVR API in any of the Resident Program calls, as defined in clause 11.15.3, shall include the Scheme and Authority parts in all cases.

A CRID that does not include an instance identifier shall be in the format:

- Scheme + Authority + Unique Identifier

A CRID that includes an Instance Identifier shall be in the format:

- Scheme + Authority + Unique Identifier + # + Instance Identifier

Examples of CRIDs with and without Instance Identifiers follow:

- crid://company.tv/123df5
- crid://broadcaster.com/hef5w#e1

11.15.3 PVR extension resident programs

11.15.3.0 List of PVR extension resident programs

The resident programs defined in this clause shall be implemented in receivers that implement PVRExtension.

NOTE: In some implementations CRIDs carried in SI are case insensitive. However, MHEG-5 operations on strings are case sensitive. So:

- the Resident Programs defined in this clause shall be case preserving;
- operations relative to broadcast SI shall be case insensitive.

In general, using names that can be distinguished on non-case sensitive file systems will help authoring and content interchange.

Table 11.64 lists the ResidentPrograms that receivers implementing PVRExtension shall implement.

Table 11.64: Mandatory Resident Programs for receivers that implement PVRExtension

ResidentProgram		Invocation			Reference
		Typical use		Never Fork	
Description	Name	Call	Fork		
PVR_MakeBooking	PMB	✓			Clause 11.15.3.1
PVR_CancelBooking	PCB	✓			Clause 11.15.3.2
PVR_ListBookings	PLB	✓			Clause 11.15.3.3

11.15.3.1 PVR_MakeBooking

Synopsis: Adds an event to the PVR schedule.

Arguments: PMB(crid, cridType, name, description, result). See table 11.65.

Table 11.65: PVR_MakeBooking parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString	crid	
input	GenericInteger	cridType	The type of CRID being referenced
(input	GenericOctetString	name	A descriptive name for the event
input)	GenericOctetString	description	A brief description of the event
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	result	The result of the operation: 0 = booking successful 1 = alternate instance booking successful -1 = conflict with a previous booking -2 = CRID not found -3 = CRID already booked for recording -4 = booking cancelled by user -5 = booking failed for other reason -6 = booking failed due to insufficient space -7 = booking failed due to too many bookings

Description: This resident program adds an event to the receiver's list of scheduled events to record. The type of CRID can be any of a single programme or series - the type is defined in cridType to aid the receiver in finding the required CID. Where the CRID is a series CRID the booking relates to all programme events that are part of the series.

The receiver shall validate the CRID and check that resources are available for the booking. This may involve searching for multiple instances of an event until one is found that does not clash with a previous booking. Where no such instance is found the receiver may choose to indicate the conflict to the viewer, giving them the option to cancel one or more of the bookings.

Where the instance chosen to record (either automatically or by user intervention) is not the (temporally) first instance found the call shall return with a result value of 1.

The name and description can be used by the receiver to describe the event when it is placed in the booking list.

11.15.3.2 PVR_CancelBooking

Synopsis: Removes an event from the PVR schedule.

Arguments: PCB(crid, cridType, result). See table 11.66.

Table 11.66: PVR_CancelBooking parameters

In/out/in-out	Type	Name	Comment
input	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	crid	
input	GenericInteger	cridType	The type of CRID being referenced
output	GenericInteger (shall provide an IndirectReference to an IntegerVariable)	result	The result of the operation: 0 = booking removed -1 = the event is being recorded -2 = CRID not found -3 = the event has already been recorded

Description: This resident program removes the requested event from the list of events currently booked for recording. The event is referenced by its CRID, along with the CRID type.

Cancelling a series CRID shall cause all future events in the series to be ignored, in addition to ones currently visible in the schedule.

11.15.3.3 PVR_ListBooking

Synopsis: Returns a list of CRIDs and CRID types currently being monitored.

Arguments: PLB(crids_and_crid_types). See table 11.67.

Table 11.67: PVR_ListBooking parameters

In/out/in-out	Type	Name	Comment
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	crids	A space separated list of full CRIDs currently being monitored.

Description: This resident program returns an OctetString carrying a list of the currently valid bookings. The format for each booking shall be a CRID followed by CRID type. A single space (0x32) character separates each CRID and CRID type. A single space (0x32) character separates each booking.

The list shall contain all currently valid bookings previously added using the PMB Resident Program. There shall be no duplicate bookings in the list. In each booking the CRID and CRID type shall be in the format that was used when the booking was created using the PMB Resident Program. The order of bookings in the list is not defined. Bookings that are no longer valid because they have lapsed or have been successfully recorded or have been removed by other means shall not be present in the list.

EXAMPLE: An example of the OctetString returned from the PVR_ListBookings call describing one programme CRID and one Series CRID is as follows:

- "crid://service1.tv/4df6a2 49 crid://service1.tv/news 50"

11.16 InputMaskExtension

11.16.0 Introduction

The MHEG extension provides more flexibility in defining which remote control keys are captured by a running MHEG application and which are ignored, to be used by the receiver's native applications. The standard mechanism, defined in the MHEG-5 specification, allows only fixed sets of keys (called "Input Registers") to be enabled and disabled. If for some reason a profile wishes to add or remove a key from one of these sets then a new Input Register shall be defined within the profile. This extension allows an application to dynamically change the set of keys that are active from a pre-defined set of values that may or may not be implemented on the receiver.

The extension consists of the following changes:

- A new Scene attribute, the `InputEventMask`, defines the current set of key values to be captured for the application.
- A new `ElementaryAction` to change the value of the `InputEventMask` by providing a bitmask representing the key values to add to the mask.
- A `ResidentProgram` that allows an application to test if the receiver implements a specific set of keys before changing the `InputEventMask`. Again, the keys are represented as a coded bitmask.

11.16.1 Operation

11.16.1.0 Outline of operation

The `InputMaskExtension` allows application authors to define an arbitrary set of keys that can be requested by the application, enabling new key combinations without having to define new `Input Register` values. In addition it allows new profiles to define extended key sets in a less complex manner.

Each key, or in a few cases set of keys, is represented as a single bit value in a bitmask, the bit is set if the key is requested and unset if it is not required. The bitmask is encoded as a byte string (an MHEG `OctetString`). The byte string is variable length. Only the minimum sequence of bytes required to define the requested keys are coded. The receiver may not implement all the keys requested; in this case it shall provide all requested keys that are available. Applications may determine the availability of keys through use of a `Resident Program`. If the `Resident Program` is not implemented then the `InputMaskExtension` is not available.

To make it clear which mechanism to use the key `Input Event Mask` shall only be used when the current `Input Register` value is 0. If the `Input Register` value is not 0 then the keys described for that `Input Register` shall be made available to the MHEG Application. The value of `Input Register` may be set in the Scene attributes and may be changed using the `SetInputRegister Elementary Action`. `Input Register 0` itself therefore contains no keys.

11.16.1.1 Engine Events

If a key is not defined in the current `Input Event Mask` and the `Input Register` is 0 then any Engine Events associated with that key shall not be generated. This overrides the operation defined in clause 11.8 for the case where `Input Register` is 0; for all other values of `Input Register` the generation of Engine Events is unchanged.

11.16.2 Changes to the MHEG specification

11.16.2.1 Input Register semantics

A value of 0 for the `InputEventRegister` indicates that the `InputEventMask` shall be used to define the list of permissible `UserInput` events for the Scene. The value of `InputEventRegister` may be set either by the Scene definition or with the `SetInputRegister Elementary Action`.

11.16.2.2 New attributes and ElementaryActions

The following exchanged attribute is added to the Scene class: `InputEventMask`.

Encoded list of permissible `UserInput` events for the Scene.

- Optional `OctetString`.
- Default value: Empty String.

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,
```



```
input-event-mask [253] OctetString OPTIONAL
}
```

Text Form:

```
SceneClass ::= "{:Scene" Group InputEventRegister
SceneCoordinateSystem [AspectRatio]
[MovingCursor] [NextScenes]
[InputEventMask] "}".
InputEventMask ::= ":InputEventMask" OctetString
```

The following Elementary Action is defined. SetInputMask(NewInputMask).

Sets the Scene's InputEventMask to NewInputMask.

Provisions of use:

- The target object shall be an available Scene object.

ASN.1 Form:

```
set-input-mask [254] SetInputMask
SetInputMask ::= SEQUENCE
{
target GenericObjectReference,
new-input-mask GenericOctetString
}
```

Text Form:

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

11.16.2.3 TestInputMask New Resident Program

Synopsis: Allows an application to determine if a specific value of Input Event Mask is supported by the receiver.

Arguments: TIM(mask, result)

Table 11.68

In/out/in-out	Type	Name	Comment
input	GenericOctetString	Mask	The input mask to be tested. If the mask is an empty OctetString then result is true if the receiver supports <i>inputMaskExtension</i> .
output	GenericBoolean (shall provide an IndirectReference to a BooleanVariable)	result	If the receiver can provide all the requested keys defined in the mask to the application then the result is true, otherwise result is false.

11.16.2.4 Key values table

Table 11.69 defines for all key functions the UserInputEventData value they will generate and the bit in the InputEventMask that will enable the key to be processed by MHEG.

Byte 0 is the first byte in the bitmask OctetString, so the value 0:0x01 describes the LSB of the first byte.

Table 11.69

Function Name	UserInput EventData	InputEvent Mask Bit
Up	1	0:0x01
Down	2	0:0x02
Left	3	0:0x04
Right	4	0:0x08
0	5	0:0x10
1	6	0:0x10
2	7	0:0x10
3	8	0:0x10
4	9	0:0x10
5	10	0:0x10
6	11	0:0x10
7	12	0:0x10
8	13	0:0x10
9	14	0:0x10
Select	15	0:0x20
Cancel	16	0:0x40
Help	17	0:0x80
Red	100	1:0x01
Green	101	1:0x02
Yellow	102	1:0x04
Blue	103	1:0x08
Text	104	1:0x10
Info	105	1:0x20
Stop	120	1:0x40
Play	121	1:0x80
Pause	122	2:0x01
Skip Forward	123	2:0x02
Skip Back	124	2:0x04
Fast Forward	125	2:0x08
Rewind	126	2:0x10
Guide	300	2:0x20
Play/Pause	127	2:0x40
Record	400	4:0x01

EXAMPLE: The bitmask value for the current Input Register 3 is: Text + Cancel + Red + Green + Yellow + Blue, which is $1:0x10 + 0:0x40 + 1:0x01 + 1:0x02 + 1:0x04 + 1:0x08 = 0:0x40 + 1:0x1F$.

In MHEG text format this is '=40=1F'.

To create a bitmask that provides register 3 and the Guide key the value would be '=40=1F=20'.

To handle only the Guide key then the first 2 bytes are empty, but shall be added. So this would be '=00=00=20'.

11.17 File System Acceleration Extension

11.17.0 Introduction

This clause specifies extension for File System Acceleration which is a mechanism that allows receivers to persistently store a pre-selected part of a File System in order to improve the speed of subsequent file access.

File Groups that are signalled for storage shall persist across channel change and power cycles. Stored File Groups shall be accessible to any service that signals the presence of those groups using their group identifiers. Each file group has an owner id, a group id and a version to ensure file-system consistency. File Groups can consist of files from any accessible File System e.g. Object Carousel or Interaction Channel. The Group Location identifies the mount point from which the File Group shall be acquired and the default Group Location is the root of the current Object Carousel.

The physical medium used to store File Groups is not within scope of the present document.

11.17.1 File Groups

The parts of the File System to be stored are identified as File Groups. Signalling in the Object Carousel can define any number of file groups. File Groups are assigned a priority attribute to help receivers manage the store when total file group size exceeds receiver resources. File Groups include a receiver-profile to identify a sub-set of the total receiver population which is required to store that group.

File Groups are defined independently of File System structure. In particular there is no mapping between groups and DSM-CC modules, or between group-version and DSM-CC module version. Neither is there a relationship between the ordering of files listed in the File Group and the order in which those files shall be transmitted in a broadcast carousel or the order in which they shall be obtained from other sources.

When the File Group changes, due to files being added, removed, moved or updated the File Group version shall change. This enables a receiver to identify that a stored group is stale and to reacquire that File Group. Each File Group is versioned as a single unit and shall be re-acquired in its entirety on any version change.

11.17.2 Scope of File Groups

Each File Group is identified by an Owner Id and a Group Id, the Group Id being unique within the scope of an Owner Id. A File Group may be signalled as present in multiple File Systems and across multiple services. The receiver need store only a single copy of each File Group and make it available to each service in which the File Group is signalled as present.

11.17.3 Stored groups descriptor

The stored_groups_descriptor identifies the File Groups present in a File System and is located in the ServiceGatewayInfo::UserInfo bytes of the DownloadServerInitiate messages. Each instance of the stored_groups_descriptor may describe multiple groups and multiple instances of the descriptor may be used to describe all the File Groups available in the current service.

The structure of the stored_groups_descriptor is described in table 11.70.

Table 11.70: Stored Groups Descriptor

Syntax	bits	Type
stored_groups_descriptor {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (j=0;j<N;j++) {		
owner_id	16	uimsbf
group_id	16	uimsbf
group_priority	8	uimsbf
use_from_carousel	1	bslbf
use_version	1	bslbf
reserved	6	bslbf
receiver_profile	8	uimsbf
group_version	8	uimsbf
private_data_length	8	uimsbf
for (k=0;k<M;k++) {		
private_data_bytes		
}		
}		
}		

Where:

descriptor_tag: this 8-bit field identifies the descriptor. In the stored groups descriptor it is set to 0x80.

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

owner_id: this 16-bit field defines the owner of the group used to identify the broadcast corporation or platform identifier.

group_id: this 16-bit field defines the identifier for this group, which shall be unique within the scope of the owner_id.

group_priority: this 8-bit field defines the relative storage priority of the file group, as described in table 11.71.

NOTE 1: The priority used for the storage of individual applications needs to be profiled and managed by the managing authority so that a consistent behaviour is maintained across regions and across multiplexes within regions.

Table 11.71: Group priority values

Group_priority	Description
0x00	System group (platform wide)
0x01-0x09	User application priorities
0x0a-0xff	Reserved for future use

use_from_carousel: this 1-bit flag shall be set to '1' if the file group is accessible from the object carousel directly. It shall be set to '0' if the group may be accessed from store only and may not therefore be accessed directly from the File System.

NOTE 2: When the use_from_carousel flag is set to '0' this is intended to signal that the group will be acquired and stored and that only after the group has been stored in its entirety will the files within the group be made available to the application. Attempts to load files within the group will fail until such time that the group has been acquired and stored in its entirety.

NOTE 3: This flag is so called for historical reasons but is applicable to File Systems other than Object Carousel and the behaviour is as described above for all File Systems.

use_version: this 1 bit flag signals how the receiver should interpret the group_version. When set to '0' it indicates that a cached group shall be made available to the application irrespective of the signalled and stored group_version. If signalled and stored versions differ or the group is not present in the cache no attempt shall be made to update the cache. When set to '1' the receiver shall interpret the group_version as normal in order to maintain cached group coherency.

receiver_profile: this 8-bit field defines a receiver profile identifier that may be used to target receivers that conform to various minimum receiver profiles as described in table 11.72.

NOTE 4: A receiver may be part of multiple profiles.

Table 11.72: Receiver profile values

Receiver_profile	Description
0x00	All receivers
0x01	Receivers supporting the HDGraphicsPlaneExtension defined by see clause 12.11.1.1
0x02	Receivers supporting the HDVideoExtension defined by see clause 12.11
0x03	Receivers supporting the InteractionChannelExtension defined by see clause 15.1.2 and table 11.14
0x04	Receivers supporting the ICStreamingExtension defined by see clause 11.5.4 and table 11.15
0x05	Receivers supporting the ICEncryptedStreamExtension defined by see clause 11.5.4.4 and 15.16
0x06	Receivers supporting the LifeCycleExtension defined by see clause 8.1.7
0x07	Receivers supporting the NativeApplicationExtension defined by [xref]
0x08	Receivers supporting the DownloadableFontExtension defined by see clause 13.3.1
0x09-0x7f	Reserved
0x80	Receivers supporting the ServiceInformationExtension defined by clause 11.14.
0x81	Receivers supporting the PVRExtension defined by clause 11.15

group_version: this 8-bit field defines the version of the file group identified by the owner_id and group_id.

private_data_length: this 8-bit field defines the length of the private data bytes.

private_data_bytes: this field is not described in the present document.

11.17.4 Group Location Descriptor

The `group_location_descriptor` defined in table 11.73 may be included in the private data bytes of the `stored_groups_descriptor` and defines the Group Location. The Group Location is the mount point from which the Group Manifest and group files can be acquired.

Table 11.73: Group Location Descriptor

Syntax	bits	Type
<code>group_location_descriptor {</code>		
<code>descriptor_tag</code>	8	uimsbf
<code>descriptor_length</code>	8	uimsbf
<code>for (j=0; j<N; j++) {</code>		
<code>location_byte</code>	8	uimsbf
<code>}</code>		
<code>}</code>		

descriptor_tag: this 8-bit field identifies the descriptor and shall be set to 0x81.

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

location_byte: these 8-bit fields contain the string indentifying the Group Location. If the Group Location is a sub-directory then the location_bytes shall not include a trailing '/.

If the `group_location_descriptor` is not included then the default Group Location is the root of the current Object Carousel i.e. "DSM:/".

11.17.5 Group Manifest

The Group Manifest identifies the files that are to be stored. Each Group Manifest file is located in the Group Location and named after the following convention:

```
"/<owner_id>-<group_id>.man"
```

where `<owner_id>` and `<group_id>` are the values from the `stored_groups_descriptor`, which are 16 bit values each represented as 4 hexadecimal digits with lower case characters. If Group Manifest file is not present or is not correctly structured the File Group shall not be stored.

If describing a file group to be stored with `owner_id = 0x0001` and `group_id = 0x0002` the manifest file name is `"/0001-0002.man"`

The Group Manifest file shall use [JSON] format; a simple, open data structure, the format of which encourages concise documents. The structured format enables a schema to be created for the purpose of document validation.

Table 11.74 defines a [JSON] schema describing the format of the manifest file.

Table 11.74: JSON Schema for manifest

```

{
  "description": "Stored Group Manifest Schema",
  "type": "object",
  "properties": {
    {
      "version": {"type": "integer"},
      "size": {"type": "integer"},
      "nodes": {"type": {
        {
          "items": {
            {
              "type": "object",
              "properties": {
                {
                  "node": {"type": "string"},
                  "count": {"type": "integer", "optional": true}
                }
              }
            }
          }
        }
      }
    }
  }
}

```

Where:

version: is a version indicator for the file group. The version shall be changed when any files within the group change or files are added to or removed from the group. This value shall be consistent with the version field carried in the stored_groups_descriptor.

size: indicates the total size in bytes of the sum of files that are contained in the group.

nodes: is a container for node objects.

node: defines the path from the Group Location to the file that shall be stored as part of the current group. The default Group Location as defined in the current profile is the root of the current Object Carousel "DSM:/" but can be explicitly by the group_location_descriptor. The group location is implied so the terms "DSM:", "hybrid:" and "~" shall not be used. All paths shall begin with "/" to indicate the default group location. Each file required to be stored as part of the group shall be defined explicitly. Directory entries that comprise paths to files shall not be described.

count: is an optional field that if set describes that a number of files of given filename may be located at the current directory level. Each file has a numeric suffix in the range 001 to count. The extension to the filename shall be in the format "nnn" and shall include leading zeros. There is no implied dot separator between the filename and the extension. If a dot separator is required this shall be defined explicitly in the filename. The count mechanism may be used as a short-cut to describe a group that contains many, similarly named files and is intended to reduce the size of the manifest file only. The storage manager shall resolve and store the absolute path to each file at the time of storage in the same manner as when each file is referenced individually.

If the manifest includes fields not defined by the present document then they shall be ignored.

Table 11.75 provides an example of a manifest file that conforms to the previously describes JSON schema. White space has been added to the example to aid readability only. White space is not required in conformant JSON documents.

Table 11.75: Example JSON manifest

```

{"version" : 0,
 "size" : 2500000,
 "nodes" : [
   {"node" : "/path/to/files/file001"},
   {"node" : "/path/to/files/file002"},
   {"node" : "/path/to/lots_of_files/file", "count" : 99}
 ]
}

```

11.17.6 Group Manifest management

The group update and acquisition process relies on the group version signalled in the `stored_groups_descriptor` and in the manifest file to be of equal value. Changes to the descriptor and manifest file shall therefore be synchronized. The client shall attempt to reacquire the manifest file pertinent to a group for a period of five minutes after change in group version is signalled within the `stored_groups_descriptor`. If the group manifest file is not updated within the defined period this is an error state and the client may halt the acquisition of the group until either of the following is true:

- 1) a new version is signalled in the `stored_groups_descriptor`;

OR

- 2) the current Object Carousel is un-mounted and an Object Carousel is mounted that makes reference to the group.

The manifest file pertinent to a group shall be present in the File System for the same period over which the file(s) it describes are present in the File System.

The manifest file shall not be self referential or as an entry in any other manifest and shall therefore not be stored by this mechanism. However, receiver implementations may store the manifest for faster access on subsequent reads.

If the Group Location specifies an Interaction Channel file system then the Group Manifest file shall NOT be subject to Authentication mechanism defined in clause 15.13. Application Code and Content are accessed transparently from the Store and shall therefore be authenticated as normal. Certificate, Hash and Signature files may also be included in File Groups and cached.

11.17.7 Stored File Access

11.17.7.1 Read access

Read access to stored files shall be transparent to any process that accesses the File System in that file references into the File System shall instead reference the stored equivalent if the following conditions are true:

- 1) The file exists in the store and the group to which it belongs is not listed as stale.

AND

- 2) If the `use_version` flag is set the stored owner, group and version that reference the file agree with the owner, groups and version in the `stored_groups_descriptor`.

OR

- 1) The file exists in the store and the group to which it belongs is not listed as stale.

AND

- 2) If the `use_version` flag is not set the stored owner and group identifier that references the file agrees with the owner and group identifier in the `stored_groups_descriptor`.

The use of the stored File System by an MHEG application shall not interfere with other data delivery mechanisms such as Stream Events, Interaction Channel and Hybrid File System. If an MHEG application request data specified with a content-cache-priority of zero the file shall be obtained from the target File System and not from a stored File Group.

11.17.7.2 Write access

During the period over which a File Group is initially acquired from the File System or during the period over which an existing File Group is reacquired, the File Group shall be marked as 'stale' by the cache manager. Files in a stale group shall not be made available until the entire group has been acquired. Requests for files from a stale group shall resolve to the target File System instead of the cache only if the `use_from_carousel` flag in the `stored_groups_descriptor` is set to "1".

11.17.8 Stored Group Deletion

Groups shall be removed either:

- 1) When an update to the group is signalled and the new version of the group is being acquired.

OR

- 2) When receiver resources are limited and a request to store a different group of higher priority is made. In this instance the memory used by the lower priority group(s) shall be relinquished and the associated reference to the stored group(s) shall be removed from cache.

It shall also be possible to relinquish memory allocated to stored groups by explicit signalling methods. This can be achieved by signalling a standard group version change in the stored_groups_descriptor and associated manifest file. The manifest shall signal that the group size is zero bytes and shall describe an empty nodes section. Any previously used memory by the files in the group defined shall be relinquished.

If the receiver implements a factory reset feature and it is invoked this shall reset the File System Acceleration cache.

11.18 Application Launch Extension

11.18.1 GetEngineSupport 'feature' strings

Table 11.76: GetEngineSupport

String		Constraint
Standard	Short	
ApplicationLaunchExtension(N)	ApE(N)	Shall return "true" for N=0 if the receiver supports ApplicationLaunchExtension. Shall return "true" for N=-1 if the receiver supports ApplicationLaunch() from the ApplicationLaunchExtension. For other values of N, shall return "true" if the receiver supports launching applications whose MIME type=N. Receivers shall consider that MIME type strings are case insensitive when determining if a MIME type is supported.

11.18.2 Resident Programs

11.18.2.0 List of resident programs

Table 11.77 lists the ResidentPrograms that receivers implementing *ApplicationLaunchExtension* shall implement.

Table 11.77: Mandatory Resident Programs for receivers that implement ApplicationLaunchExtension

Resident program		Invocation			Reference
		Typical Use		Never Fork	
Description	Name	Call	Fork		
ApplicationLaunch	ApL	✓			11.18.1.2.1 "ApplicationLaunch"
GetLaunchArguments	GLA	✓			11.18.1.2.2 "GetLaunchArguments"

11.18.2.1 ApplicationLaunch

Synopsis: Hands control of execution to another application of an arbitrary type.

Arguments: ApL (location, [name, value]..., success).

Table 11.78

in/ out/in-out	type	name	comment
input	GenericOctetString	location	Location of the application to run.
input	GenericOctetString	name	List of name/value pairs to be passed to the application.
input	GenericBoolean or GenericInteger or GenericOctetString	value	
output	GenericBoolean (Shall provide an IndirectReference to a BooleanVariable)	success	True if the application started successfully, false otherwise.

Description: Causes a new application to be started with the specified arguments. The application to run is specified by the location parameter.

A side-effect of the resident program may be that the MHEG engine is stopped (killing the application). In all cases the state of the True Persistent Storage shall not be affected.

The resident program takes a variable number of arguments. Zero or more name/value pairs may be present. If any name/value pairs are present, the name/value pairs are used to construct a data set of content type *application/x-www-form-urlencoded* as specified by section 17.13.4 of HTML 4.01 [26] except that references to IETF RFC 1738 [27]. shall be taken as references to IETF RFC 3986 [28], which updates it. Given "name1%" = "value1" and "name2" = "contains spaces", this produces a data set of the form *name1%25=value1&name2=contains+spaces*, where each of the names and values has been percent-encoded, except that space characters are replaced with '+'.

The data set may contain characters that are not represented in the US-ASCII character set; consequently the percent-encoding shall be carried out as specified for characters from the Universal Character Set in section 2.5 of IETF RFC 3986 [28]. Characters are assumed to be encoded as UTF-8. For example, the character Latin Capital Letter A With Grave is represented in UTF-8 by the octets 0xC380. In the text representation of MHEG, this character would be written as '=C3=80'; after percent-encoding this would become "%C3%80".

The data set is appended to the location, with a '?' character as a separator; this forms a URI which references both the application to launch and the parameters to be passed to it.

GenericOctetString arguments are treated directly as strings. GenericInteger arguments are converted to strings as decimal integers with no leading zeros. GenericBoolean arguments are converted to the string "true" if true and to "false" if false.

In any case where an invalid set of arguments is supplied (such as a missing value argument) the resident program call shall fail in accordance with clause 11.10.12. It shall not be possible to launch an MHEG Application using this resident program.

The location URI shall:

- Follow the rules for use of reserved characters in HTTP URIs according to clause 16.3.2.5.
- Be either a HTTP, HTTPS or dvb: URL identifying an application.

11.18.2.2 GetLaunchArguments

Synopsis: Retrieves an argument set by another application of an arbitrary type.

Arguments: GLA(name, value)

Table 11.79

In/out/in-out	type	name	comment
input	GenericOctetString	name	Name of argument to be retrieved
output	GenericOctetString (shall provide an IndirectReference to an OctetStringVariable)	value	Value of argument

Description: Retrieves the value of the named argument. Arguments can be set by applications of an arbitrary type, other than MHEG, when launching an MHEG application.

The value of the argument is provided to the application as an OctetStringVariable. It is the responsibility of the application to convert this value into another type (using the SetVariable elementary action or one of the type conversion resident programs) if required.

If the argument to be retrieved does not exist then the resident program succeeds and the value parameter is a zero length string.

11.18.3 Referencing MHEG Applications from other Presentation Technologies

An MHEG application may only be referenced from another non-MHEG Presentation Technology with one of the following identifiers:

- A DVB URI referencing an AIT.
- A URI referencing an XMLAIT, where the XMLAIT contains a DVB URI referencing the MHEG initial object. For example "dvb://current" contained in a TextualId element in a OCTransportType (see clause 5.4.4.21 of ETSI TS 102 809 [3]).

For AIT and XMLAIT references the application type field is encoded in the table 9.1.

MHEG applications cannot be launched from another Presentation Technology using URIs with a scheme of "http" or "https" to reference the MHEG initial object since the Interaction Channel security model requires the initial application to be launched from a broadcast carousel. URIs with a scheme of "http" or "https" may be used to reference an XMLAIT as described above.

When an MHEG application is launched from another Presentation Technology, any launch arguments shall be available to this initial application via the GLA resident program as defined by clause 11.18.2.2. Launch arguments shall not be available to any subsequent application Launched or Spawned from the initial application. Launch arguments shall be available to the initial application when any spawned applications quit. Where XMLAIT is used the launch arguments available to the MHEG application shall include all of those defined by the launching Presentation Technology in addition to arguments defined within the XMLAIT.

A receiver shall accept URLs of up to and including 2 048 bytes in length.

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.)

The Desktop defined by MHEG-5 shall be visible at the bottom of the display stack wherever there is no graphics plane object or other visible overlapping it. It can be set to any single RGB colour, where the colour format has at least the same colour depth as the graphics plane. Its default setting shall be black (i.e. opaque, with the red, blue and green components all zero).

- The drawing area available for applications has pixel dimensions of 720 x 576.
- Visible area: see clause 17.4.2.
- Graphics/video pixel alignment: see clause 14.4.4.
- 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

12.2.0 Minimum colour support

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.

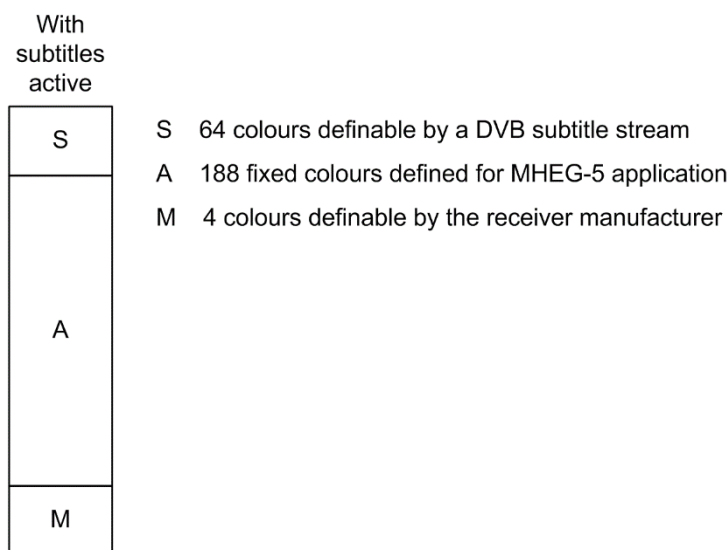


Figure 12.1: Colour palette division between concurrent processes

12.2.1 Reservation for MHEG-5 applications

12.2.1.0 Reserved locations

188 locations (the "A" palette in figure 12.1) 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 12.1 defines the colour combinations in the "A" palette.

Table 12.1: "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 12.2 illustrates the opaque colours in the palette.

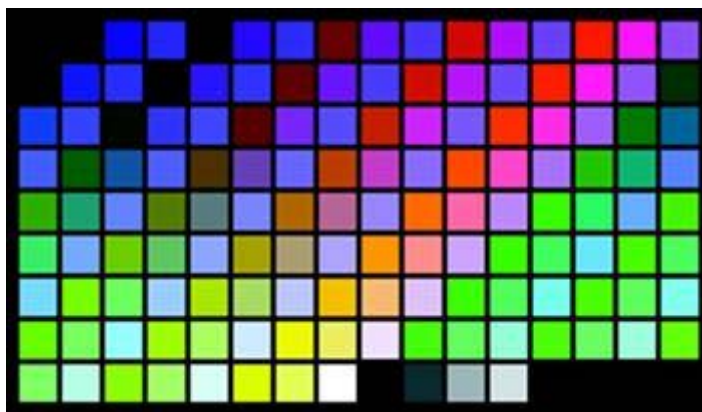


Figure 12.2: "A" palette (showing opaque colours only)

12.2.2 Reservation for DVB subtitles

64 locations (the "S" palette in figure 12.1) 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 12.1) are reserved for receiver manufacturer use.

12.3 Colour representation

12.3.1 Colour space

The engine shall be responsible for converting between colour spaces.

Depending on the content type the MHEG-5 engine handles colours in both RGB (colour for buttons, text etc. and PNG graphics) and YCrCb (MPEG stills and DVB subtitles) colour spaces (see figure 12.3 and figure 12.4).

Broadcasts shall use colorimetry as defined by Recommendation ITU-R BT.470-7 [22]. This defines the relationship between RGB and Y-Cr-Cb, which shall be used wherever a transformation from one representation to the other is necessary.

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.

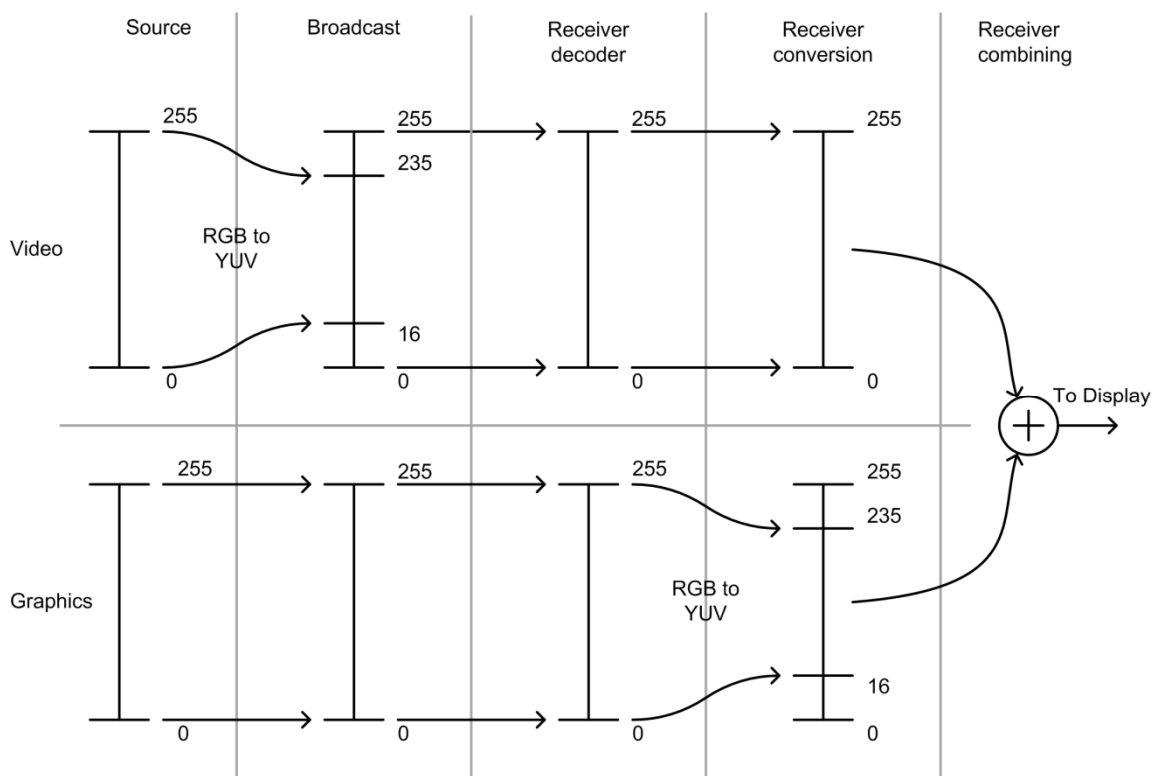


Figure 12.3: A receiver with video and graphics combined in YUV form

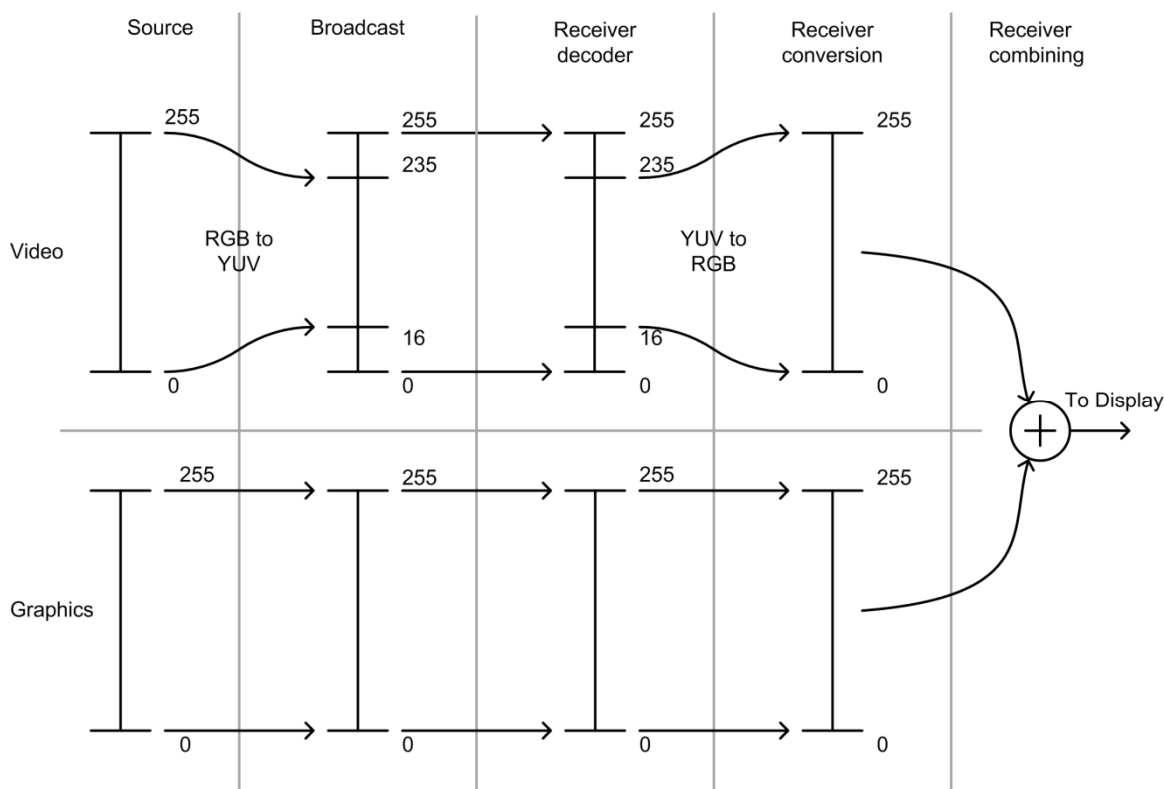


Figure 12.4: A receiver with video and graphics combined in RGB form

12.3.2 Gamma

MPEG video and DVB subtitles deliver YCrCb data in accordance with Recommendation ITU-R BT.601 [24]. 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 precorrection 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 12.5.

First						Last					
7	Red	0	7	Green	0	7	Blue	0	7	Transparency	0

Figure 12.5: 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 shall 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 semitransparent 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 shall not be allowed to "punch through" opaque pixels (see figure 12.6).

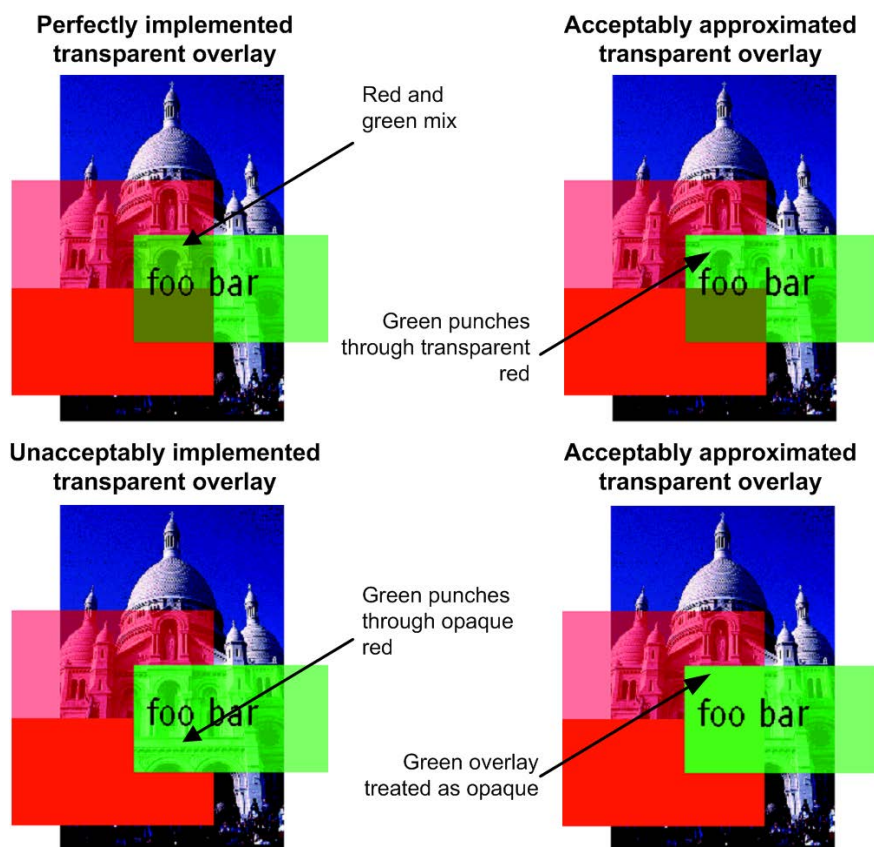


Figure 12.6: Approximation of transparency

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 semitransparent 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 shall 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 12.7).

`DynamicLineArt` objects with `BorderedBoundingBox` set `False` have no border and drawing operations on such objects clip only at the edges of the object.

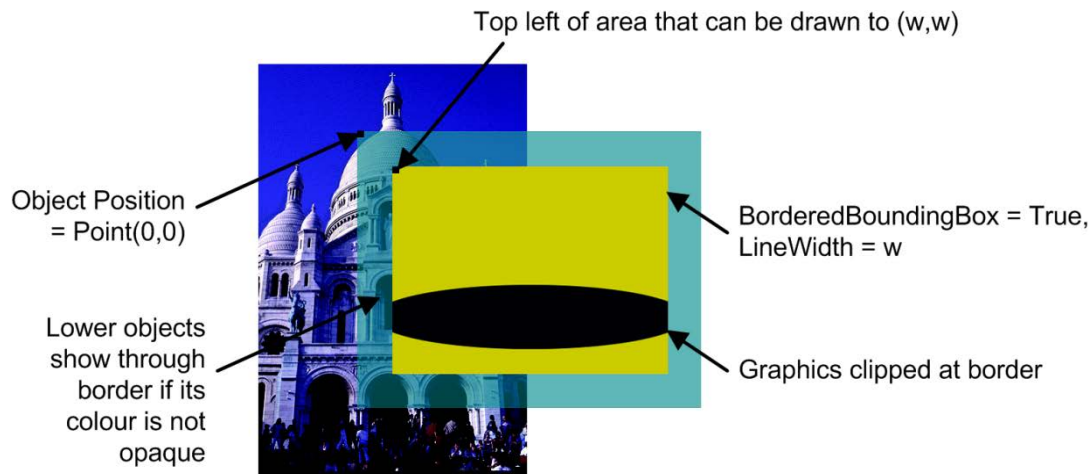


Figure 12.7: Graphics clipped at border

12.5.1.5 DrawSector

Figure 12.8 illustrates the results of a `DrawSector` action where `RefLineColour` is green and `RefFillColour` is yellow.

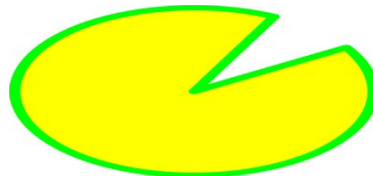


Figure 12.8: DrawSector illustrated

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 alpha mixing shall not be 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 shall be 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).

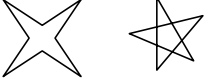

12.5.1.7 Co-ordinate system

The co-ordinates used in `DynamicLineArt` drawing operations address the corners of display pixels such that the point $(0, 0)$ represents the top left-hand corner of the object and not the centre of the first pixel. A filled rectangle drawn from the point $(0, 0)$ to the point $(4, 4)$ with no line width is therefore four pixels square.

12.5.2 Limitations

The allowances in table 12.2 are made to assist receiver implementation. Authors should take account of the implied authoring constraints.

Table 12.2: 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	<p>The receiver behaviour when filling certain shapes is not defined. These shapes are:</p> <ul style="list-style-type: none"> • concave polygons; and • self-crossing polygons. <p>All other shapes shall be completely filled with the colour defined by the RefFillColor attribute.</p>	<p>Avoid using filled concave polygons and filled self-crossing polygons, i.e. avoid filling shapes such as:</p>  <p>If shapes such as these are required to be filled they may be constructed from primitive elements such as triangles which are guaranteed to fill in a predictable way.</p>
Self-crossing polygons and polylines	The appearance of pixels at the junction of self-crossing lines is not defined.	<p>Avoid self-crossing lines such as:</p> 

12.6 Text, EntryFields and HyperText

Text objects (and the text areas in EntryField and HyperText objects) shall be 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 the present document apply to these layers (see clause 12.4.1).

Therefore, the "paper" colour (the BackgroundColour of the Text) (if any) shall be 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) shall be 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 12.3. ETSIEngineProfile1 decoders shall ignore any additional chunks.

Table 12.3: 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	
tRNS	
cHRM	Receiver shall skip.
gAMA	Receiver shall skip.
sBIT	Receiver can skip.
bKGD	Receiver shall skip.
hIST	Receiver can skip.
pHYs	This field shall be ignored by receivers that do not support HDGraphicsPlaneExtension. Such receivers shall render all images pixel for pixel into the graphics plane. Receivers supporting HDGraphicsPlaneExtension (see clause 12.11.1.1) shall consider all images to be of SD resolution unless they contain a pHYs chunk with a unit specifier of '1' and one of the following pixel resolutions: <ul style="list-style-type: none"> • 1 280 x 720 images: 3 543 pixels per metre (90 dpi). • 1 920 x 1 080 images: 5 315 pixels per metre (135 dpi).
tIME	Receiver shall skip.
tEXt	Receiver shall skip.
zTXt	Receiver shall skip.

See also clause 11.5.2.

12.7.2 Colour encoding

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

See table 12.4.

Table 12.4: 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 shall 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 shall be 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 the present document 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.

Receivers supporting HDGraphicsPlaneExtension shall use the pHYs chunk of the PNG image data to determine how to render the image, as specified in clauses 12.7.1 and 12.11.3.3.

12.8 MPEG-2 stills

12.8.1 File format

The payload of a file delivering an MPEG-2 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 shall be 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, 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.

Text objects shall also be displayed as an empty Text object while any referenced Font data is being retrieved.

12.11 High definition graphics model

12.11.0 Requirements for support

High definition receivers (i.e. ones that are capable of decoding and presenting HD resolution video) shall observe the standard Graphics Model described in clauses 12.1 to 12.10 with the exceptions specified in the following clauses. Support for decoding, managing and presenting HD resolution video under MHEG control is hereafter referred to as HDVideoExtension.

12.11.1 Resolution

12.11.1.0 Resolution support requirements

Support for an HD resolution graphics plane as described in clause 12.11.1.1 is hereafter referred to as HDGraphicsPlaneExtension.

Receivers claiming support for HDGraphicsPlaneExtension shall also have all of the following:

- an HD output (e.g. HDMI);
- support for SD and HD resolution JPEG and PNG images;
- support for the square text style and 20 and 22 point text.

The resolution of the physical graphics plane for MHEG applications shall be independent of the resolution of any video being presented and shall be fixed for at least the lifetime of the running MHEG application.

NOTE: In practice this resolution is likely to be dependent on the configuration of the HDMI output.

12.11.1.1 HD resolution graphics plane

12.11.1.1.0 Scope

For a receiver to support HDGraphicsPlaneExtension (an HD resolution graphics plane compliant with this profile) the following shall be observed.

12.11.1.1.1 Resolution

The graphics plane resolution shall be at least 1 280 x 720 pixels, with the recommended resolution being 1 920 x 1 080 pixels. Other resolutions between these two limits are also permitted but it shall be noted that applications can only provide HD images in one of those two resolutions and so image rendering would need to support additional scaling factors and picture quality could be adversely affected.

NOTE: If the SCART output (as opposed to the HDMI output) is configured by the viewer for use as the primary output (where such a feature is provided) the graphics plane resolution may be reduced to 720 x 576 pixels if this provides a superior visual output compared to down-conversion of an HD resolution graphics plane.

12.11.1.1.2 Colour range

The graphics plane shall provide an RGB colour space with at least 16 bits per pixel. At least 4 bits shall be used for each of the red, green, blue and transparency components. The graphics plane shall provide at least 16 evenly spaced levels of transparency.

12.11.1.1.3 Direct/absolute colours

Colour definitions within MHEG-5 Visible are encoded with 8 bits per component. Bitmap content may use a variety of different bit depths. Where the graphics plane supports fewer bits per channel, colours and transparency values shall be mapped by dropping the least significant bits. Where this contradicts anything stated in clause 12.3.4, the present clause shall take priority.

12.11.1.1.4 Text rendering

Text shall be rendered in the manner described in clause 13.5.1.

12.11.1.1.5 Bitmap format and resolution

Bitmap images encoded using the PNG format as described in clause 12.7 and the JPEG images as described in clause 12.12 shall be supported at both SD and HD resolutions.

12.11.2 Mapping the MHEG application co-ordinate system to the graphics plane

MHEG applications use a 720 x 576 co-ordinate system for describing all graphics presentation, video scaling and positioning. This logical co-ordinate system is used to describe services for presentation on both SD and HD capable receivers.

The 720 x 576 MHEG co-ordinate system shall describe an area that maps to the entirety of the supported graphics plane, even when the resolution of the graphics plane is different. Further, the entirety of the supported graphics plane shall map to the entirety of the video plane, even when the resolutions of the two are different.

NOTE: A consequence of this is that an existing MHEG application authored using the 720 x 576 co-ordinate system for presentation as part of an existing SD resolution service can also be used for presentation as part of a HD resolution service.

Where the resolution of the MHEG application co-ordinate system differs from that of the supported graphics plane the act of mapping the application description into a rendered graphics output shall involve a process of "intelligent rendering" (see clause 12.11.3). The mapping from MHEG application co-ordinate system to supported graphics plane shall be fixed for at least the lifetime of the running MHEG application such that the visual presentation of graphics aspects of an MHEG application does not vary. This shall be the case even when the resolution of any video being presented changes, e.g. from SD to HD or vice-versa, as a result of an application-initiated change of video source.

12.11.3 Intelligent rendering

12.11.3.0 Scope

This clause applies only to receivers supporting HDGraphicsPlaneExtension.

12.11.3.1 Introduction

12.11.3.1.0 Overview of process

The process of intelligent rendering involves:

- Transforming the bounding box of each MHEG Visible object within the 720 x 576 co-ordinate system used to describe an MHEG application into an HD co-ordinate system that maps one-to-one with the resolution of the supported graphics plane.
- Rendering the visual appearance of each MHEG Visible object directly into the supported graphics plane reflecting the transformed bounding box.

12.11.3.1.1 Co-ordinate transformation

The transformation from SD co-ordinates to HD co-ordinates shall be performed by multiplying the original SD co-ordinate with the ratio of HD to SD dimensions, and then rounding down the result.

The transformation from an SD horizontal co-ordinate x_{SD} to an HD horizontal co-ordinate x_{HD} shall be:

$$x_{HD} = \left\lfloor x_{SD} \cdot \frac{xres_{HD}}{xres_{SD}} \right\rfloor \quad (2)$$

The transformation from an SD vertical co-ordinate y_{SD} to an HD vertical co-ordinate y_{HD} shall be:

$$y_{HD} = \left\lfloor y_{SD} \cdot \frac{yres_{HD}}{yres_{SD}} \right\rfloor \quad (3)$$

These transformations ensure that the extreme values of the SD co-ordinate system (e.g. 0, 0 and 720, 576) map to the extreme values of the HD co-ordinate system (e.g. 0, 0; and 1 920, 1 080 for a 1 920 x 1 080 graphics plane).

Where objects "touch" in the MHEG-5 application, then platforms shall ensure that the objects "touch" when displayed, i.e. no gaps are allowed between adjacent objects. If such objects are the same colour, no unspecified "invented" colours are allowed at the boundary between the objects. The mechanism by which these constraints are achieved is by transformation of the Bounding box as described in clause 12.11.3.2.

12.11.3.2 Bounding box transformation

The rectangular nature of the bounding box for MHEG Visible makes the transformation relatively straightforward. However, since the mapping from the MHEG application co-ordinate system and any HD resolution graphics plane does not involve convenient multiples, the mapping is not linear.

The transformation of a Visible object's bounding box shall be based on separate transformations of the top-left and bottom-right co-ordinates. This method of transformation preserves the author's intention that touching objects in the SD co-ordinate system should touch in the HD co-ordinate system. Receivers shall ensure that where objects of the same colour touch, no other colour is visible at the boundary.

EXAMPLE: In figure 12.9, the yellow object has an SD position of (1, 0) and a box size of (2x2). The transformation is applied to the top left and bottom right co-ordinates, (1, 0) and (3, 2) so that the resulting object on a 1 920 x 1 080 graphics plane has a top-left co-ordinate of (2, 0) and a box size of (6x3). The blue object is mapped in the same way and has an HD position of (8, 0) and a box size of (5x3).

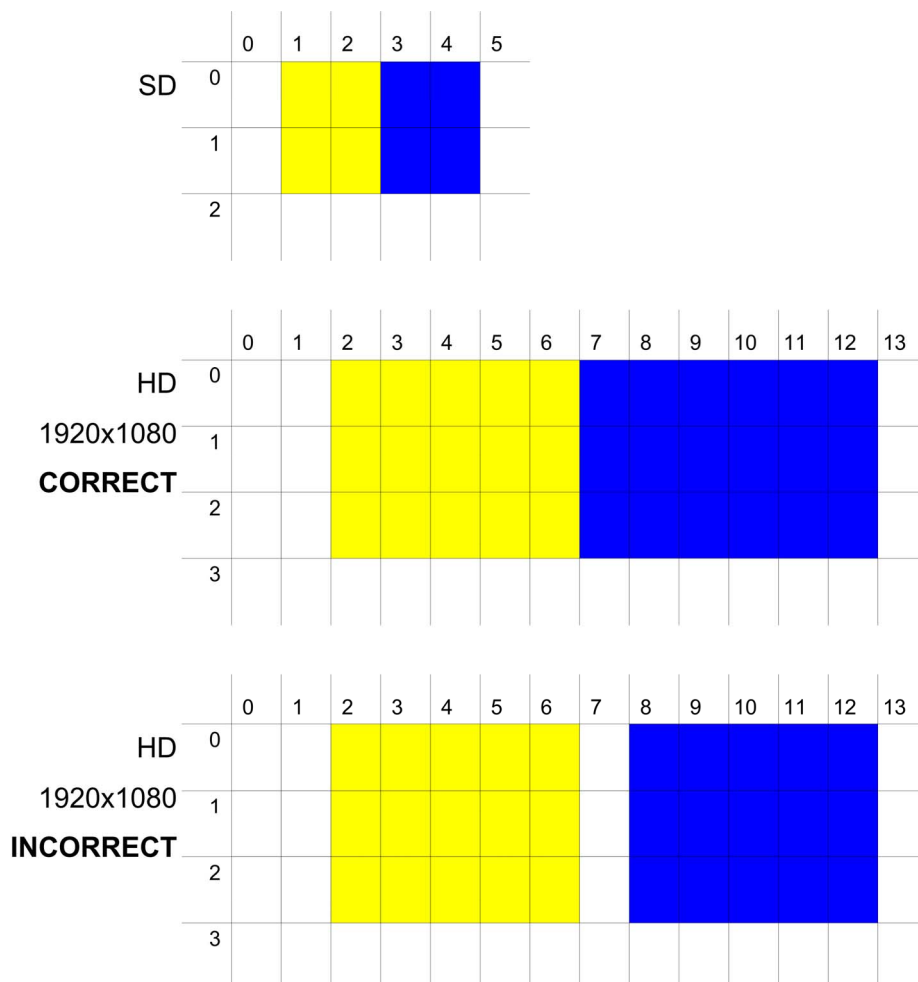


Figure 12.9: Bounding box transformation

The two objects touch in the HD graphics plane, just as they did in the SD co-ordinate space but they have different sizes in the HD graphics plane. The correct rendering of these objects on a 1 920 x 1 080 graphics plane is shown in the second picture of figure 12.9. The third picture of figure 12.9 shows the effect of separately mapping the position and the box size, resulting in incorrect behaviour with a gap between the objects.

12.11.3.3 Visual appearance

12.11.3.3.1 Text

Each line of text shall be rendered directly into the HD resolution graphics plane. Constraints on the renderer used are defined in clause 13.5.1.

In order that each line of text occupies the same proportion of the screen, the font size as described in the MHEG application shall be scaled before being passed to the text renderer. This scaling will vary depending on the resolution of the graphics plane as described below.

Receivers supporting HDGraphicsPlaneExtension shall support an additional two font sizes smaller than those required for an SD receiver. The complete list of font sizes required and the corresponding vertical point sizes for rendering on HD resolution graphics planes are shown in table 12.5.

Table 12.5: HD resolution text appearance

Informative name	Font size at 720 x 576	Point size equivalent at 1 280 x 720	Point size equivalent at 1 920 x 1 080
Heading/Large subtitle	36	45	67,5
Subtitle	31	38,75	58,125
Body	26	32,5	48,75
Footnote	24	30	45
HD22	22	27,5	41,25
HD20	20	25	37,5
NOTE 1: If the precise point sizes specified cannot be realized, receivers may round down to the nearest integer point size.			
NOTE 2: For other resolutions, the HD font size can be calculated as:			
$size_{HD} = \frac{yres_{HD}}{yres_{SD}} size_{SD} \quad (4)$			

Text rendered on SD displays shall be scaled to give a compromise 14:9 presentation that gives acceptable results on both 4:3 and 16:9 displays. For backwards compatibility, this scaling shall also be performed by default when rendering onto HD resolution graphics planes. However, the square pixel nature of the HD resolution graphics planes means that the required conversion from points to pixels in the horizontal dimension differs from that used when rendering into the SD graphics plane. Whereas receivers assume that SD pixels have a width of 56/45 points (see clause 13.5.3), receivers shall assume that HD pixels have a width of 7/8 points. The consequence of this is that the text rendered on a 16:9 HD display shall, by default, have the same aspect ratio as text rendered onto a 16:9 SD display.

Receivers supporting HDGraphicsPlaneExtension shall also support rendering of text without 14:9 scaling. This option is invoked by the use of the "square" style in the FontAttributes string (see clause 13.4.1).

12.11.3.3.2 Images

Images encoded at SD resolution shall be scaled during rendering to an HD graphics plane using appropriate scale factors such that an image that is the same size as its SD bounding box fills the transformed HD bounding box.

If an image is encoded at an HD resolution it may still need to be scaled up or down depending on the resolution of the HD graphics plane provided by the receiver.

Bitmap objects with the Tiling attribute set to "true" shall be scaled and rendered such that there are no gaps or disturbances in the rendering at the boundaries of the tiles.

12.11.3.3.3 Line Art

The SD-specified line width needs to be transformed. This shall be performed using the rounded down mean of the vertical and horizontal scaling factors.

12.11.3.3.4 DynamicLineArt

A receiver may render dynamic line art at the SD resolution and rescale as if it were an SD image being upscaled to a HD resolution.

However, for a better quality of presentation, a receiver may render dynamic line art directly into the HD resolution graphics plane. This option is preferred.

If the receiver renders dynamic line art at the HD resolution, the co-ordinates of each vertex in the SD co-ordinate system shall be separately transformed into HD co-ordinates using the equations in clause 12.11.3.1. For shapes based on ellipses, the start and end angles shall also be transformed as follows:

$$\theta_{HD} = \arctan \left\{ \frac{xres_{SD} \cdot yres_{HD}}{xres_{HD} \cdot yres_{SD}} \tan \theta_{SD} \right\} \quad (5)$$

The transformation shall be applied separately to the start and end angles and not to the arc angle. Also note that the value of θ_{HD} will become $-\theta_{HD}$ in the second and fourth quadrants.

There are no special "touching" requirements for the rendering of DynamicLineArt objects since all touching elements described necessarily touch after the transformation.

12.12 JPEG bitmaps

JPEG images shall be encoded in conformance with ISO/IEC 10918-1 [47] using the JFIF file exchange format [48].

Images shall only be coded using the sequential DCT-based mode and Huffman coding. Progressive DCT-based, lossless and hierarchical modes and images using arithmetic coding need not be supported and shall not be broadcast. Any thumbnail images present shall be ignored.

Receivers that do not support HDGraphicsPlaneExtension shall ignore pixel aspect ratio and resolution information in the JFIF APP0 marker. Such receivers shall render all images pixel for pixel into the graphics plane.

Receivers supporting HDGraphicsPlaneExtension (see clause 12.11.1.1) shall consider all images to be of SD resolution unless they contain an APP0 marker with a units specifier of '1' or '2' and one of the following pixel resolutions encoded in the Xdensity and Ydensity fields:

- 1 280 x 720 images: 35 dots per cm or 90 dots per inch.
- 1 920 x 1 080 images: 53 dots per cm or 135 dots per inch.

12.13 H.264/AVC stills

12.13.1 File format

The payload of a file delivering an H.264/AVC [39] I frame shall:

- be a valid video sequence;
- contain one I frame;
- be formatted as a raw byte stream;
- be flagged as a full_frame_snapshot (receivers shall ignore the snapshot_id).

I.e. an IDR access unit, with intra coded slice data and a sequence end.

The structure is:

- 1) IDR access unit delimiter.
- 2) Sequence Parameter Set (SPS).
- 3) Picture Parameter Set (PPS).
- 4) Supplemental enhancement information (SEI).
- 5) Coded slice of IDR picture.
- 6) End of Sequence.

The file shall contain neither SEI pic_timing messages nor SEI buffering_period messages (see annex D of H.264/AVC [39]).

12.13.2 Semantics

An H.264/AVC video decoder conforming to the same behaviour as the main video decoder shall be used to decode the fragment of data containing the I-frame.

12.13.3 Presentation

See clause 14.4.4 for description of the presentation of H.264/AVC stills.

13 Text and interactibles

13.1 Text rendering overview

13.1.0 Application of text rendering rules

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 13.1.

Table 13.1: Application of rules to classes

	Character encoding (see clause 13.2)	FontAttributes (see clause 13.4.1)	Text rendering (see clause 13.5)	Text mark-up (see clause 13.6)	HyperText mark-up
Text	✓	✓	✓	✓	
EntryFields (see note 1)	✓	✓	✓		
HyperText (see 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 11.1) no processing shall be 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

13.2.0 Character encoding format

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 13.2 reproduces the UTF-8 coding scheme. The character repertoire defined in the Tiresias™ Screenfont [46] only requires 1-byte, 2-byte or 3-byte codes. Where text is in English, Welsh or Gaelic, the majority of characters are coded on 1 byte.

Table 13.2: UTF-8-bit distribution

ISO/IEC 10646 [13] value	1 st byte	2 nd byte	3 rd byte	4 th 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 +	1111 0uuu	10uu zzzz	10yy yyyy	10xx xxxx
1101 11yy yyxx xxxx	(see note)			
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 13.3 identifies the minimum set of values of CharacterSet (as used in the Application and Text classes) that the engine shall support.

Table 13.3: Engine CharacterSet attributes

Attribute	Character set
< 10	Reserved for future use or other application domains.
10	The subset of ISO/IEC 10646 [13] available within the font.
> 10	Reserved for future use.

13.3 Fonts

13.3.1 Downloading

13.3.1.0 Application of downloadable fonts

Receivers implementing DownloadableFontExtension shall support downloadable fonts using the MHEG-5 Font class; other receivers shall not support downloadable fonts or the Font class.

Application references to non-downloadable fonts shall be direct (i.e. an OctetString representing the name of the font). Text objects referencing downloadable fonts shall do so using a reference to an MHEG-5 Font object. Downloadable fonts cannot be referenced by name.

13.3.1.1 OpenType fonts

13.3.1.1.0 Application of OpenType fonts

Receivers implementing DownloadableFontExtension shall support OpenType® fonts with TrueType™ outlines as defined by the Open Font Format specification [43].

OpenType fonts are identified using a CHook value of 10.

13.3.1.1.1 Profile of OpenType

Receivers implementing DownloadableFontExtension shall support the 'required' tables, the 'tables related to TrueType outlines' and the 'kern' table (format '0' horizontal kerning only) of the Open Font Format specification [43]. Support for other tables is optional. Application authors are advised to avoid using fonts that make use of optional tables as these may be rendered differently by different receivers.

Receivers may support TrueType Collections. Application authors shall not use TrueType Collections; in the present document, one MHEG-5 Font object provides a single font.

13.3.1.1.2 Font parameters

For OpenType fonts, table 13.4 defines the values to be used for the font metrics parameters referenced in clause 13.5.

Table 13.4: OpenType font parameters

Parameter name	Obtained from
metricsResolution	unitsPerEm field, defined in the Font Header ('head') table
outlineResolution	
advanceWidth, charSetWidth	advanceWidth values, defined in the Horizontal Metrics ('hmtx') table (see note)
xMin, yMin, yMax	xMin, yMin, yMax, defined in the Font Header ('head') table
kern	value, defined in the Kerning ('kern') table
NOTE: For monospaced fonts, only a single advance width may be defined.	

13.3.1.1.3 Text Styles

When referencing a downloaded font, the 'plain' text style (see clause 13.4.1) shall refer to the single text style present in the referenced font file, whatever that may be. If the receiver supports HDGraphicsPlaneExtension, the 'square' text style shall also refer to the single text style present in the referenced font, but without 14:9 aspect ratio correction, as described in table 13.6.

13.3.1.2 Presentation

If the receiver implements DownloadableFontExtension, Text, EntryField and HyperText objects referencing a downloaded font shall not be rendered until the receiver has downloaded the required font data. Until then, the object shall be presented as defined in clause 12.10.

Where an attempt to download content for an MHEG-5 Font object fails, any Text objects dependent on it shall be rendered in the receiver's in-built font.

13.3.1.3 Defensive response

All receivers shall implement the following measures to ensure robust behaviour with any applications that attempt to use in-built or downloadable fonts that are not available, or font characteristics that the receiver does not recognize:

- If the font requested by a Font object is not available, the receiver shall use its in-built font.
- If any of the attributes of the Font object are invalid, e.g. an unsupported content hook, the receiver shall use its in-built font.
- If the font style of a Text object is recognized then it shall be used; if not, the style referenced by 'plain' shall be used.
- Where a font contains a limited range of available sizes or the receiver can only display a limited range of sizes, if the requested font size of a Text object does not match one of those available, the receiver shall substitute the next smaller size available. If the required font is smaller than the smallest available, then the smallest available size shall be used.
- Any character not supported by the engine shall not be considered as part of the input.

13.3.1.4 Font resource model

Receivers shall be able to support the simultaneous presentation of at least 4 downloaded fonts which are comparable (in terms of both number and graphical complexity of characters) to the in-built font, together with the in-built font.

13.3.2 Embedded font

13.3.2.1 The DTG/RNIB font characteristics (informative)

Tiresias™ Screenfont [46] is a kerned sans-serif (like Helvetica or Arial) font designed to look good on both 4:3 and 16:9 displays.

NOTE: Tiresias™ is the trade name of a product supplied by Monotype Imaging Inc. and owned by the Royal National Institute of the Blind. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of the product named.

13.3.2.2 Font version

When the font is referenced as "rec://font/eu1" receivers shall use the metrics and character repertoire defined in v8.04 of the Tiresias™ Screenfont [46]. When the font is referenced as "rec://font/uk1" the metrics xMin, xMax, yMin and yMax shall take the values defined in v7.51 of the Tiresias™ Screenfont font [54].

It is recommended that any receiver implementing 1-bit/pixel rendering use the bitmaps also contained in the 7.51 [54] release.

The bitmaps contained in v7.51 [54] 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).

13.3.2.3 Required sizes and styles

Receivers shall implement the built-in font in at least the sizes identified in table 13.5.

Table 13.5: "UK1" sizes and styles

Size (points)	TV lines over "Cap-V" (see note 1)	Informative name	Styles	
			Plain	Square
36	24	Heading/large subtitle	✓	✓ (see note 3)
31	21	Subtitle	✓	✓ (see note 3)
26	18	Body	✓	✓ (see note 3)
24	16	Footnote	✓ (see note 2)	✓ (see note 3)
22	15	HD22	✓ (see note 3)	✓ (see note 3)
20	14	HC20	✓ (see note 3)	✓ (see note 3)

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).
NOTE 3: Required only in receivers supporting HDGraphicsPlaneExtension.

13.3.3 Invoking the font

The font shall be accessible using two different names (that is, the FontName for an Application object Font attribute or the OriginalFont attribute of the Text class). When the font is invoked using the name "rec://font/eu1" the engine shall render characters from the font using its default font metrics. Where the font is invoked using the name "rec://font/uk1" the engine shall render characters using metrics from that font except for the values xMin, xMax, yMin, yMax which shall use the values defined for v7.51 of Tiresias™ Screenfont [54] (see table 16.1).

The default font for engines conforming to ETSIEngineProfile1 shall be "rec://font/uk1" (see table 16.1). Engines shall render any character available in the required font. This means that a Text object with a font reference of "rec://font/uk1" may render characters that are in v8.04 of the font but are not present in v7.51 [54].

13.4 Text object attributes

13.4.1 FontAttributes

13.4.1.0 FontAttribute formats

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.

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 13.6.

Table 13.6: Long form text format parameters

Field	Set of allowed values	Meaning
style	'plain'	Plain text.
	'square'	Text to be rendered without correction for a 14:9 aspect ratio (required only in receivers supporting HDGraphicsPlaneExtension).
size	'20' '22' '24' '26' '31' '36'	Font size in points as decimal integer strings.
linespace	'0' to '255' (see note)	Space between the baselines of adjacent lines of text in points as decimal integer strings.
letterspace	'-32767' to '32767' (see 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 13.7.

Table 13.7: Short form text format parameters

Syntax	Bits	Type	Allowed values
style	8	bslbf	See table 13.8
size	8	uimsbf	0x14, 0x16, 0x18, 0x1A, 0x1F or 0x24
linespace	8	uimsbf	0 to 255
letterspace	16	tcimsbf	-32 767 to 32 767

style: An 8-bit string coded as shown in table 13.8.

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 13.8: Coding of "style"

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		
\square	\circ	ϵ	\circ	σ	σ	σ	σ	Plain	" " "0" "@"
x	x	x	0	0	0	0	0	Square	'0' 'P'
x	x	x	1	0	0	0	0		

NOTE 1: Only receivers implementing HDGraphicsPlaneExtension need to support the 'Square' style.

NOTE 2: Italic, Bold, Underline and Outline are not supported in the present document.

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 13.9.

Table 13.9: Required set of text flow modes

Attribute	Required values	Optional flow modes		Notes
		Optional value (see note 1)	Replacement alternative (see 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 17.9.2.

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.

For a receiver that does not support the HDGraphicsPlaneExtension, 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.

For a receiver supporting HDGraphicsPlaneExtension it is recommended that text be rendered using anti-aliasing with at least 8 levels to be mapped to colours between the relevant Text object's TextColour and BackgroundColour. This recommendation shall apply at all times, i.e. regardless of the resolution of any video being presented.

NOTE 1: If the Text object's BackgroundColour is transparent or partially transparent, the intermediate colours for anti-aliasing are also partially transparent.

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:
 - having correctly applied the rules relating to the flow of text, the length of the line of text to render can be wider than the available width for rendering (from Step 1); for instance where there are no suitable wrapping points in the line or when `TextWrapping` is set to `False` (see 13.5.6). 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 2: The order of these steps is illustrative and may vary between implementations for reasons of convenience or efficiency.

13.5.2 Font definition

13.5.2.0 Introduction

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 (<http://www.bitstream.com>).

Receivers need not embed font data using this format; 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 13.1).

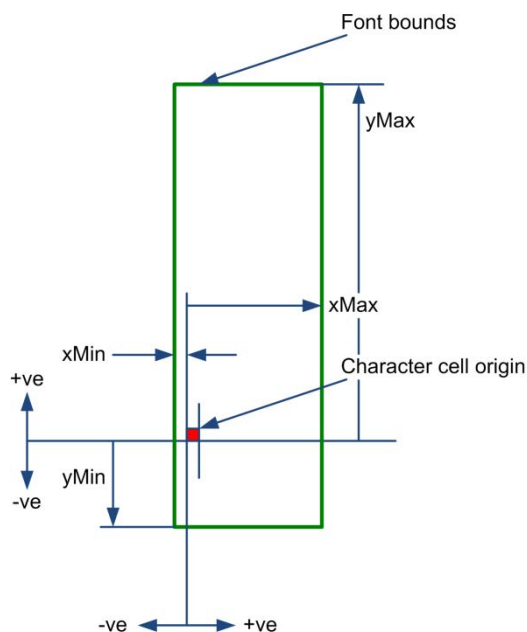


Figure 13.1: Font bounds

In the present document these parameters shall be 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

13.5.3.0 Approach

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 shall be preserved for the vertical dimension of the SD MHEG-5 display (i.e. each of 576 SD OSD lines is considered to be 1 point high). When rendering on HD resolution graphics planes, one line is again considered to be 1 point but text sizes are scaled up as described in clause 12.11.3.3.

13.5.3.2 Horizontal resolution

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

To simplify the ETSIEngineProfile1 all receivers shall assume that each of the 720 SD 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.

For compatibility with SD receivers, receivers supporting HDGraphicsPlaneExtension shall also render text in the 'plain' style with the appropriate correction required for 14:9 text rendering. However, due to the square pixel nature of HD graphics planes, the correction required is a factor of 7/8.

Receivers supporting HDGraphicsPlaneExtension shall render text in the 'square' style without applying any correction for 14:9 text rendering.

Table 13.10 summarizes the pixel aspect ratio values to be used when rendering text:

Table 13.10: Pixel aspect ratios

Text Style	Co-ordinate system	Pixel aspect ratio for text rendering (parX/parY)
'plain'	SD (non-square-pixel)	56/45
	HD (square-pixel)	7/8
'square'	SD (non-square-pixel)	64/45
	HD (square-pixel)	1/1

All text flow calculations (see clauses 13.5.4, 13.5.5, 13.5.6 and 13.5.9) and positioning calculations (see clauses 13.5.7 and 13.5.8) shall be performed at SD resolution. The final rendering of text shall be performed at HD resolution.

The 'square' style is required only in receivers supporting HDGraphicsPlaneExtension but to ensure consistency in rendering between receivers using different HD graphics plane resolutions, text flow calculations shall nonetheless be performed at SD using the specified pixel aspect ratio.

Clause 13.10 shows an example of the text rendering process when using an HD graphics plane.

13.5.4 Rendering within the limits of the Text object

13.5.4.0 Approach

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 the present document 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

$$yOffsetTop_{\text{pixels}} = \begin{cases} \text{div}(yMax_{\text{outlineResolution}} \times \text{fontSize}, \text{outlineResolution}) & yMax > 0 \\ 0 & yMax \leq 0 \end{cases} \quad (6)$$

- yOffsetBottom

$$yOffsetBottom_{\text{pixels}} = \begin{cases} \text{div}(-yMin_{\text{outlineResolution}} \times \text{fontSize}, \text{outlineResolution}) & yMin < 0 \\ 0 & yMin \geq 0 \end{cases} \quad (7)$$

- xOffsetLeft

$$xOffsetLeft_{\text{pixels}} = \begin{cases} \text{div}(-xMin_{\text{outlineResolution}} \times \text{fontSize} \times \text{parY}_{SD}, \text{outlineResolution} \times \text{parX}_{SD}) & xMin < 0 \\ 0 & xMax \geq 0 \end{cases} \quad (8)$$

where parX_{SD} and parY_{SD} are values from table 13.10.

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 13.2, $yOffsetTop$, $yOffsetBottom$ 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 the present document.

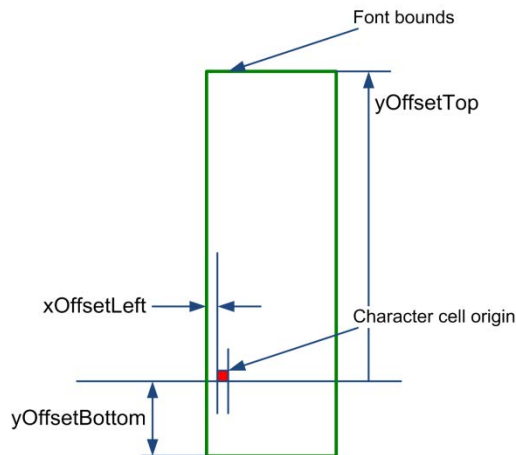


Figure 13.2: Font Bounds

13.5.4.1 Vertical limits

The origin of any character shall be at least $yOffsetTop$ (see clause 13.5.4) inside the top edge of the **Text** object and at least $yOffsetBottom$ (see clause 13.5.4) 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 shall be:

$$\text{num_lines} = \text{floor}((\text{height_Text_object} - (yOffsetBottom + yOffsetTop))/\text{linespace}) + 1 \quad (9)$$

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 $\text{floor}(A)$ 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 shall be:

$$\text{available_width} = \text{width_Text_object} - \text{xOffsetLeft} \text{ (see clause 13.5.4)} \quad (10)$$

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

13.5.5.0 Approach

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

13.5.5.1.0 Parameters used

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

- text font size (one of the values in table 13.5);
- 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 has 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 shall 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 shall be computed.

Logical width of N characters_{points} =

$$\text{div}((N-1) \times \text{letter space}, 256) + \text{div}\left(\text{fontsize} \times \left(\sum_{i=1}^N \text{charSetWidth}[i] + \sum_{i=1}^{N-1} \text{kern}[i, i+1]\right), \text{metricsResolution}\right) \quad (11)$$

$$\text{Logical width of N characters}_{\text{pixels}} = \text{div}(\text{logical width of N characters}_{\text{points}} \times \text{parY}_{SD}, \text{parX}_{SD}) \quad (12)$$

Where in $\text{div}(A, B)$:

- B is unsigned and A is signed; and
- $\text{div}(A, B) = \text{ceil}(A/B)$.

Where '/' is a rational divide and $\text{ceil}(A)$ 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 ' parX_{SD} ' and ' parY_{SD} ', see table 13.10.

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 shall be independent of the **Text** object's justification settings and shall 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.)

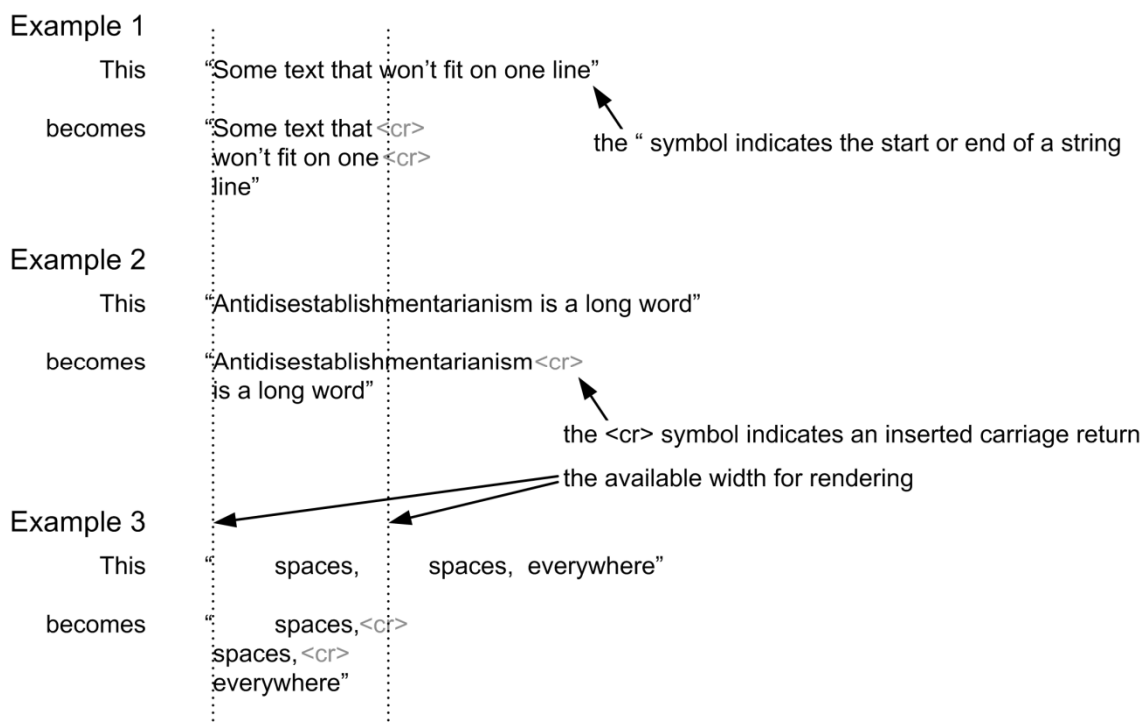


Figure 13.3: Text wrapping examples

After the text wrapping pre-processing is performed, the text truncation rules in clause 13.5.7 shall still apply. The present document does not support hyphenation and sequences of non-breaking characters that exceed the available width are not wrapped (see example 2 in figure 13.3) but shall 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 shall 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 13.4. When VerticalJustification = start (top aligned text) the baseline of the first (top most) line shall be yOffsetTop (see clause 13.5.4) inside the top edge of the Text object and each following line shall be spaced according to the value of linespace.

When positioning lines of text vertically and 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 SD pixel).

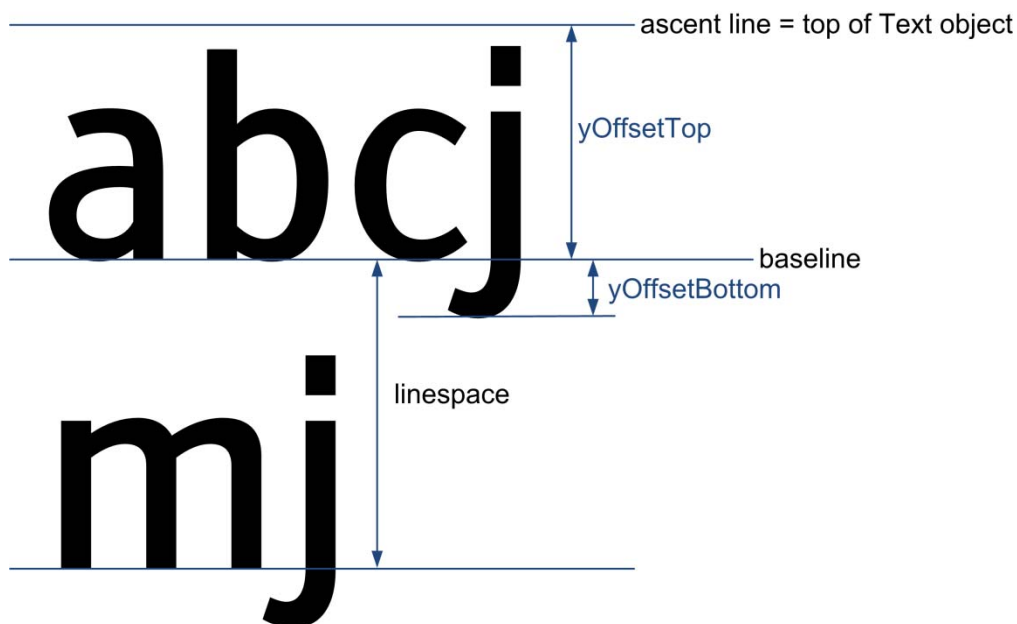


Figure 13.4: Vertical measures

When `VerticalJustification = end` (bottom aligned text) the origin of the last (bottom most) line shall be `yOffsetBottom` (see clause 13.5.4) 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 13.5.

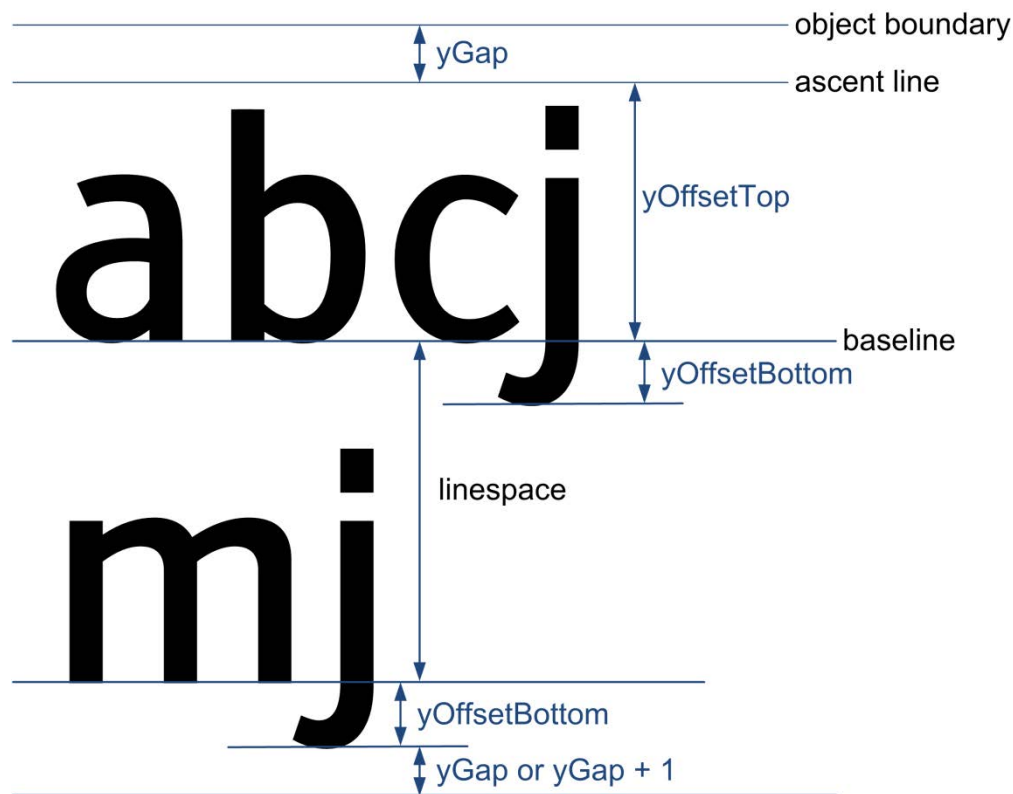


Figure 13.5: Vertical measures

13.5.7.3 Void

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 based on the text's logical width. 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` (see clause 13.5.4) 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 shall be rendered with its right hand edge aligned to the text box. In the "centre" case the centre of the string shall 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` (see clause 13.5.4) 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` (see clause 13.5.4).

When positioning lines of text vertically and `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 SD pixel). Note that this may place the first character's origin to the left of `xOffsetLeft`.

13.5.8.3 Examples

See figure 13.6.

H Justification->	start	centre	end	
Right:	This text is too lo	his text is too lon	is text is too long	String: This text is too long
Wrong:	This text is too lor	This text is too lor	his text is too long	
Wrong:		This text is too lo	This text is too lo	

Figure 13.6: Horizontal positioning example

13.5.8.4 Scaling for HD resolution graphics planes

Receivers that support `HDGraphicsPlaneExtension` shall perform the final rendering process at the HD graphics plane resolution, using the scaled text sizes described in clause 12.11.3.3.

13.5.9 Tabulation

In left aligned text (`HorizontalJustification = start`) tab stops are defined horizontally every 45 SD 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. See figure 13.7.

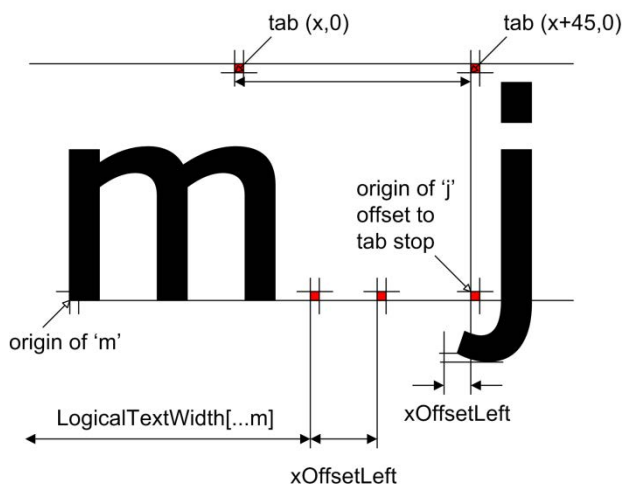


Figure 13.7: Effect of horizontal tabulation

- 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` (see clause 13.5.4). If the normal origin of the next character to be rendered after the tab character is after a tab stop, a tab character shall 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 shall start 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 13.8.

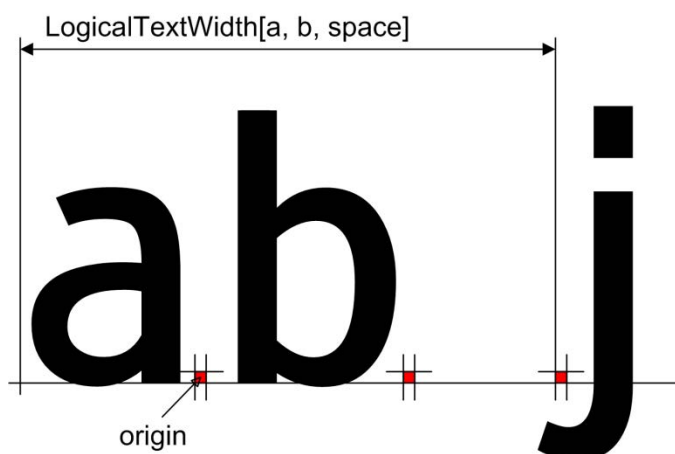


Figure 13.8: Calculation of character placement

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.

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 cannot be implemented. Thus whilst the "logical" rules shall 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 13.11.

Table 13.11 also indicates those characters that may be considered as candidates for line breaking when TextWrapping is enabled.

Table 13.11: 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 depends 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.

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 Resident Programs.

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 clauses 13.6.4 and 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 13.12 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 13.13. 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 13.14.

Engines shall ignore start and end mark-up sequences that do not conform to the present document by skipping past the $N + 3$ byte start sequence or 2 byte end sequence. This includes mark-up sequences in which the `parameters_length` field (N) does not match that expected for the `markup_start_identifier`.

Table 13.12: General format for start of text mark-up

Syntax	Bits	Value	Note
<code>start_of_markup</code>	8	0x1B	Escape
<code>markup_start_identifier</code>	8	0x40 to 0x5E	"@" to "^"
<code>parameters_length</code>	8	N	
<code>for(i=0; i<N; i++) {</code> <code> parameter_byte</code> <code>}</code>	8	0x00 to 0xFF	

Table 13.13: General format for end of text mark-up

Syntax	Bits	Value	Note
<code>end_of_markup</code>	8	0x1B	Escape
<code>markup_end_identifier</code>	8	0x60 to 0x7E	" to ~"

Table 13.14: 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 (see 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 the present document.			

13.6.5 Future compatibility

Whenever future extensions to the set of mark-up codes are defined, the `markup_end` identifier for each pair shall 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 13.15.

Table 13.15: Characters supported by EntryField

InputType	Set of characters	Comment
numeric	0 to 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		

If the receiver implements `InteractionChannelExtension` then it shall provide the augmented entry field input specified in table 13.16.

Table 13.16: Characters supported by EntryFields when InteractionChannelExtension implemented

InputType	Set of characters	Comment
numeric	0 to 9 (UTF-8 coded Unicode 0x30 to 0x39).	
alpha	A to Z (UTF-8 coded Unicode 0x41 to 0x5A) and, a to z (UTF-8 coded Unicode 0x61 to 0x7A).	See clause 13.7.4
any	All printable characters in the character repertoire. Note that input methods may not support all of these characters - see clause 13.7.4.2 (Minimum requirements).	See clause 13.7.4
listed	Any application-defined subset of the set of characters required by the 'any' InputType.	See clause 13.7.4

13.7.2 Appearance

13.7.2.1 Receivers that do not implement InteractionChannelExtension

See figure 13.9.

The EntryField shall 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. Similarly, the appearance of any EntryField that contains new line characters is undefined.

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 SD pixels on all sides from the BoundingBox.

The 4 SD 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.

The shape of the caret is implementation specific but shall be clearly visible and its colour shall be the EntryField's TextColour. The caret shall not affect the flow of text within the entry field.

If the ObscuredInput attribute is set True each character (including any OriginalContent) shall be represented by the asterisk character (0x002A).

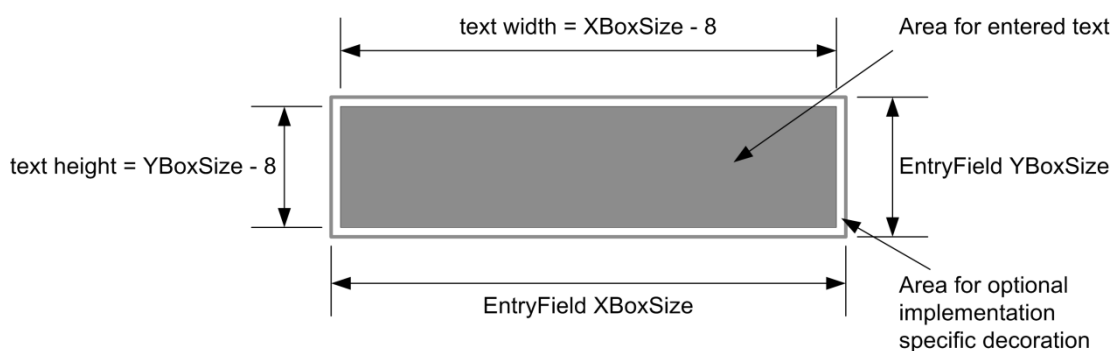


Figure 13.9: Area for text in an EntryField

13.7.2.2 Receivers that implement InteractionChannelExtension

Receivers that implement the InteractionChannelExtension modify the behaviour specified in clause 13.7.2.1 as described in this clause.

The restriction to a single line of text is removed. This removes the specifications imposed by the first paragraph in clause 13.7.2.1.

The TextWrapping attribute shall be supported for EntryFields. If at any time the content is too big to fit in the EntryField, the appearance of the content shall be as for a Text object with the same HJustification and VJustification attributes. The engine shall not provide scrolling.

If the ObscuredInput attribute is set to True each character (including any OriginalContent) shall be represented by the asterisk character (0x002A). However if the InputType is alpha, any or listed and a multi-press entry mechanism is implemented the character being selected shall be unobscured and shall only be obscured once it has been added to the text of the EntryField.

13.7.3 Behaviour

13.7.3.1 Character encoding

The content of an EntryField shall not include any of the non-printing characters described in clause 13.6.3 nor the carriage return character. Receiver behaviour is undefined if such characters are used. However, the content may contain characters not permitted for the InputType of the EntryField, 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 shall use UTF-8 encoding. In the present document, except where InteractionChannelExtension is implemented, 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

The present document defines the following additional semantics for the EntryFieldFull event and the MaxLength attribute:

- the EntryFieldFull event shall 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 shall not be triggered;
- the EntryFieldFull event shall only be triggered as the result of viewer interaction and shall never be 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 shall 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 shall be the responsibility of the application;
- SetData on an EntryField shall 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification, setting the value of EntryPoint to less than 0 shall have the same effect as setting it to 0.

13.7.3.4 Successive character entry

When an EntryField accepts input characters they shall be added to the TextData in the normal order for the LineOrientation and StartCorner of the object.

In the present document the only required `LineOrientation` and `StartCorner` shall be 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 11.8):

- `select` shall terminate interaction;
- `cancel` shall terminate interaction and shall also set the `TextData` attribute to an empty string;
- `left` shall result 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 shall replace the character most recently referenced by the `EntryPoint` (i.e. the character just after the insertion point shall be replaced and then the insertion point shall move one character space so that it is just after the character just entered).
- If the `OverwriteMode` attribute is set to `False` any new character shall be inserted before the character originally referenced by the `EntryPoint`.
- During interaction the behaviour shall be 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 shall never be considered part of the `TextData` attribute.

In the examples given in table 13.17 the initial string is "abc" and the entered string is "123".

Table 13.17: `EntryField` interaction examples

<code>OverwriteMode</code>	<code>Initial EntryPoint</code>	<code>Resulting String</code>	<code>Resulting EntryPoint</code>
False	0	123abc	3
False	1	a123bc	4
False	3	abc123	6
True	0	123	3
True	1	a123	4
True	3	abc123	6

13.7.4 Non-numeric input

13.7.4.0 Scope

This clause shall only apply to receivers that implement `InteractionChannelExtension`.

13.7.4.1 Introduction

EntryField objects allow for the input of non-numeric characters. The behaviour described in clause 13.7.4 applies where an EntryField has an InputType of 'alpha' or 'any', or where the InputType is 'listed' and the list of characters includes any character other than the numbers '0' to '9' (Unicode 0x30 to 0x39) (see table 13.16).

13.7.4.2 Minimum requirements

The implementation of the entry field shall meet at least the following requirements:

- Receivers shall support at least one method of non-numeric input.
- Shall be capable of returning both lower-case and upper-case characters.
- All input methods shall be able to return at least the characters listed in table 13.16.
- The default input method shall not make the MHEG presentation unreadable.
- Any optionally implemented input method that substantially obscures the MHEG presentation shall only be activated following informed deliberate user action other than an entry field gaining focus (e.g. an implementation dependent method such as pressing a dedicated button, opening a pull-out keyboard, picking up a specialized input device).
- Where the InputType is other than 'any' then the input method should filter any feedback to the viewer so that only acceptable characters can be entered.

13.7.4.3 SMS entry method

13.7.4.3.0 Scope

This clause defines a recommended non-obscuring method that meets the minimum requirements above and can be implemented using a remote control without alphabetic keys.

Receivers implementing this method shall use a "multi-press with timeout" method for entry of non-numeric input into an EntryField, as follows.

13.7.4.3.1 Basic Method

Each numeric key shall have a set of non-numeric symbols associated with it. The first time that a particular numeric key is pressed, the first non-numeric symbol shall be added to the content of the EntryField. For each subsequent press of the same key within a timeout period, the most recently entered character shall be replaced with the next non-numeric symbol for that key (or the first non-numeric symbol if there are no more) and a new timeout period shall begin.

After a timeout period has expired, any key press shall cause a new character to be entered.

13.7.4.3.2 Timeout Period

The key press timeout should be 1 ½ seconds unless the key repeat rate achievable with the receiver's remote control dictates a longer timeout period.

13.7.4.3.3 Appearance during input

The receiver shall represent the caret of the EntryField in a different form during the timeout period, returning to the standard form once the timeout expires.

13.7.4.3.4 Character to key mappings

Receivers shall use the basic mappings of alphabetic characters to keys defined by ISO/IEC 9995-8:1994 [29] with the addition of the key's own numeric character to the end of each list. In addition, certain symbol characters shall be assigned to the keys 1 and 0. The complete list of key assignments shall be as follows.

Table 13.18: Character to key mappings

Key	Characters (Unicode value in parentheses)			
1 (or 0) (see note 1)	Space (0x20), Comma (0x2C), Question mark (0x3F), Apostrophe (0x27), Semicolon (0x3B),		Full stop (0x2E), Hyphen-minus (0x2D), Exclamation mark (0x21), Colon (0x3A), Solidus (0x2F),	1 (0x31)
2	a (0x61), A (0x41),	b (0x62), B (0x42),	c (0x63), C (0x43),	2 (0x32)
3	d (0x64), D (0x44),	e (0x65), E (0x45),	f (0x66), F (0x46),	3 (0x33)
4	g (0x67), G (0x47),	h (0x68), H (0x48),	i (0x69), I (0x49),	4 (0x34)
5	j (0x6A), J (0x4A),	k (0x6B), K (0x4B),	l (0x6C), L (0x4C),	5 (0x35)
6	m (0x6D), M (0x4D),	n (0x6E), N (0x4E),	o (0x6F), O (0x4F),	6 (0x36)
7	p (0x70), P (0x50),	q (0x71), Q (0x51),	r (0x72), R (0x52),	s (0x73), S (0x53), 7 (0x37)
8	t (0x74), T (0x54),	u (0x75), U (0x55),	v (0x76), V (0x56),	8 (0x38)
9	w (0x77), W (0x57),	x (0x78), X (0x58),	y (0x79), Y (0x59),	z (0x7A), Z (0x5A), 9 (0x39)
0 (or 1) (see note 1)	Ampersand (0x26), Low line (0x5F),		Commercial at (0x40),	0 (0x30)
NOTE 1: The mapping of the 1 and 0 key may be interchanged. However, the 1 and 0 characters shall be mapped to the corresponding keys.				
NOTE 2: The order of values is left to right, then top to bottom, i.e. Space, Full stop ... Solidus, 1 for Key 1 and Ampersand, Commercial at, Low line, 0 for Key 0.				
NOTE 3: The UTF-8 encoding for each of these characters is one byte long.				

13.7.4.3.5 Character subsets

Applications may restrict the range of characters to be permitted in an EntryField using the 'listed' InputType. Where the range of characters is limited, any character that is not present in the application-supplied character list shall be skipped when the key associated with that character is pressed.

13.8 HyperText

13.8.0 HyperText markup format

In addition to the mark-up codes identified for Text objects in clause 13.6 HyperText objects can also include the markup in table 13.19. See clause 17.9.

Table 13.19: 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. 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 13.20. 0xnn is the length of the body_attr_bytes OctetString.

Markup sequences defined in table 13.19 shall not be nested. Receiver behaviour is undefined if such nested markup is encountered.

Table 13.20: 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 13.20 are as follows:

anchor_wrapping_flag: When set to True this flag indicates that the input focus shall wrap 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 shall remain on the limiting visible anchor and the top event or bottom event (as appropriate) shall be fired (see clause 13.8.3.3).

Where there is no body attributes markup present, anchor wrapping shall be disabled.

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 shall be the default given in clause 13.8.2.2.

The colour encoding is shown in figure 12.5.

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 shall be the default given in clause 13.8.2.2.

The colour encoding is shown in figure 12.5.

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. Visited anchors shall appear the same colour as unvisited ones (see anchor_colour in table 13.20).

reserved_byte: Byte reserved for future use.

13.8.1 HyperText anchors

After the application transfers the focus of user interaction to a HyperText object (by using the SetInteractionStatus action) the engine shall manage the 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 shall be returned as the associated data of the AnchorFired event and shall be placed in the LastAnchorFired internal attribute.

The engine shall be 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.

13.8.2 Appearance

13.8.2.1 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 shall still appear in the colours defined for an anchor.

13.8.2.2 Default anchor colours

See table 13.21.

Table 13.21: Default anchor colours

anchor_colour	0x0000FF00 (saturated blue)
active_anchor_colour	0xFF000000 (saturated red)

13.8.2.3 Highlight

The HighlightStatus attribute shall not affect the visible appearance of HyperText objects in the present document.

13.8.3 Behaviour

13.8.3.0 Basic behaviour

The exact behaviour during interaction depends 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.

13.8.3.1 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 shall move the focus to the previous anchor. The "down" or "right" user interface functions shall move the focus to the next anchor. The focus can move to anchors that are not currently visible on screen in this way.

13.8.3.2 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 shall generate a FocusMoved event (see clause 11.12.12.2). 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 shall not be 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.

13.8.3.3 Special behaviour at boundaries

NOTE: This clause addresses the case when the anchor_wrapping_flag (see table 13.20) 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` (see table 13.20) 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"` (see table 16.1);
- 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"` (see table 16.1);
- 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 - see above) shall be fired instead of the `FocusMoved` event (see clause 11.12.12.2).
 - The application author is responsible for appropriately "using" the "boundary" event. For example, a `Link` may be provided to set the interaction status of the `HyperText` object to `False` and to then highlight some other `Interactable`.
 - 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 shall continue 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 (see above) and pressing down/right shall generate the bottom event (see above).

Therefore, if the user navigation direction is "up" (up or left arrow key) when the focus is on the top most anchor there shall be no visual feedback unless specifically authored into the application. A subsequent "down" (down or right arrow key) user input shall 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 - see table 16.1) event is generated as the result of user navigation away from the object. For example, an engine using cursor/mouse based interaction may generate an `AnchorFired` event with `EventData "rec://htext/top"` (see table 16.1) when the cursor leaves the upper half of the bounding box of the `HyperText` object and "rec://htext/bot" (see table 16.1) when the cursor leaves the lower half of the bounding box.

For clarification, when the boundary events are generated (i.e. `AnchorFired` events are generated with either "rec://htxt/top" or "rec://htxt/bot") the `HyperText` object's `LastAnchorFired` attribute shall also be set according to the `EventData` value.

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 shall be its dimension along the axis defined by the **Orientation** attribute. The Thumb's position and length shall vary depending upon the **SliderStyle** attribute as follows:

- If the **SliderStyle** is set to normal, the Thumb shall be rendered as a 9 SD 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. Positions for intermediate values of **SliderValue** shall be evenly spaced within these limits (to within one SD pixel).
- If the **SliderStyle** is set to thermometer, the Thumb's position and length shall be as defined in the **SliderStyle** attribute in the ISO/IEC 13522-5 [14] (MHEG-5) specification.
- If the **SliderStyle** is set to proportional, the Thumb's position and length shall be as defined in the **SliderStyle** attribute in the ISO/IEC 13522-5 [14] (MHEG-5) specification.

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 Text rendering example (informative)

This example illustrates the steps involved in rendering the following MHEG-5 Text object onto a 1 920 x 1 080 high definition graphics plane. Note the example text does not include any tab characters or any words that require truncation; the steps shown therefore omit calculations for those eventualities.

```
{:Text 1
  :OrigContent 'This is a test of high definition text rendering'
  :OrigBoxSize 300 60
  :OrigPosition 100 100
  :FontAttributes 'plain.24.24.0'
  :Hjustification start
  :Vjustification start
  :TextWrapping true
}
```

The sequence of steps is as follows:

- 1) Using SD resolution calculations, determine the number of lines that can be rendered and the width available for rendering. See clause 13.5.4. Result: num_lines = 2; available_width = 296 (xOffsetLeft = 4)
- 2) Using SD resolution calculations and a horizontal conversion factor of 45/56, determine the lines of text to be rendered. See clauses 13.5.4 and 13.5.6.
Result:
line1 = "This is a test of high definition text", width = 290 (metrics width = 30697, points width = 360)
line2 = "rendering", width = 79 (metrics width = 8314, points width = 98)
- 3) Using SD resolution calculations, determine the vertical position of each line of text. See clause 13.5.7.
Result:
line1 baseline y = 122 (yOffsetTop = 22)
line2 baseline y = 146 (linespace = 24)
- 4) Using SD resolution calculations, determine the horizontal position for the start of each run of text. See clause 13.5.8.2.
Result:
line1 run 1 x = 104 (xOffsetLeft = 4)
line2 run 1 x = 104
- 5) Map the positions of the runs of text to the HD co-ordinate system. See clause 12.11.3.1.
Result:
line1 baseline start position = (277, 228)
line2 baseline start position = (277, 273)
- 6) Map the text size to the HD co-ordinate system. See clauses 12.11.3.3 and 13.5.3.
Result:
text size = 45pt with an 8/7 horizontal stretch factor
- 7) Render the text using the values calculated in steps 5 and 6.

For the same text object but with FontAttributes 'square.24.24.0', the calculations are as follows:

- 1) num_lines = 2; available_width = 297 (horizontal factor now 45/64, xOffsetLeft = 3)
- 2) line1 = "This is a test of high definition text", width = 254 (factor 45/64)
line2 = "rendering", width = 69 (factor 45/64)
- 3) line1 baseline y = 122
line2 baseline y = 146
- 4) line1 run 1 x = 103 (xOffsetLeft = 3)
line2 run 1 x = 103
- 5) line1 baseline start position = (274, 228)
line2 baseline start position = (274, 273)
- 6) text size = 45pt with no horizontal stretch factor

If the text had included tab characters, there would have been additional runs of text to consider from step 4 onwards. If text wrapping had not been used or if there had been a single word longer than the available_width, truncation would have been performed, either after step 4 or at step 7.

14 MHEG 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

14.2.1.0 Default behaviour

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 shall all be 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification 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.*"

In the present document the physical rendering shall continue 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 shall be 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, as defined by clause 14.2.4.

Once the **Application** object is activated, stream decoders shall 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 Synchronizing stream decoder state

14.2.4.0 Point of synchronization

When an application is started, the receiver shall delay synchronizing the state of its stream decoders with the new state set by the application (as determined by the presence or absence of any active MHEG-5 Stream or I-frame Bitmap objects) until the point when the synchronous action queue first becomes empty after the completion of the Application object's Activation behaviour.

NOTE 1: This point occurs after the processing of any Links handling Application IsRunning events but before any processing in response to asynchronous events, such as ContentAvailable. This means that the synchronization also occurs after any Scene transition performed by such an IsRunning Link but before a Scene transition performed in response to a ContentAvailable or other asynchronous event.

If at this point, the screen has been locked, this synchronization of the state of any Stream and I-Frame Bitmap objects shall be delayed until the screen is unlocked, in accordance with clause 10.1.3 of the ISO/IEC 13522-5 [14] (MHEG-5) specification.

Synchronization comprises:

- 1) Turning audio and video stream decoders on or off (as a result of the presence, state or absence of the corresponding MHEG-5 Video and Audio objects i.e. control 'E1' in figure 14.2 and figure 14.3).
- 2) If enabled, setting the presentation state of the audio and video decoders to reflect that of the internal attributes of the corresponding MHEG-5 Video and Audio objects i.e. presentation control 'P1' in figure 14.2 and figure 14.3.
- 3) Turning subtitles on or off in receivers which support simultaneous presentation of MHEG-5 and subtitles i.e. in response to the states of controls 'E1' and 'E2' in figure 14.4.
- 4) Initiating presentation of any I-frame Bitmap image in the video plane. Presentation of I-frames from a previous application shall not continue following a Launch, Spawn or Quit elementary action.

NOTE 2: This behaviour delays some of the effects of the Preparation and Activation behaviours described in clause 37.3 of the ISO/IEC 13522-5 [14] (MHEG-5) specification when these occur during application startup.

14.2.4.1 Graphics Plane

Unless the screen is locked, it is implementation dependent whether changes to the graphics plane resulting from Video and Bitmap object operations take effect prior to the synchronization point.

14.2.4.2 Stream component selection

The timing of the selection of the appropriate components from the broadcast i.e. selection control 'S1' in figure 14.2 and figure 14.3 shall be controlled by the behaviours described in section 37.3 of the ISO/IEC 13522-5 [14] (MHEG-5) specification and shall not be delayed. This includes both service and component selection.

14.2.5 Stream continuance on Application object deactivation

When an MHEG-5 Application is deactivated all other objects, including any active Stream objects, shall also be deactivated. Clause 37.3 of the ISO/IEC 13522-5 [14] (MHEG-5) specification 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, or if the Stream object has the Storage attribute set to stream and does not reference a broadcast stream, observe the standard behaviour.

- If the **Stream** object has the **Storage** attribute set to **stream** and it references a broadcast stream, override this behaviour. Specifically, the stream decoders associated with a **Stream** object do not automatically stop, and audio, video and subtitles (as relevant) shall 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 8.2.

NOTE: This deactivation behaviour is irrelevant if the application is being killed by the receiver (see clause 14.2.1).

14.2.6 Locating components carried in Transport Streams

14.2.6.0 Identifying a component

This clause shall only be 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 the present document 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).
 - For broadcast multiplexes the data shall resolve to identifying a service via the triple: `original_network_id`, `transport_stream_id` and `service_id`. For Interaction Channel references the "multiplex" is described by a URI that is resolved by the remote server to a file or data stream containing an MPEG program.
- 2) Identify individual components within this "multiplex" via the **ComponentTag** attribute.

14.2.6.1 Multiplex references

14.2.6.1.1 DSM-CC Stream object

In this mechanism the **OrigContent** attribute identifies a DSM-CC Stream object. In the present document the receiver need only support taps in the **Stream** object of **BIOP_PROGRAM_USE**. This kind of tap maps the "multiplex" onto a single DVB Service. See table 15.18.

14.2.6.1.2 URL explicit format

In this mechanism the **OrigContent** attribute contains a URL, as described in clause 16.3.3. This shall map the "multiplex" onto a single DVB Service.

Although most of this functionality can be provided by the DSM-CC Stream object, references can be built into the application. Also it is the same notation as used for the **SI_TuneIndex** (see clause 11.10.8.2) Resident Program and so provides a degree of consistency, i.e. the same data can be used both to preview and then to tune to a service.

14.2.6.1.3 URL inheritance formats

In this mechanism the **OrigContent** attribute contains one of two forms of URL:

- the **OctetString** "rec://svc/def" (see table 16.1) shall map the "multiplex" to the service most recently tuned to. For example, by the receiver's built-in navigator or the **SI_TuneIndex** (see clause 11.10.8.2) ResidentProgram; or
- the **OctetString** "rec://svc/cur" (see table 16.1) shall map 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" (see table 16.1) may be different from that referenced by "rec://svc/def" (see table 16.1) 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" (see table 16.1) shall be the same if service selection was done by the receiver's built-in navigator or the SI_TuneIndex (see clause 11.10.8.2) ResidentProgram.

See also clause 16.1.

14.2.6.1.4 Interaction Channel format

In this mechanism the OrigContent attribute contains a URI with a source of "http:". This indicates that the "multiplex" refers to a data stream that is sourced via the Interaction Channel. The data stream delivered is a Transport Stream containing a single MPEG Program, from which components are selected.

14.2.6.1.5 Services in other transport streams

In the present document all forms of multiplex reference shall be constrained to refer to services in the same transport stream as that delivering the current object carousel. Selection of services in other transport streams shall be supported by SI_TuneIndex (see clause 11.10.8.2).

14.2.6.1.6 StreamEvent events

Streams shall only support StreamEvents when referenced as a DSM-CC Stream object (see clause 14.2.6.1.1), and where the contained events are "do-it-now" events.

Because an NPT time base is not supported in the present document DSM-CC scheduled events are not supported. See clause 15.2.4.5.

14.2.6.1.7 CounterTrigger events

CounterTrigger events shall only be generated in the case where the Stream class references a stream obtained from the IP connection.

See clause 11.13.2.

14.2.6.1.8 Content management

Receivers shall ensure that any content management controls that apply to the referenced service are observed.

When more than one set of content management controls is active, e.g. because video and audio are selected from different services, receivers shall apply the most restrictive combination of the content management controls.

14.2.6.2 Component references

The mapping of ComponentTag values to service components is described in clause 15.3.

EXAMPLE: An application object might include:

```
{:Stream 1
  :OrigContent :ContentRef( "rec://svc/cur" ) (see table 16.1)
  :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 shall stop at the end of application object preparation.

14.2.7 Locating components carried in an Elementary Stream

NOTE: This is only relevant for Stream objects with the Storage attribute set to Memory.

The stream shall be identified via the Content internal attribute, which provides a reference to the relevant file. The ComponentTag attribute shall be ignored.

EXAMPLE: An application object might include:

```
{:Stream 117
  :InitiallyActive false
  :CHook 11
  :OrigContent :ContentRef('/sound1')
```



```

:Shared true
:Multiplex
(
  { :Audio 118
    :ComponentTag 2 //this tag value shall be ignored
  }
)
:Storage memory // shall be this type
}

```

14.2.8 Stream presentation errors

Where an application requests presentation of media that cannot be presented by the receiver, no other media shall be presented in its place. For example: a request to present video from a DVB service that is not known to the receiver shall not cause that service to be substituted with another; neither shall failure to resolve the `ComponentTag` of a stream component cause that component to be replaced with another.

Service references using DSM-CC `Stream` or `StreamEvent` messages shall also follow this behaviour but note that default behaviour is defined within the association tag mapping rules (see clause 15.3.3.2).

Failure to resolve a reference to a broadcast service in a `Stream` object shall not cause a `ContentRefError` engine event to be raised, except where the failure is in retrieving data from the carousel.

14.2.9 IC Stream buffering

14.2.9.0 Introduction

This section defines a buffer reference model that allows an application to:

- Present "near seamless" transitions between successive content.
- Insert secondary content such as adverts at arbitrary points within primary content.

The implementation of IC Stream content buffering is receiver dependent but shall support the buffer reference model as a minimum.

14.2.9.1 Buffer reference model

The IC stream buffer reference model defines how `Stream` content delivered by IP shall be downloaded and buffered in response to application control of MHEG-5 `Stream` objects. The model defines a single buffer which is managed by the receiver. Content is written to the buffer as it is downloaded and is consumed from the buffer as it is presented.

Content associated with a playing or paused `Stream` can be considered as 'primary'. Content associated with a 'prepared' `Stream` can be considered as 'secondary'. The application declares content as primary by activating the `Stream` object and as secondary by 'preparing' the `Stream` object as defined in clause 14.2.9.2.

When the last byte of downloading content is written to the buffer the receiver shall begin downloading and buffering any secondary content. If more than one `Stream` has been prepared then the associated secondary content shall be buffered sequentially in the order in which the `Streams` were prepared.

14.2.9.2 Application control of the IC stream buffer

IP-delivered stream content may be 'prepared' by an application by executing one of the following elementary actions on a `Stream`:

- A `Preload` on a `Stream` object with `AvailabilityStatus` set to `False`.
- A `SetData` on a `Stream` object with `RunningStatus` set to `False`.
- A `SetCounterPosition` on a `Stream` object with `RunningStatus` set to `False`.

The content associated with a prepared `Stream` object is considered secondary and shall therefore be downloaded when buffer space becomes available.

The application may define the end point of a Stream's content by executing the `SetCounterEndPosition` elementary action. When the last byte of content associated with this counter position has been acquired the receiver shall halt the download and begin downloading any secondary content. The application may reset the content end point to the natural end by executing a `SetCounterEndPosition` with a value of -1.

When the first byte of IP-delivered stream content is buffered a `ContentAvailable` event shall be generated. The event shall be generated as a result of the `Preload` or `SetData` elementary actions (when the content is to be started from the beginning) and as a result of a `SetCounterPosition` elementary action (when the content is to be started from some arbitrary mid point). If a secondary Stream's buffered content is discarded for any reason (e.g. as a result of a `SetCounterPosition` or `SetData` on a primary Stream) then the content shall be re-buffered when possible but shall not result in further `ContentAvailable` events being generated.

14.2.9.3 Restrictions

The receiver shall not buffer more than 6×10^6 bytes of content associated with each prepared or active Stream object prior to that content being presented. This is a maximum amount; receivers may choose to present before the buffer is full based on criteria such as the encoded bitrate of the content or the rate of buffer fill in an implementation dependent way.

A receiver may also provide additional buffering of presented content to improve skip backwards performance but this is not defined within the scope of the Buffer Reference Model.

Receivers shall discard any previously buffered content for a Stream object when that object is destroyed or when the `ContentPreparation` behaviour begins for new content as defined in section 20.6 of ISO/IEC 13522-5:1997/Cor.1:1999(E) [14].

When reading the data from a URL into the buffer for presentation during uninterrupted linear playback under ideal network conditions, receivers shall not make multiple requests for the content. Some implementations may need to use multiple requests to optimize stream playback, for instance to configure stream playback or during trick play. For these reasons, a receiver may issue multiple concurrent HTTP requests for short periods (e.g. until stream playback has started or once trick play has been completed).

Multiple concurrent HTTP requests are not allowed for the purposes of fast filling any buffers in the receiver.

NOTE 1: This is so as to minimize the overheads on the server.

All buffer management shall be done by the client application reading not more than the required amount of data from the TCP socket. This allows the TCP stack to manage the TCP session.

EXAMPLE: Figure 14.1 shows the application, reference buffer and presentation over time Streams B and C are inserted into Stream A.

NOTE 2: Figure 14.1 is not drawn to scale and exaggerates the gaps in presentation.

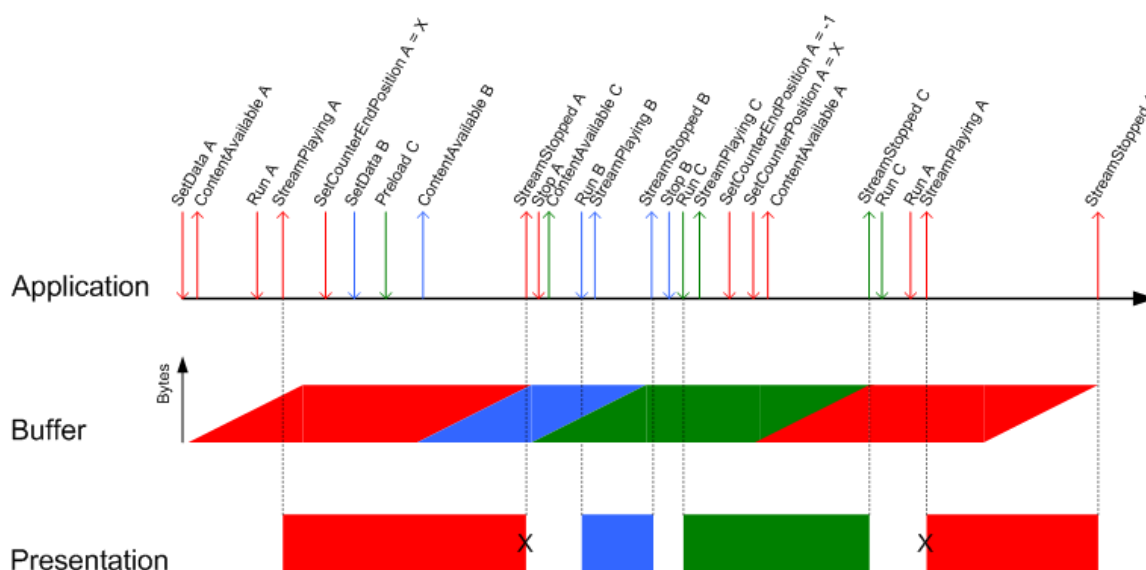


Figure 14.1: Non-linear stream buffer example

The application prepares Stream A by executing a SetData and content download begins. When sufficient content has been stored a ContentAvailable is generated. The application then Runs A and a StreamPlaying event is generated when A is presented.

The application then sets Stream A insertion position X with SetCounterEndPosition and prepares Streams B and C with SetData and Preload respectively. When the presentation of Stream A ends at position X a StreamStopped event is generated and the application stops A and runs B.

When presentation of Stream B ends a StreamStopped event is generated and the application stops B and runs C. Stream A end point is reset with SetCounterEndPosition and then prepared to begin at position X with SetCounterPosition. When Stream C's StreamStopped event is generated the application stops C and runs A and the presentation begins from position X.

14.3 Application interaction with user control of linear content decoders

14.3.0 Introduction and undefined behaviour

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.

If an MHEG Stream object (with a CHook value of 10) containing an MHEG Audio object has modified the default audio component (see clause 15.3.4.1) for a service, the rules for selection of an Audio Description component are not defined.

If an MHEG Stream object (with a CHook value of 10) containing an MHEG Video object has modified the default video component (see clause 15.3.4.1) for a service, the rules for selection of the Subtitle component are not defined.

14.3.1 Video decoder

See figure 14.2.

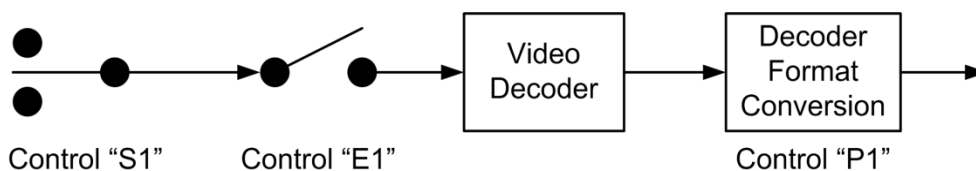


Figure 14.2: Video control - logical model

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.

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.

Presentation controls: P1 represents processing to control the scaling and positioning of the decoded video. It is based on a number of factors including (but not exclusively) viewer preferences and application signalling. See clause 14.5.

14.3.2 Audio decoder

See figure 14.3.

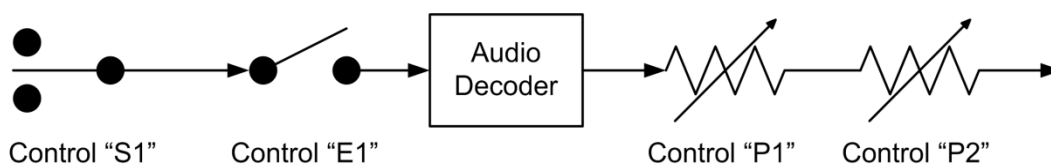


Figure 14.3: Audio control - logical model

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.

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.

Presentation controls: P1 is the application's control of the audio volume level. In ETSIEngineProfile1 this control shall 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 should be operated by the "Mute", "Vol+", and "Vol-" keys on the remote controller.

14.3.3 Subtitle decoder

See figure 14.4.

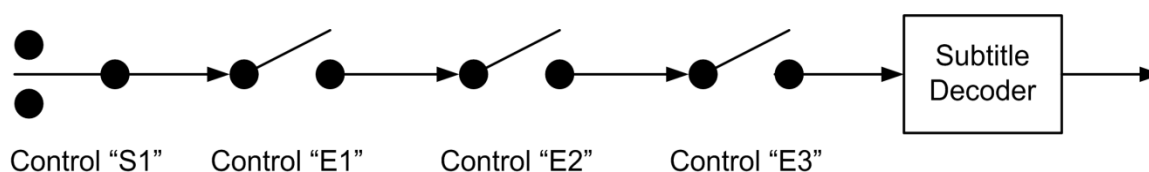


Figure 14.4: Subtitle decoder control - logical model

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 should be defined as part of receiver set-up and/or a "Subtitle" key on the remote controller.

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.

Presentation controls: None.

14.3.4 Selection of subtitle and audio description components

Where subtitles and audio description are selected by the viewer for presentation, and subject to presentation of subtitles as controlled by the SetSubtitleMode ResidentProgram, the components shall be presented.

Where the Stream reference is to the current service and Video and Audio objects (if present) have a ComponentTag of -1 then the choice of subtitle and audio description components shall be as if there were no MHEG application running. In all other cases of ComponentTag value or Stream reference the rules for selection of a subtitle or audio description component are not defined.

14.4 Application impact on stream decoder specification

14.4.1 DVB subtitles

14.4.1.1 Flexibility of control

14.4.1.1.0 Scope

If the receiver implements ICStreamingExtension then it shall support the simultaneous presentation of MHEG-5 applications and DVB subtitles.

14.4.1.1.1 Subtitles are a facet of full-screen video

In the present document subtitles shall be 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, or the origin of decode is moved to a point other than (0, 0), the subtitles shall not be presented unless both video and subtitles are scaled with an identical scaling factor and translated with an identical change of origin.

Implementations are encouraged to scale and translate subtitles along with video. Where possible, this should be done consistently for video and subtitles supplied by both broadcast and over IP.

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

14.4.3.0 Standard behaviour

Receivers shall not implement trick modes. However, as described in the ISO/IEC 13522-5 [14] (MHEG-5) specification, 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 shall be as follows:

- If the stream object contains an active Video object, the video decoder output shall freeze on and maintain the current or a subsequent frame in an implementation dependent way. If subtitles are active, they shall either freeze or disappear from the screen.
- If the stream object contains an active Audio object and has its Storage attribute set to stream then whilst it is paused, the audio output shall mute and the associated stream component decoders shall discard incoming data.

- If the stream object contains an active Audio object and has its Storage attribute set to memory then whilst it is paused, the audio output shall mute. If the stream object is resumed (e.g. by using the SetSpeed(1) action), the presentation of the audio shall continue from the point at which it was paused.

Stream component decoders that are controlled by other Stream objects shall not be affected by this behaviour.

If the stream object is playing streamed content from an IP connection then whilst it is paused, the audio output shall be muted and video shall be paused. If the stream object is resumed (e.g. by using the SetSpeed(1) action), the presentation of the stream shall continue from the point at which it was paused.

14.4.4 MPEG presentation

14.4.4.0 Introduction

This clause describes the scaling of the output of the MPEG video decoder when an MHEG-5 application is executing. It applies to the presentation of video streams, MPEG-2 I frames and, where supported, H.264/AVC 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 14.5. 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.

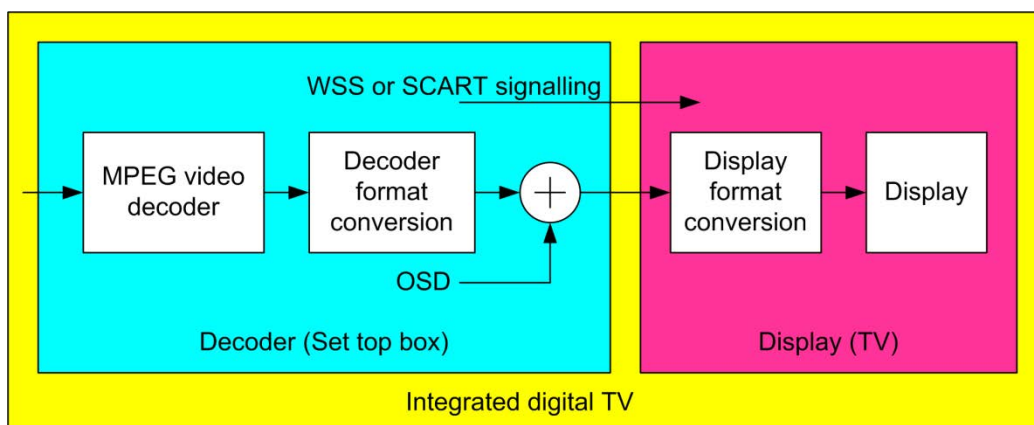


Figure 14.5: Receiver and display format processing reference model

Default scaling values are indicated in table 14.1.

Table 14.1: Default Scaling Values

Attribute		Default Value
BitmapScale (see note)	XBitmapScale	720
	YBitmapScale	576
VideoScale	XVideoScale	720
	YVideoScale	576
NOTE: In the present document, bitmap scaling is only supported for content hook 2 and where supported, content hook 7.		

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 shall always be 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 [i.2] 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 shall 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 should 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 shall implement 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 shall logically be 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 the present document are as follows:

- Full-screen (720 x 576) and quarter-screen (360 x 288) shall be supported for all video resolutions. For source video that has a `horizontal_size` of 545 to 720 pixels (i.e. where no up-sampling applies), additional scaling factors are required for which the resulting image width shall be any one of 1 440, 1 080, 720, 540 and 360 and the image height shall be any one of 1 152, 576 and 288.
- 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).

For the effect of scaled video on subtitles 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

14.4.4.5.0 Behaviour

All receivers shall display images that meet the following restriction:

- The decoded image, before any masking, shall meet the requirements defined in clauses 11.4.1.1 and 11.4.1.2.

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 need not be met.

14.4.4.5.1 Restricted capability

Some receivers do not have the capability to place decoded images at odd pixel positions on one or more axes. In these cases, receivers may round down an odd co-ordinate of the decoded image to the nearest even co-ordinate value. The implementation of the VideoToGraphics resident program shall take into account any such deterministic errors.

EXAMPLE: If an MHEG Video object were positioned at co-ordinates (25, 34) with a decode offset of (-12, 45), then normally a receiver shall apply a decode position of (13, 79). For receivers with restricted capability, a decode position of one of (12, 79), (13, 78) or (12, 78) may be applied instead.

14.4.5 Content management of IC streams

Receivers shall not allow streamed content delivered by IP to be:

- Redistributed through any local or remote network.
- Stored on storage media other than as required by the buffer reference model, see clause 14.2.9.
- Presented through any HDMI output without HDCP being enabled.

14.4.6 Multiple stream objects

14.4.6.0 Behaviour

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.

So, 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.

Receivers are not required to support the simultaneous presentation of Streams from both broadcast and an IP connection.

Receivers are not required to support the simultaneous presentation of more than one Stream object that is playing streamed content from an IP connection.

14.4.6.1 Simultaneous Demultiplexing

Receivers supporting the InteractionChannelExtension are required to present A/V stream components from IC delivered streams whilst simultaneously monitoring broadcast delivered life cycle signalling and DSM-CC object carousel updates. Receivers shall be able to simultaneously demultiplex at least one IC delivered transport stream and at least one broadcast delivered transport streams.

14.5 Application control of aspect ratio

14.5.0 Introduction

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 and scaling of 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 video greater than quarter-screen.

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 14.2.

Table 14.2: WSS signalling - no active Video object

Display aspect ratio	Scene AspectRatio		
	4:3	16:9	Unspecified
4:3	4:3	16:9 (see note 1)	4:3
16:9	4:3 (see note 2)	16:9	16:9
NOTE 1: WSS signalling to 4:3 displays will in many cases not be correctly interpreted. See clause 17.5.4.			
NOTE 2: 4:3 presentation may not be supported over HDMI and the presentation may be distorted to the display's native aspect ratio.			

14.5.2 I-frames

MPEG-2 and H.264/AVC 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 quarter-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 14.3 illustrates the preferred DecFC transformations for each state of WSS (obtained from clause 14.5.1).

Table 14.3: WSS signalling - quarter-screen video

WSS	Video coded frame	
	4:3	16:9
4:3	None	Centre cut-out
16:9	Pillar box (see note)	None
NOTE: Where a pillar box transformation cannot be supported, the preferred DecFC is None.		

14.5.4 Video greater than quarter-screen

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 14.4.

Table 14.4: Video full-screen or greater

Video Scene	4:3 coded frame		16:9 coded frame	
	4:3	16:9 (see note 1)	4:3	16:9
WSS (see note 2)	4:3	16:9	4:3	16:9
DecFC	None	None	As selected (see note 3)	None (see 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 clause 14.5.1 the "no active video object" and clause 14.5.2. See clauses 17.5.4 and 17.5.5.
 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 should 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 figure 14.6 and figure 14.7.

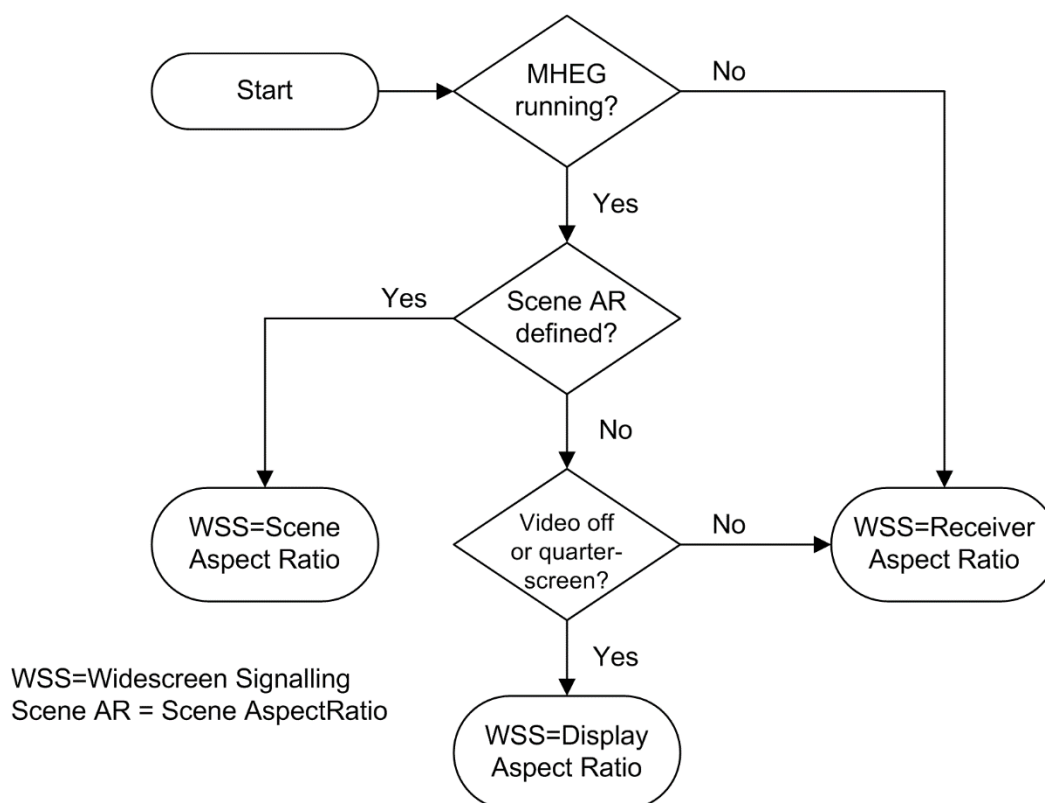


Figure 14.6: Widescreen Signalling (WSS) decision tree

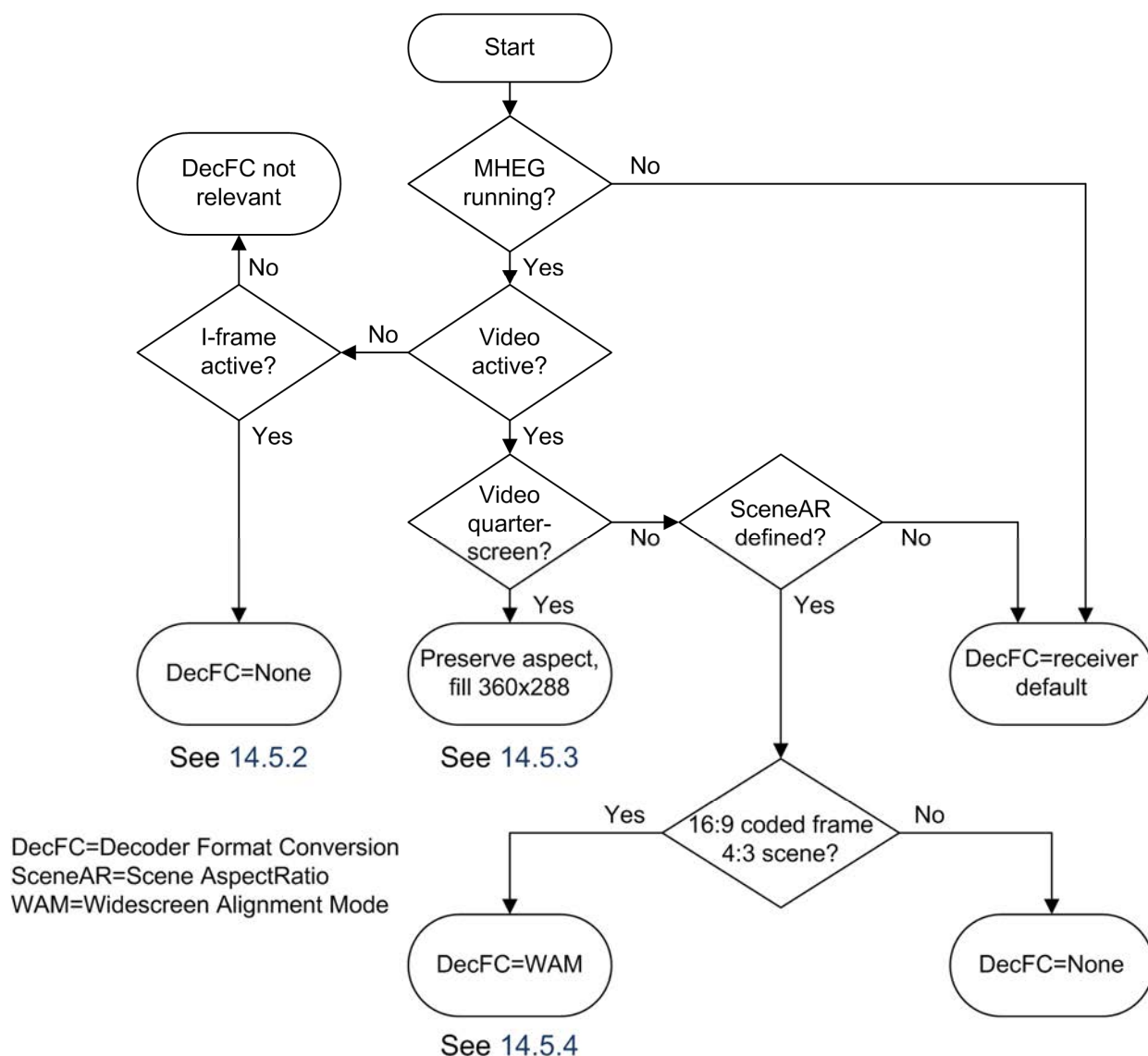


Figure 14.7: Decoder Format Conversion (DecFC) decision tree

14.6 Persistent storage

The engine shall provide "persistent" storage for 1 024 bytes of data.

For receivers that implement InteractionChannelExtension the persistent storage should be implemented in such a way that data is retained whilst the receiver is in standby mode.

The file name used to access this storage shall be of the form "ram://<name>" (see table 16.1). 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.

Receivers that implement InteractionChannelExtension shall implement storage that persists through a power cycle. This is referred to as "true persistent storage" and is described in 14.7.

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 shall be stored (not type information);
- the decoder model for memory use in the persistent store is given in table 14.5;
- the set of values shall be stored in the order they are enumerated in the ASN.1 DER encoding of the StorePersistent InVariables list;
- the behaviour of ReadPersistent() shall be undefined unless the set of values is enumerated in the same order as the StorePersistent() that created them.

Table 14.5: Memory model for persistent storage

Data	
Integer	4 bytes This shall be able to contain all of the possible values for an IntegerVariable as defined under clause 11.11.2.
Boolean	1 byte This shall be able to contain all of the possible values for an BooleanVariable as defined in clause 11.11.1.
OctetString	Number of bytes in OctetString + 4 bytes This shall be able to contain all of the possible values for an OctetString as defined in clause 11.11.3. 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 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 11.11.5. This model assumes ObjectReferences are stored like an OctetString with an Integer.
ContentReference	Number of bytes in ContentReference + 4 bytes This shall be able to contain all of the possible values for an OctetString as defined in clause 11.11.3. This model assumes ContentReferences are stored like an OctetString.

The <name> part of the file name "ram://<name>" (see table 16.1) shall be eight bytes long. The receiver shall provide storage for at least 32 such file names associated with the "ram://" persistent store.

Note that references to the hybrid file system cannot be used to access persistent storage. See clause 17.8.4.

14.7 True persistent storage

14.7.0 Behaviour

For receivers that implement InteractionChannelExtension, true persistent storage shall be provided. True persistent storage is defined as storage that persists through a typical user power cycle of the device; this may include a shutdown to standby followed by a small delay and then removal of the mains power.

A receiver shall make available at least 100 bytes of true persistent storage per service for at least 800 services, which can be used by a service to store one file. The receiver only needs to provide the ability for each file to be overwritten a minimum of 64 times every 28 days. The file name used to access this storage shall be of the form "pst://<name>".

The <name> part of the file name "pst://<name>" shall be a maximum of 8 bytes long. The first 3 bytes shall be a unique 3 character string provided to the broadcaster/application author by the DTG.

If more than 800 services have stored true persistent files, the receiver may delete true persistent files assigned to services that are not in the current service list.

Once a file has been stored, it can be read or overwritten by any service. If a service that did not originally create the file overwrites it, the service that the file is assigned to is changed to be that of the new service and any file assigned to that service is deleted. The restriction that the file is not overwritten more often than 64 times every 28 days is independent of the service that is trying to overwrite it.

When writing a file to true persistent storage the receiver shall execute the following steps:

- 1) If true persistent storage has been disallowed by a user (see clause 14.7.1, the file shall not be saved and the action shall complete returning StoreSucceeded is False.
- 2) If the size of the file is greater than 100 bytes, the file shall not be saved and the action shall complete returning StoreSucceeded is False.
- 3) If there is a true persistent file already assigned to the current service which was overwritten more than 64 times within the 28 day period, the receiver may choose not to save the file and complete the action returning StoreSucceeded is False.
- 4) If there is a true persistent file of the same file name assigned to any service which was overwritten more than 64 times within the 28 day period, the receiver may choose not to save the file and complete the action returning StoreSucceeded is False.
- 5) If there is a true persistent file of the same file name assigned to any service, the file shall be overwritten, assigned to the current service, any other file assigned to the current service shall be deleted, and the action shall complete returning StoreSucceeded is True.
- 6) The file shall be written, assigned to the current service, any other file assigned to the current service shall be deleted, and the action shall complete returning StoreSucceeded is True.

NOTE: The memory model and other rules for writing a file to persistent storage (see clause 14.6) apply equally to true persistent storage. References to the hybrid file system cannot be used to access true persistent storage.

14.7.1 Management of true persistent storage

Receivers shall clear all true persistent storage and any associated write counters when the user invokes a 'Factory Reset', but shall not clear the storage when a retune is performed, even if the service that a particular file is assigned to is not found.

Receivers may implement a true persistent storage management utility where the user is able to delete files assigned to particular services.

Receivers may implement a utility to allow a user to disable all access to true persistent storage. By default, access shall be enabled.

14.8 Receiver resource model

14.8.1 Memory

A complete model of how receiver memory is consumed by a running application is not defined in the present document. However, some aspects have been defined in the present document as follows:

- persistent file store and directory structures (see clause 14.6);
- the application identifier stack (see clause 8.2);
- buffers required to receive data from the network (see clause 15.2.7);
- HTTP cookies (see clause 15.7.5);
- the hybrid file system mapping table (see clause 15.12);
- downloadable fonts (see clause 13.3.1.4).

14.8.2 Numbers of objects

14.8.2.0 Minimum number of active objects using stream decoders

The minimum number of concurrently active MHEG-5 objects using stream decoders that the present document shall support are:

- 1 Video object (an MPEG video stream) or 1 Bitmap object using MPEG-2 or H.264 I-frame encoding;
- 1 Audio object (an MPEG audio stream) with source Stream or Memory.

See clause 17.4.1.

The numbers of other presentable object types (e.g. PNG bitmaps, buttons, etc.) are only limited by the decoder memory model.

14.8.2.1 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.

14.8.3 Link recursion behaviour

Engines shall allow at least 256 concurrent Actions, and at least 4 096 ElementaryActions, pending processing.

14.8.4 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.

14.8.5 Timer duration

Receivers shall support timer durations up to at least 86 400 000 ms (24 hours).

14.8.6 HD graphics bitmap requirements

Receivers supporting HDGraphicsPlaneExtension shall be able to decode and present PNG and JPEG bitmaps totalling 2.08 Mpixels simultaneously with either full-screen HD video or an H.264/AVC I-frame. Receivers shall be deemed to pass this requirement if they can present:

- 1) full-screen HD video, together with either:
 - a half full-screen 1 920 x 1 080 resolution PNG bitmap (1,04 Mpixels) concurrently with 99 equally sized PNG bitmaps totalling 1,04 Mpixels;
 - the above scenario using JPEG images.

- 2) an H.264/AVC I-frame, together with either:
 - a half full-screen 1 920 x 1 080 resolution PNG bitmap (1,04 Mpixels) concurrently with 99 equally sized PNG bitmaps totalling 1,04 Mpixels;
 - the above scenario using JPEG images.

14.9 Receiver process priority

14.9.1 OSD arbitration

Access to display resources shall be (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 the actual implementation is not specified in detail, in all cases the receiver shall manage stream decoders appropriately.

NOTE 1: 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 may lose 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 necessary.
 - There may be certain receiver overlaps that take some MHEG keys for their own UI (user initiated or otherwise) and allow other keys to be available to MHEG. However, manufacturers are encouraged to consider the impact of these overlays, and to consider usability.
- 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 MHEG application, the engine shall be responsible for re-establishing the state of the display, including the presentation of stream components. When the MHEG application is re-prioritized the presentation of streamed content should continue in the same state (for example scaling, screen position and time offset) as when the application was de-prioritized.

NOTE 2: Other resources may also be removed from the MHEG engine and in the limiting case the engine is completely descheduled (clause 14.9.2).

- 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 application, the engine shall be responsible for re-establishing the state of the display, including the presentation of stream components and I-frames. Stream components shall be restored to the state of the MHEG objects representing them. Any stream components that the Application has paused shall return to the paused state after the display resource is returned. Where the receiver cannot restore a paused video frame, it shall present black until the video is restarted.

14.9.2 Event handling whilst de-prioritized

14.9.2.1 Transparently

If the MHEG-5 engine is "transparently" de-prioritized it shall cease to get the asynchronous events that are a consequence of user interaction but it shall receive and process all other asynchronous events.

With respect to the present document the events not received are: `AnchorFired`, `EntryFieldFull`, `InteractionCompleted` and `UserInput`. The events that shall be received are: `AsynchStopped`, `ContentAvailable`, `CounterTrigger`, `EngineEvent`, `StreamEvent`, `StreamPlaying`, `StreamStopped`, and `TimerFired`.

When an Application is transparently de-prioritized, `StreamPlaying` and `StreamStopped` events shall be generated only as a result of the application's actions. No `StreamPlaying` or `StreamStopped` events relating to the deprioritization itself shall be delivered to the running application, for example if video were suspended for the display of a receiver guide.

14.9.2.2 Non-transparently

If the MHEG-5 engine is "non-transparently" de-prioritized the state of the application shall be preserved, but some or all of the resources supporting the engine can be 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 11.10).

14.10 Interaction with DVB Common Interface module system

14.10.1 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.

14.10.2 Introduction of CI sourced file system

Under certain conditions (see clause 14.10.3.1) 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.

14.10.3 Guidelines for using Application MMI resource

14.10.3.0 Introduction

This clause describes the use of the application MMI introduced by ETSI TS 101 699 [18] in the context of the present document.

14.10.3.1 Resource contention

See clause 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 CENELEC EN 50221 [5] and ETSI TS 101 699 [18].

The reasons defined by CENELEC 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 of the present document 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 may not be guaranteed a MMI session by the host.

14.10.3.2 RequestStart

14.10.3.2.0 Behaviour

See clause 6.5.2 in ETSI TS 101 699 [18].

14.10.3.2.1 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.

14.10.3.2.2 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 shall not be killed and the CI file system shall be mounted and become available to the application.
- If InitialObjectLength is 0 and no MHEG-5 application is currently running then the CI file system shall be mounted and the receiver shall continue 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 shall be 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 shall be mounted.

Also, with reference to the "Application context" (see clause 8.1.5):

- If InitialObjectLength is 0 there shall be no change in the application context for any currently running application.

In particular, the mounted object carousel (if any) shall remain the same and the Current source shall remain unchanged.

- If InitialObjectLength is > 0 then the application context shall become that of the newly introduced application.

In particular, the service's Initial carousel (if any - see clause 8.1.5.1) shall be mounted and the Current source (see clause 8.1.5.3) shall become 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 shall implicitly be "CI://" so the string "foo/bar" specifies file "bar" in the sub-directory "foo" of the root of the CI device.

14.10.3.3 RequestStartAck

See clause 6.5.3 in ETSI TS 101 699 [18].

14.10.3.4 FileRequest

See clause 6.5.4 in ETSI TS 101 699 [18].

The FileRequest message shall be overloaded to allow the transmission to the module of two different request types. The first shall be a file request as defined in ETSI TS 101 699 [18]. This shall be used to retrieve MHEG-5 files and content from the module. The second shall allow the creation of a private data pipe between the host and the module.

See table 14.6.

Table 14.6: FileNameByte field of the FileRequest

Syntax	Bits	Mnemonic
<pre> RequestType if (RequestType == "file") { for (i = 0; i < (N -1); i++) { FileNameByte } } if (RequestType == "data") { for (i= 0; i < (N - 1); i++) { DataBytes } } </pre>	8	
	8	
	8	

RequestType: Defines the type of request being made by the host. See table 14.7.

Table 14.7: RequestType field of FileRequest

RequestType	RequestType value
File	0x00
Data	0x01

FileNameByte: FileNameByte is a valid name for a DVB CI file as defined in clause 8.1.3.

DataBytes: The data bytes for the module.

14.10.3.5 FileAcknowledge

See clause 6.5.5 in ETSI TS 101 699 [18].

The FileAcknowledge shall 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.

FileOK: This 1-bit field shall be 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 14.8.

Table 14.8: FileByte field of the FileAcknowledge

Syntax	Bits	Mnemonic
<pre> RequestType if (RequestType == "file") { FileNameLength for (i = 0; i < FileNameLength; i++) { FileNameByte } FileDataLength for (i = 0; i < FileDataLength; i++) { FileDataByte } } if (RequestType == "data") { For (i= 0; i < (N-1); i++) { DataBytes } } </pre>	<p>8</p> <p>8</p> <p>8</p> <p>32</p> <p>8</p> <p>8</p>	

RequestType: Defines the type of request being responded to by the host. See table 14.9.

Table 14.9: RequestType field of FileAcknowledge

RequestType	RequestType value
File	0x00
Data	0x01

FileNameLength: The number of bytes in the name of the file.

FileNameByte: FileNameByte is the name of the file requested by the host. This data shall be returned to the host to allow for implementations that may request more than one file simultaneously.

FileDataLength: The number of bytes of data in the file.

FileDataByte: A byte of the file requested.

DataBytes: The data bytes for the host.

14.10.3.6 AppAbortRequest

See clause 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 shall not be terminated by this message. The host shall follow the defined auto-boot procedure following the termination of an application.

14.10.3.7 AppAbortAck

See clause 6.5.7 in ETSI TS 101 699 [18].

No values of AbortAckCode are defined in this application domain.

14.10.3.8 Asynchronous events

A mechanism for the DVB CI module to send asynchronous events or messages to an MHEG-5 application is not defined in the present document.

14.10.4 Application Info Resource "Enter_Menu"

The host shall provide access to module applications under user control from the resident navigator software.

15 File system profile

15.1 Introduction

15.1.1 Broadcast file system

The broadcast applications shall be transmitted using the DSM-CC user-to-user object carousels.

The present document 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 [i.4] - Implementation guidelines for data broadcasting.

With the constraints and extensions described in clauses 15.2 to 15.5.

Table 15.1 shows the key to certain notations that are used in the "value" columns of the syntax tables.

Table 15.1: 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.

15.1.2 Interaction channel

Clauses 15.6 to 15.15 of the present document are specific to receivers implementing InteractionChannelExtension.

Receivers that implement InteractionChannelExtension shall implement the IC file system. This is based on the following specifications:

- IETF RFC 1034 [i.6] - Domain Names - Concepts and Facilities
- IETF RFC 1035 [i.7] - Domain Names - Implementation and specification
- IETF RFC 1982 [i.8] - Serial Number Arithmetic
- IETF RFC 6265 [32] - HTTP State Management Mechanism
- IETF RFC 2131 [i.9] - Dynamic Host Configuration Protocol
- IETF RFC 2132 [i.10] - Dynamic Host Configuration Protocol Options and BOOTP Vendor Extensions
- IETF RFC 2181 [i.11] - Clarifications to the Domain Name System Specification
- IETF RFC 2246 [37] - The TLS Protocol Version 1.0
- IETF RFC 2459 [33] - Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- IETF RFC 2616 [30] - HTTP/1.1
- IETF RFC 2818 [31] - HTTP Over TLS
- IETF RFC 3986 [28] - Uniform Resource Identifier (URI): Generic Syntax

- Recommendation ITU-T X.509 [35] - Information technology - Open Systems Interconnection - The Directory: Authentication framework
- Recommendation ITU-T X.680 [6] - Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation
- Recommendation ITU-T X.690 [7] - Information Technology - ASN.1 Encoding Rules: Specification Of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) And Distinguished Encoding Rules (DER)
- ETSI TS 101 812 [36] - Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.3 with the constraints and extensions described in clauses 15.6 to 15.15.

15.2 Object carousel profile

15.2.1 DSM-CC sections

15.2.1.0 DSM-CC section format

All object carousel messages shall be transmitted using DSM-CC section format. The DSM-CC Section format is defined in clause 9.2 of the DSM-CC specification (ISO/IEC 13818-6 [12]).

The DSM-CC standard provides an option to use either a CRC32 or a checksum for detecting bit errors. In the present document restrictions are shown in table 15.2.

Table 15.2: 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: <ul style="list-style-type: none"> • all modules intended to be retrieved shall have the last section number $\leq 0xFE$; • 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 may contain parts of no more than four sections.

15.2.2 Data carousel

15.2.2.1 General

The definitions in table 15.3 shall apply to both the `dsmccDownloadDataHeader` and the similar `dsmccMessageHeader`.

Table 15.3: 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 <code>dsmccAdaptationHeader</code> 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 15.4.

Table 15.4: Restrictions on the DII

Field	Restrictions	Source
blockSize	Maximum size 4 066 (max. section payload - DDB-header size (18)). The recommended blockSize is 4 066.	ISO/IEC 13818-6 [12] (DSM-CC) the present document
windowSize	0 (not used for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
ackPeriod	0 (not used for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
tCDownloadWindow	0 (not used for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
tCDownloadScenario	0 (not used for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
compatibilityDescriptor(): compatibilityDescriptorLength	0 (no compatibility descriptor for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
PrivateDataLength	The receiver may ignore the possible contents of the privateData field.	DVB

15.2.2.3 DownloadServerInitiate

The DownloadServerInitiate shall be 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 15.5.

Table 15.5: Restrictions on DSI

Field	Restrictions	Source
compatibilityDescriptor(): compatibilityDescriptorLength	0 (no compatibility descriptor for object carousels).	ISO/IEC 13818-6 [12] (DSM-CC)
privateData	Contains the ServiceGatewayInfo structure.	ISO/IEC 13818-6 [12] (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 15.6.

Table 15.6: Restrictions on the DDB

Field	Restrictions	Source
moduleId	Module ids shall be unique within the scope of the object carousel.	Clause 11.2.3 ISO/IEC 13818-6 [12] (DSM-CC)

15.2.2.5 ModuleInfo

The moduleInfo structure shall be placed in the moduleInfo field of the DownloadInfoIndication (see clause 15.2.2.2) of the data carousel. It contains the information needed to locate the module.

Restrictions are shown in table 15.7.

Table 15.7: Restrictions on the DII moduleInfo field

Field	Restrictions	Source
moduleTimeOut, blockTimeOut, minBlockTime	These fields are defined in units of μ s. An appropriate value shall 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.	Clause 17.17.3. The present document
BIOP::ModuleInfo::Taps	The first tap shall have the "use" value 0x0017 (BIOP_OBJECT_USE). The id and selector fields shall not be 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 15.8.

Table 15.8: BIOP::ModuleInfo syntax

Syntax	Bits	Type	Value	Comment
BIOP::ModuleInfo() { moduleTimeOut blockTimeOut minBlockTime taps_count { { id use assocTag selector_length } for (j=1; j<N1; j++) { id use assocTag selector_length for (j=0; j<N2; j++) { selector_data } } userInfoLength for (k=0; k<N3; j++) { userInfo_data } }	32 32 32 8 16 16 16 8 16 16 16 8 8 8 8	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf	+ + + N1 0x0000 0x0017 + 0x00 + + + N2 + N3 +	≥ 1 user private BIOP_OBJECT_USE Possible additional taps that may be ignored by receivers.

15.2.2.6 ServiceGatewayInfo

The ServiceGatewayInfo structure shall be carried in the DownloadServerInitiate message and provides the object reference to the ServiceGateway object.

Restrictions are shown in table 15.9.

Table 15.9: 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 15.10.

Table 15.10: ServiceGatewayInfo() syntax

Syntax	Bits	Type	Value	Comment
ServiceGatewayInfo(){ IOP::IOR() downloadTaps_count for (i=0; i<N1; i++) { DSM::Tap() } serviceContextList_count for (i=0; i<N2; i++) { context_id context_data_length for (j=0; j<N3; j++) { context_data_byte } } userInfoLength for (i=0; i<N5; i++) { userInfo_data } }	8	uimsbf	+ N1	See table 15.23. software download Taps
	8	uimsbf	N2	serviceContextList
	32	uimsbf	N3	
	16	uimsbf	N5	user info
	8	uimsbf	+	

15.2.2.7 Download Cancel

There is no semantic for this message in the present document. 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 that shall be used by all the BIOP (Broadcast Inter-ORB Protocol) messages.

Restrictions are shown in table 15.11.

Table 15.11: 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 shall not be transmitted in object carousels.	ISO/IEC 13818-6 [12] (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 clause 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 15.12.

Table 15.12: Location of CORBA format strings

String	Length field size (bits)	Location
objectKind_data	32	Table 15.14
objectKind_data	32	Table 15.16
id_data	8	
kind_data	8	Table 15.18
objectKind_data	32	
objectKind_data	32	Table 15.20
eventName_data	8	
type_id_byte	32	Table 15.23
id_data	32	
kind_data	32	

15.2.3.3 BIOP FileMessage

The BIOP FileMessage shall be used for carrying file objects.

Restrictions are shown in table 15.13.

Table 15.13: Restrictions on the BIOP FileMessage

Field	Restrictions	Source
MessageSubHeader::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::ContentSize attribute. This is optionally followed by a loop of descriptors.	The present document
MessageSubHeader::ServiceContextList	The receiver may skip the possible serviceContextList structures.	The present document

Syntax is given in table 15.14.

Table 15.14: BIOP::FileMessage syntax

Syntax	Bits	Type	Value	Comment
BIOP::FileMessage() { magic biop_version.major biop_version.minor byte_order message_type message_size objectKey_length for (i=0; i<N1; i++) { objectKey_data } objectKind_length objectKind_data objectInfo_length DSM::File::ContentSize for (i=0; i<N2-8; i++) { objectInfo_data } serviceContextList_count for (i=0; i<N3; i++) { context_id context_data_length for (j=0; j<N4; j++) { context_data_byte } } messageBody_length content_length for (i=0; i<N5; i++) { content_byte } }	4 x 8 8 8 8 8 32 8 8 8 8 8 8 64 8 8 32 16 8 32 32 8	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf	0x42494F50 0x01 0x00 0x00 0x00 # N1 + 0x00000004 0x66696C00 N2 + N3 N4 + # N5 +	"BIOP" BIOP major version 1 BIOP minor version 0 big-endian byte ordering ≤ 4 "fil" type_id alias objectInfo serviceContextList actual file content

15.2.3.4 BIOP DirectoryMessage

The BIOP DirectoryMessage shall be used for carrying the directory objects.

Restrictions are shown in table 15.15.

Table 15.15: 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 15.16.

Table 15.16: BIOP::DirectoryMessage syntax

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 (see 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 15.13
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 [i.4])
}				
}				
}				
} BindingType	8	uimsbf	+	0x01 for nobject 0x02 for ncontext objectRef see table 15.23
IOP::IOR()			+	
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 clause 11.3.2.2, "Directory Message Format" in ISO/IEC 13818-6 [12] (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 in clause 15.2.3.4) 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

15.2.4.0 Selection of message type

There are two versions of stream messages. The BIOP StreamMessage shall be used for carrying stream objects that do not use DSM-CC stream events. The BIOP StreamEventMessage shall be 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 15.17.

Table 15.17: Restrictions on the BIOP StreamMessage

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	The ObjectInfo field contains the DSM::Stream::Info_T structure and optionally other data after the Stream Info structure. Within the present document there is no defined use of the DSM::Stream::Info_T structure and the possible other object info data following it. Receivers conforming to the present document shall ignore this information. Broadcasts may set the duration field to zero to indicate undefined duration.	The present document
MessageSubHeader::ServiceContextList	The receiver may skip the possible serviceContextList structures.	The present document
MessageSubHeader::MessageBody	The MessageBody carries a sequence of taps. 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). Receivers may ignore possible other taps (such as BIOP_ES_USE and STR_NPT_USE). See note.	The present document
NOTE:	Although the tap will be ignored, if present there shall be at most one instance of a STR_NPT_USE tap for compatibility with MHP.	

Syntax is given in table 15.18.

Table 15.18: BIOP::StreamMessage syntax

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 {				
aDescription_length	8	uimsbf	N3	objectInfo
for (i=0; i<N3; i++) {				
aDescription_bytes	8	uimsbf	+	aDescription
}				
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	+	
}				
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 ETSI TR 101 202 [i.4]
}				
assocTag	16	uimsbf	+	
selector_length	8	uimsbf	0x00	No selector
}				
}				

15.2.4.2 BIOP StreamEventMessage

15.2.4.2.0 Restrictions and syntax

Restrictions are shown in table 15.19.

Table 15.19: Restrictions on the BIOP StreamEventMessage

Field	Restrictions	Source
MessageSubHeader::ObjectInfo	<p>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).</p> <p>See table 15.17 regarding the DSM::Stream::Info_T. Receivers may ignore the possible other data following the DSM::Stream::EventList_T.</p> <p>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.</p>	The present document
MessageSubHeader::ServiceContextList	The receiver may skip the possible serviceContextList structures.	The present document
MessageSubHeader::MessageBody	<p>The MessageBody carries a sequence of taps followed by a sequence of event ids.</p> <p>The sequence of taps follows the following rules:</p> <ul style="list-style-type: none"> 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). 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. <p>Receivers may ignore possible other taps (such as BIOP_ES_USE and STR_NPT_USE).</p> <p>See note.</p>	The present document
NOTE:	Although the tap will be ignored, if present there shall be at most one instance of a STR_NPT_USE tap for compatibility with MHP.	

Syntax is given in table 15.20.

Table 15.20: BIOP::StreamEventMessage syntax

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 {				
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
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 ETSI TR 101 202 [i.4]
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 shall 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 shall be resolved during the ContentPreparation behaviour of the relevant MHEG-5 Stream object. Thus subsequent changes to the BIOP StreamEventMessage shall only be 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 shall be resolved during the ContentPreparation behaviour of the MHEG-5 Stream object. Thus subsequent changes to the BIOP StreamEventMessage shall only be 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 shall 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 shall resolve 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 that shall be supported in the present document are "do-it-now" events. During subscription, 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, the event shall be "re-fired". See clause 15.2.4.6.

15.2.4.4.2 Section number

In the present document receivers shall only consider section number zero.

15.2.4.4.3 Current_next_indicator

In the present document the current_next_indicator shall be set to 1 for DSM-CC Sections carrying Stream Descriptors (i.e. sections with a table_id of 0x3d).

15.2.4.4.4 Stream event life time

The set of stream events described in a particular BIOP::StreamEventMessage may 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 need not 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 15.21).

Table 15.21: 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 clause 5.5.2.2.1 in ISO/IEC 13818-6 [12]).

15.2.4.4.6 Resources to monitor stream events "do it now" events

"do it now" events are single shot events. Accordingly, receivers shall make special efforts to ensure that they can be reliably received.

Broadcasters shall be 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 are any active links waiting for such events.

See clause 17.12.

15.2.4.5 Stream descriptors

15.2.4.5.1 Stream event descriptor

In the present document 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 may only parse the section header in order to act upon the event. This stream event descriptor should be 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. An event is said to be "subscribed" when the MHEG-5 **Stream** object referencing the relevant **StreamEvent** message is running and at least one MHEG-5 **Link** object that captures the event is active. An event is "unsubscribed" when either the **Stream** object is no longer running or all **Links** that capture the event are deactivated. 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

15.2.5.0 Restrictions and syntax

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. In the present document, 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 15.22.

Table 15.22: 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 15.23.

Table 15.23: IOP::IOR syntax

Syntax	Bits	Type	Value	Comment
IOP::IOR { type_id_length for (i=0; i<N1; i++) { type_id_byte } taggedProfiles_count IOP::taggedProfile() for (n=0; n<N2-1;n++) { IOP::taggedProfile() } }	32	uimsbf	N1	
	8	uimsbf	+	Short alias type_id (e.g. "dir")
	32	uimsbf	N2	Profile bodies For objects in broadcast carousels: either BIOPProfileBody (see clause 15.2.5.1) or LiteOptionsProfileBody (see clause 15.2.5.2). Receiver may ignore other profiles (2 to N1) if present

15.2.5.1 BIOPProfileBody

The BiopProfileBody is used for references to objects within the same object carousel.

Restrictions are shown in table 15.24.

Table 15.24: Restrictions on the BIOP ProfileBody

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 μ s. An appropriate value shall 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 17.17.3.	The present document

Syntax is given in table 15.25.

Table 15.25: BIOP ProfileBody syntax

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++) {				
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.
				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 the present document, the following restrictions apply:

- LiteOptionsProfileBody shall only refer to objects within the same transport stream. Therefore, the use of the object carousel shall 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 15.26.

Table 15.26: 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 15.27.

Table 15.27: Syntax of LiteOptionsProfileBody with ServiceLocation component

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() CosNaming::Name() { nameComponents_count	160	uimsbf	+	See table 15.28. pathName
for (i=0; i<N2; i++) { id_length	32	uimsbf	N2	NameComponent id
for (j=0; j<N3 j++) { id_data	8	uimsbf	+	
}	32	uimsbf	N4	NameComponent kind
kind_length for (j=0; j<N4 j++) { kind_data	8	uimsbf	+	As type_id (see table 4-4 in ETSI TR 101 202 [i.4]).
} } initialContext_length	32	uimsbf	N5	
for (n=0; n<N5 n++) { InitialContext_data_byte	8	uimsbf		
} for (n=0;n<N6;n++) { BIOP::LiteComponent{ componentId_tag	32	uimsbf	+	N6=N1-1
component_data_length	8	uimsbf	N7	
for (i=0; i<N7; i++) { component_data_byte	8	uimsbf		
} } } }				

See table 15.28.

Table 15.28: DVB carousel NSAP address

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 shall 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.
dwb_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	
}				

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 ISO/IEC 13818-6 [12]. 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 the present document

The term "incremented" is used in the DSM-CC specification ISO/IEC 13818-6 [12]. Within the scope of the present document this shall be interpreted as "changed".

When a module is changed, the version number of the module shall be changed. This implies that the DownloadInfoIndication message that references the module shall also be updated. Since the DownloadInfoIndication is updated, the transactionId shall also be 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 are specified as follows.

The transactionId is split up into a number of sub-fields defined in table 15.29. 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 shall be independent of the expected filtering in target receivers.

Table 15.29: Sub-fields of the transactionId

Bits	Value	Sub-field	Description
0	User-defined	Updated flag	This shall be toggled every time the control message is updated.
1 to 15	User-defined	Identification	This shall be all zeros for the DownloadServerInitiate message. All other control messages shall have one or more non-zero bit(s).
16 to 29	User-defined	Version	This shall be incremented/changed every time the control message is updated.
30 to 31	Bit 30 - zero 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 shall cause the transactionId of that control message to be incremented. Any change to a Module shall necessitate incrementing its moduleVersion field. This change shall 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 shall be incremented to indicate a new version of the message. Also, any change in the DownloadServerInitiate message implies that its transactionId should also be incremented. However, when the transactionId is divided into subfields as specified above, updating a message shall change only the Version part of the transactionId while the Identification part shall remain 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 references should not be updated every time the control message is updated. 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 should 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 17.20.2.1).
- 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 shall be 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.

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 IETF RFC 1950 [i.1].

Table 15.30 shows the syntax of the compressed_module_descriptor.

Table 15.30: compressed_module_descriptor

Syntax	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	
}			

Table 15.31 shows the syntax of the ZLIB structure.

Table 15.31: zlib structure

Syntax	No. of bytes
zlib structure(){	
compression_method	1
flags_check	1
compressed_data	n
check value	4
}	

The receiver shall support the Deflate compression algorithm as specified in IETF RFC 1951 [25]. This shall be signalled by setting the least significant nibble of the `compression_method` to 0x8 (i.e. `compression_method` is xxxx1000). The receiver need not support other compression algorithms.

15.3 AssociationTag mapping

15.3.1 Association tags in "taps"

The ISO/IEC 13818-6 [12] 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 present document, taps shall be 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 should 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 shall use the BIOP_PROGRAM_USE tap to identify a service (see clauses 15.2.4.1 and 15.2.4.2). This tap contains a DSM-CC association tag value that can 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, shall be 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 shall resolve 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 may choose any valid transport stream ID.

15.3.3.3 Service association tag mapping decision tree

See figure 15.1.

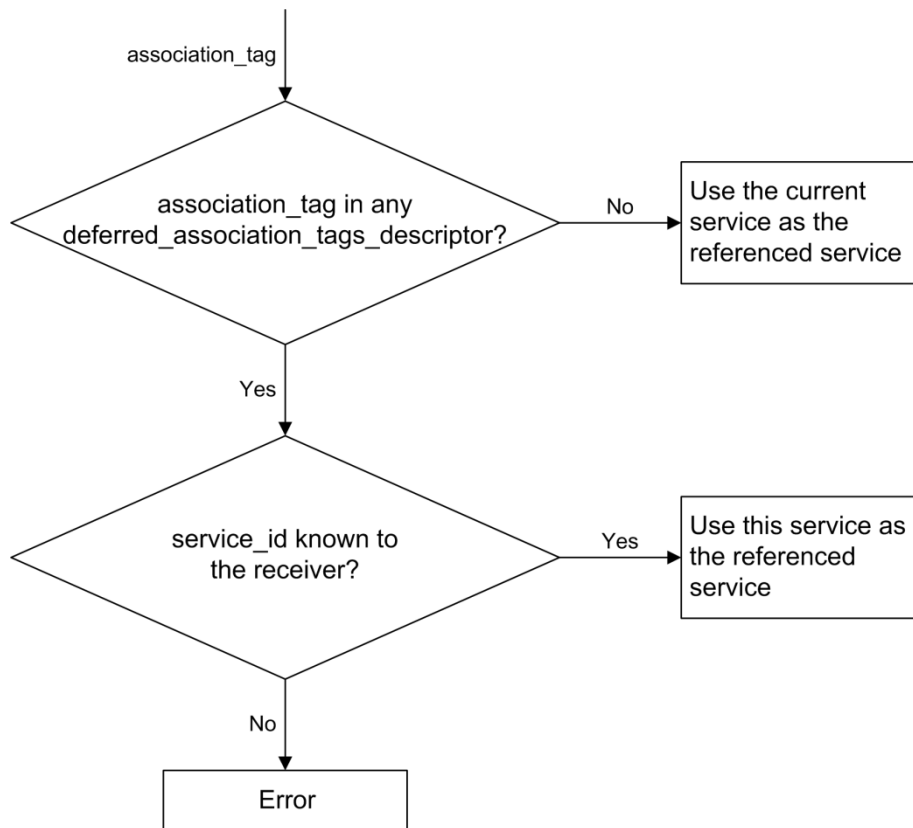


Figure 15.1: Service association tag mapping decision tree

15.3.4 Using an association tag to reference an elementary stream

15.3.4.1 MHEG-5 ComponentTags to DSM-CC association tags

15.3.4.1.0 Selecting an elementary stream from a ComponentTag

The MHEG-5 ComponentTag attribute of MHEG-5 Audio or Video objects shall be used to select the elementary stream which is to be decoded within the service indicated by the enclosing MHEG-5 Stream object (see clause 14.2.6). The two least significant bytes of the MHEG-5 ComponentTag shall 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" (see table 16.1), 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" (see table 16.1). Receivers may ignore the value of ComponentTag in this case.

See also clause 14.2.6.2.

15.3.4.1.2 Explicit component references

Receivers shall 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 shall be responsible for ensuring that the components carry data suitable for the MHEG-5 stream component type.

Receivers shall, by default, use the stream_type information from the PMT to determine the correct codec to use to decode the stream component. Where the stream_type does not indicate the correct codec to be used (for example, if the stream_type indicates private PES data), the receiver shall use the service_type value for the relevant service to determine the codec, as shown in table 15.32.

Table 15.32: Codecs for stream decoding

Service type	MHEG stream component	Codec
0xA, 0x16, 0x19	Video	H.264
	Audio	See below
0x1, 0x2, 0xC	Video	MPEG-2
	Audio	MPEG-1 layer II

The service_type shall be that listed in the service_descriptor of the SDT.

If the service_type value is 0xA, 0x16 or 0x19, receivers shall determine the audio codec as follows:

- If the PMT contains an Enhanced_AC-3_descriptor (descriptor tag 0x7A) for the stream component, the Enhanced AC-3 codec shall be used.
- If the PMT contains an AAC_descriptor (descriptor tag 0x7C) for the stream component, the receiver shall use the AAC, HE-AAC or HE-AACv2 codec according to the AAC_type value indicated in the descriptor.

If necessary, receivers may additionally make use of information from a component_descriptor, if present.

15.3.4.1.3 Mapping Errors

If the receiver is unable to map an MHEG-5 ComponentTag in an MHEG-5 Stream object to a DVB component, presentation of that element of the Stream object shall cease (i.e. if there is a component tag mapping error for the video it shall display "black").

15.3.4.2 Mapping DSM-CC association_tags to DVB component_tags

15.3.4.2.0 Selecting a component_tag from an association_tag

The DVB component_tag is an 8-bit value that may be used to identify an elementary stream without directly referring to its PID value. Likewise, 16-bit association_tags may be used, as defined in the ISO/IEC 13818-6 [12] (DSM-CC) specification, 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 the present document, the DVB stream_identifier_descriptor shall always be used for assigning a DVB component_tag for the elementary streams. It 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

In the present document receivers need not use the deferred_association_tags_descriptor to match elementary streams. All components that constitute a valid carousel broadcast to the present document shall be present in the PMT from which the carousel was mounted.

Receivers may 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 15.2.

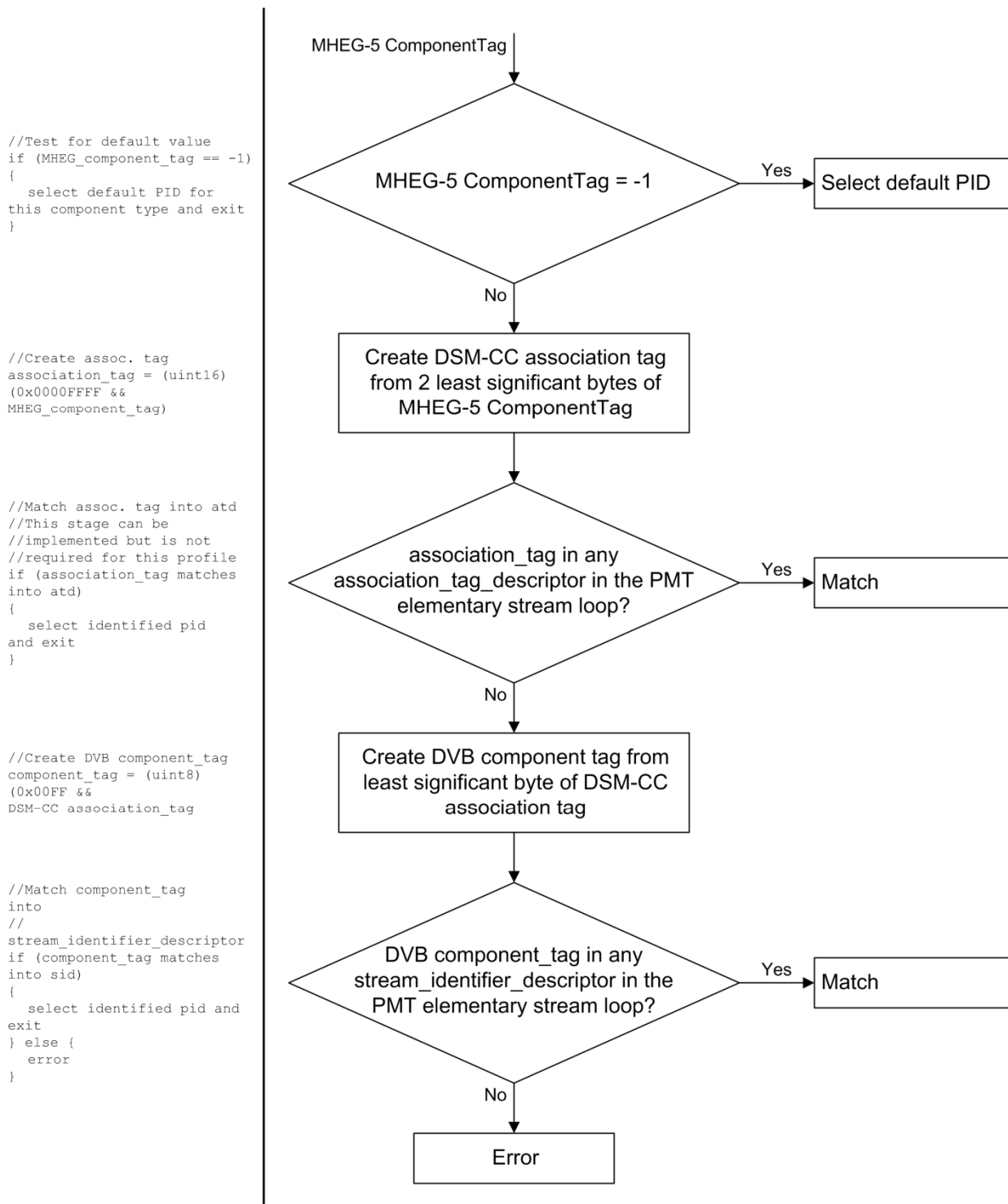


Figure 15.2: Elementary stream mapping pseudo code and decision tree

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) shall be 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 15.33.

15.4.2 Determining file version

15.4.2.0 Use of moduleVersion for 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 shall 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 shall make before returning cached content and suggests how the content could be cached thereafter. A `ContentCachePriority` value shall be associated with a piece of content until overridden by another value.

See table 15.33.

Table 15.33: 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. 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 shall 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	The ISO/IEC 13522-5 [14] (MHEG-5) specification states that when an object is declared with <code>ContentCachePriority</code> of zero: "Specific value: 0 means caching is not allowed for external content data referenced by this Ingredient." Therefore, when the action <code>SetData</code> is targeted at an object with <code>ContentCachePriority</code> of zero, it shall be guaranteed to fetch the content from the broadcast stream rather than from any cache. 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 shall 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 17.17.1.

15.6 IC file system

Receivers shall be able to retrieve content from a server over an IP connection using HTTP; such content shall be identified using an http URI. Receivers shall be able to retrieve content from a server over an IP connection using HTTP with security provided by TLS; such content shall be identified using an https URI. See clause 16.3.2.3.4.

Receivers shall implement an IC file system that accesses content identified by valid http and https URIs (as defined by IETF RFC 2616 [30], IETF RFC 2818 [31] and IETF RFC 3986 [28]). Given a suitable URI, receivers shall respond to a request to read content from this file system by connecting to the server specified by the URI and sending an HTTP GET request for the required file.

15.7 MHEG profile of HTTP

15.7.0 Profile of HTTP

Receivers shall support HTTP/1.1, as defined in IETF RFC 2616 [30], except as specified below. The profile of HTTP/1.1 defined in the present document shall be used for access to both static files and IP delivered streams.

15.7.1 Protocol parameters

Receivers need not make use of character sets, content codings and language tags. Receivers shall accept responses with the chunked transfer coding but need not make use of transfer codings.

Receivers need not make use of entity tags (see clause 15.10).

Receivers need not support Access Authentication.

Receivers need not support Content Negotiation.

Receivers need not transmit media type information, however the ReturnData resident program shall as a minimum transmit the following header field:

- Content-Type: application/x-www-form-urlencoded

Receivers may ignore media type information sent by servers. Application providers should ensure that the content of the response is valid for the target MHEG object.

15.7.2 Methods

For standard file requests, receivers shall access content using the GET method. The HEAD method should only be used by the reference checking resident programs (see clause 11.10.9). The ReturnData resident program (see clause 11.10.12.2) shall use the POST method. Receivers shall not use the PUT and DELETE methods. Receivers should not use any other method.

15.7.3 Response status codes

The response to the status codes (specified in HTTP/1.1) returned by the server shall be as follows:

- Informational 1xx: this class of status code indicates a provisional response and receivers shall continue with the request being made. The request shall not be considered to be completed. Where a request is made by the ReturnData resident program, these status codes shall not be returned to the application.
- Successful 2xx: the request shall be considered to be completed successfully.
- Redirection 3xx: receivers shall follow the redirection according to IETF RFC 2616 [30]. However, if the request method was POST (i.e. the request was made by the ReturnData resident program), no redirection shall occur as this would require interaction with the user. In this case, the request shall be considered to have failed and the 3xx status code shall be returned to the application. In order to prevent infinite redirection loops, a maximum of 5 redirections shall be followed. Receivers shall support redirection where the new URI scheme is different from the initial request (for example "https" instead of "http").
- 201 Created is an error and the request shall be considered to have failed.
- 206 Partial Content: this code may be used in response to a Range request for Stream content delivered by IP.
- 300 Multiple Choices is an error and the request shall be considered to have failed.
- 306 (Unused) is an error and the request shall be considered to have failed.
- All other values, including unrecognized responses, are an error and the request shall be considered to have failed.

15.7.4 Header fields

15.7.4.1 Request header fields

15.7.4.1.0 Header fields that need not be sent

Receivers need not send any of the following header fields:

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization
- Expect
- From

- Max-Forwards
- Proxy-Authorization
- Referer
- TE

Receivers that use only the If-Modified-Since header field for caching (see clause 15.10) need not transmit the following header fields:

- If-Match
- If-None-Match
- If-Unmodified-Since

15.7.4.1.1 User agent string

Receivers shall transmit three additional tokens in the User-Agent header field. These tokens shall appear consecutively in the order they are described below. The tokens are as follows:

- 1) The first product token shall be the string "UK-MHEG/" followed by the largest value of N which would generate a "true" response in the UEP(N) GetEngineSupport request (see clause B.2 or clause 11.4.1).
- 2) The second product token shall be a string that uniquely identifies the receiver via its make, model and version number. This string shall be as defined in clause 11.4.1.3 but with a "/" character before the version number, i.e. "mmmccc/vvv".
- 3) The third product token shall be a string that identifies the MHEG engine provider and version number. This string shall be as defined in clause 11.4.1.3 but with a "/" character before the version number, i.e. "MHGmmm/vvv".
- 4) Receivers may add further product tokens as defined by section 3.8 of IETF RFC 2616 [30].

EXAMPLE: A particular receiver might transmit the header as: User-Agent: UK-MHEG/2 FEG001/103 MHGFEG/056.

15.7.4.2 Response header fields

Receivers may ignore the following header fields:

- Proxy-Authenticate.
- Server.
- WWW-Authenticate.

Receivers that use only the If-Modified-Since header field for caching (see clause 15.10) may ignore the following header fields:

- Age.
- ETag.

15.7.4.3 General header fields

Receivers need not transmit, and may ignore, the following header fields:

- Pragma, except as specified by clause 15.10.
- Trailer.
- Upgrade.
- Via.

Receivers shall accept a response that includes the Transfer-Encoding header field to indicate the use of chunked transfer-coding but otherwise may ignore the Transfer-Encoding header field. Receivers need not transmit the Transfer-Encoding header field.

Receivers that use only the If-Modified-Since header field for caching (see clause 15.10) need not transmit, and may ignore, the following header fields:

- Warning.
- Cache-Control (except as specified by clause 15.10).

15.7.4.4 Entity header fields

Receivers need not transmit, and may ignore, the following header fields:

- Allow.
- Content-Encoding.
- Content-Language.
- Content-Location.
- Content-MD5.
- Content-Type (except as specified for the ReturnData resident program by clause 15.7.1).

Receivers that use only the If-Modified-Since header field for caching (see clause 15.10) need not transmit, and may ignore, the following header fields:

- Expires.

15.7.5 Cookie support

Receivers shall support the use of cookies via the Cookie request header and Set Cookie response header as defined by IETF RFC 6265 [32]. Receivers need not store cookies beyond a power cycle, regardless of any specified expiration date or age.

Cookies shall be retained until the auto-boot process begins (clause 9.3.4.2). Receivers shall provide storage for at least 32 cookies with a minimum combined size of 8 192 bytes of data. Receivers may discard older cookies to make space for newer ones, using, for example, a least recently-used algorithm, along with constraints on the maximum number of cookies that each origin server may set.

For compatibility with existing HTTP/1.0 servers, receivers are recommended to also accept cookies in other common formats such as "Netscape" format.

Cookies with the secure attribute specified shall only be sent over a secure connection (i.e. a connection using TLS) (clauses 11.10.12.6 and 11.10.12.7).

Cookies created using the SetCookie Resident Program shall use the IETF RFC 6265 [32] format.

The receiver need not share cookies between HTTP operations related to the InteractionChannelExtension and those related to the ICStreamingExtension (including ICEncryptedStreamExtension).

15.8 Connection setup

Receivers shall be able to:

- resolve host names;
- establish HTTP, HTTP/TLS connections.

The infrastructure that supports this is outside the scope of the present document.

15.9 Multiple connections

A priority mechanism for arbitrating between multiple uses of the interaction channel is not defined. The number of connections that can be kept open simultaneously is not specified in the present document. Receivers shall adopt a policy to manage any such restriction, queuing further content requests, should any implementation-specific limit be reached.

The present document does not specify when a receiver should close a keep-alive connection. This behaviour is implementation-dependent.

15.10 HTTP caching

Receivers shall support the same cache behaviour as specified in the object carousel profile (see clauses 15.4.1, 15.4.3 and 15.4.4), except that the words "broadcast stream" and "broadcast carousel" shall be replaced by "interaction channel".

The HTTP/1.1 caching specification (see section 13 of IETF RFC 2616 [30]) allows clients to use a number of optional techniques for working with cached data. As a minimum, receivers shall at least use the If-Modified-Since and Last-Modified header fields to implement a transparent cache.

When ContentCachePriority or GroupCachePriority is even and non-zero, the receiver shall not make a request to a server if it holds a cached copy of the data. If a request is made, the receiver shall generate a request with the following characteristics:

- A Cache-Control: max-stale no-transform header shall be present.

When ContentCachePriority is odd, or when GroupCachePriority is odd or zero, the receiver shall generate a request that will ensure that up-to-date content is retrieved. The request shall have the following characteristics:

- Appropriate header fields shall be present to ensure that up-to-date content is retrieved, as specified by section 13 of IETF RFC 2616 [30].
- If the request is conditional and no other header field relating to caching is present, an If-Modified-Since header field shall be present.
- A Cache-Control: no-transform header field shall be present.

When ContentCachePriority is zero, the receiver shall cause cached copies of the resource (if present) to be invalidated and shall generate a request with the following characteristics:

- An If-Modified-Since header field shall not be present.
- A Cache-Control: no-cache no-transform header field shall be present.
- A Pragma: no-cache header field shall be present if the server is not known to be HTTP/1.1 compliant.

15.11 Timeouts

The receiver shall consider the file request to have failed if an attempt to initially establish a connection to a server takes more than a timeout period. This timeout period shall be at least 30 seconds.

The receiver shall not time out whilst waiting for a server to communicate over an established connection after an HTTP request has been sent.

Where a file transfer from a server has started, the receiver shall consider the transfer to have failed if it does not complete within a timeout period. This timeout period shall be at least 30 seconds.

These timeouts shall also be used for requests using the ReturnData ResidentProgram.

15.12 Hybrid file system

15.12.1 Introduction

Receivers shall implement a "virtual" hybrid file system, which takes file requests and passes them on to one of the actual file systems available to the receiver, such as the broadcast carousel or the IC file system.

The hybrid file system is modelled as a mapping table. Each mapping in the table shall define a name within the hybrid file space which maps to one or more target names in the file systems available to the receiver.

When content is requested from the hybrid file system, there shall be first a resolution procedure to determine which of the file systems available to the receiver should be used and what request should be made to it. This resolution procedure shall involve a search of the mapping table and some manipulation of the string values contained within the table. Only after this procedure is complete shall the hybrid file system make a request to one of the underlying file systems.

15.12.2 Resolution of References

A hybrid file reference of the form defined in clause 16.1 shall be resolved as follows:

- 1) The Source ("hybrid:") is removed.
- 2) The reference produced by step 1 is processed according to clause 16.3.2.3.
- 3) The mapping table is searched for mappings that match the reference produced by step 2. A match is made as follows:
 - If the mapping corresponds to a file pathname (i.e. the name does not end with a "/") then the name shall match the entire reference.
 - If the mapping corresponds to a directory pathname (i.e. the name does end with a "/") then the name need only match the leftmost part of the reference.

EXAMPLE: If the reference used for a request is "hybrid://A/B/C1/D" then the mappings:

- //A/, //A/B/C1/ and //A/B/C1/D (should they exist) would all match;

while the mappings:

- //B/, //A/X/C1/D, //A/B/C1/X, //A/B/C, //A/B/C/, //A/B/C1 and //A/B/C1/D/X (should they exist) would not match.

- 4) The mapping with the most complete match, i.e. with the longest matching name if there is more than one, shall be used to resolve the requested reference.
- 5) There shall be a default mapping associated with the name //, which matches all hybrid references. The default mapping shall map the name // to DSM://. If there is no other matching name and no mapping for the name // has been defined, the default mapping shall be used. For example, if no name matches hybrid://A/B/C/D then that reference shall be resolved as DSM://A/B/C/D.
- 6) Where a hybrid name is mapped to more than one target name, content shall be requested using each target name in order until a request is successful. If content is not available from the location resolved from the first target name, the next will be tried and so on until content is available from a location or all target names have been tried. This behaviour shall not be affected by the value of ContentCachePriority or GroupCachePriority.

NOTE: Different file systems may return different data if asked to resolve a particular request. This provides a convenient means to adapt the behaviour of a running MHEG application depending on the available data delivery means.

15.12.3 Resolution Example

Assume the table contains:

Mapping 1: // is mapped to {DSM://}

Mapping 2: //A/ is mapped to {DSM://foo/, <http://www.com/foo/>}

Mapping 3: //B/ is mapped to {<http://www.com/bar/>}

Mapping 4: //B/C/ is mapped to {<http://www.com/baz/>, DSM://}

When a request is made for "hybrid://B/my_file", mapping 1 and mapping 3 are both considered as matches. Mapping 2 and mapping 4 are not matches because "//B/my_file" does not begin with the string "//A/" or the string "//B/C/".

The chosen mapping shall be the most specific one, which shall always be the one with the longest pathname in the hybrid file space. In this case, mapping 3 is more specific than mapping 1. Having chosen mapping 3, a request for http://www.com/bar/my_file shall be made.

When a request is made for "hybrid://my_file", only mapping 1 matches, so a request shall be made for "DSM://my_file".

When a request is made for "hybrid://B/C/my_file", mappings 1, 3 and 4 all match, with mapping 4 being chosen as the most specific.

A request is made for "http://www.com/baz/my_file". If this request is unsuccessful, a request shall then be made for "DSM://my_file". If this second request is also unsuccessful, the request to the hybrid file system shall be considered to have failed.

NOTE: The present document does not mandate a particular implementation. Hybrid references may be resolved by any method. One alternative is to "prune" the requested reference to make less specific references until a string that exists in the mapping table is found. With this method, a request for "//A/B/my_file" would consider "//A/B/my_file", then "//A/B/", then "//A/". At this point, the search would stop and mapping 2 would be selected, leading to a request for "DSM://foo/B/my_file".

15.12.4 Hybrid file system caching

The hybrid file system shall not itself implement caching of files, rather it shall present the results of the caching implemented in the underlying file systems and reflect the effect of the value of ContentCachePriority or GroupCachePriority as it applies to those file systems.

15.12.5 Synchronous file loading

Resolution of hybrid references shall take place within the synchronous part of the ContentPreparation behaviour of an MHEG object.

15.12.6 Size of mapping table

Receivers shall provide storage for at least 32 mappings each of which is capable of storing at least 2 048 bytes.

15.12.7 Configure mapping table

The mapping table shall be configured for each MHEG application through the MHEG ResidentProgram SetHybridFileSystem (see clause 11.10.13.1).

15.12.8 Interaction with security (informative)

The hybrid file system does not itself implement security for MHEG code and content. Instead, the security level of each underlying file system shall be applied. The following example (see figure 15.3) shows how a signed application might be loaded through the hybrid file system.

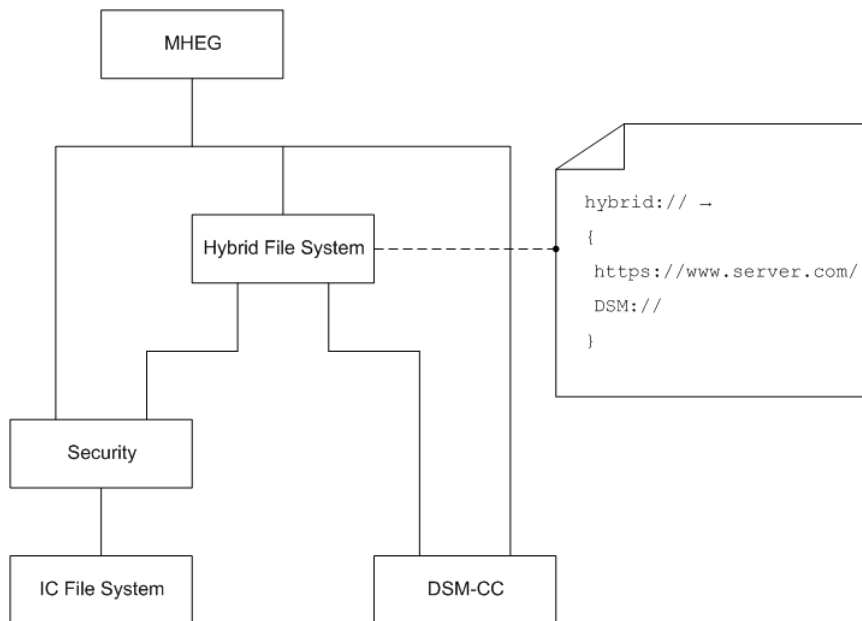


Figure 15.3: Hybrid File System Signed Application

- 1) An application transitions to `hybrid://scene.mhg`.
- 2) The appropriate mapping is looked up in the hybrid mapping table. `hybrid://` is mapped to `{https://www.server.com, DSM://}` in that order. This mapping is stored in memory until the file request is complete.

Attempt to load file from first location:

- 3) `DSM://auth.servers` is retrieved. This server list file permits access to `https://www.server.com/scene.mhg`.
- 4) `https://www.server.com/scene.mhg` is retrieved using the TLS certificate in `DSM://auth.tls.1`.
- 5) `https://www.server.com/auth.hash` is retrieved (using the server list file and TLS certificate as before). The hash file exists, its `digest_count` is greater than zero and `scene.mhg` is listed with `digest_type 2`.
- 6) The digest value is successfully checked against the contents of `scene.mhg`.
- 7) `https://www.server.com/auth.sig.1` is retrieved (again using the server list file and TLS certificate).
- 8) `DSM://auth.cert.1` does not contain a certificate that matches the signature. There are no other certificates in the object carousel and no other signature files on the server. Authentication has failed.

Attempt to load file from second location:

- 9) The second location is looked up from the mapping that was stored in memory.
- 10) `DSM://scene.mhg` is retrieved and can be returned to the MHEG engine.

EXAMPLE: See clause 17.25.1 for an example showing the use of the `SetHybridFileSystem` resident program.

15.13 Authentication of applications

15.13.1 Overview (informative)

This clause defines a mechanism for authentication of applications. This ensures that all interaction channel content accessed when the receiver is tuned to a particular broadcast service is authorized by the provider of that broadcast service.

The basic method by which this is achieved is by signing files with a private key. The corresponding public key is then placed in the broadcast carousel of any service from which access to the signed files is required.

Since the verification of public key signatures is time consuming, the present document allows for a number of files to be signed with a single signature. This is achieved by computing a cryptographic hash for each file to be signed, and then signing a file listing those hashes.

Different sources of content in a digital TV receiver have different inherent levels of security. For example, a broadcast delivery mechanism might be considered more secure than an Internet delivery mechanism.

15.13.2 Authentication levels

The present document defines the level of authentication required for each file system available to the receiver. These authentication levels are defined in table 15.34.

Table 15.34: Application authentication levels

Level	Check authentication?	Action if verification successful	Action if verification unsigned	Action if verification fails
1	No	N/A - files always to be loaded		
2	Yes	Load file	Load file	Load file, disable interaction channel
3	Yes	Load file	Load file, disable interaction channel	Load file, disable interaction channel
4	Yes	Load file	Load file	Fail
5	Yes	Load file	Fail	Fail
6	No	N/A - file accesses always fail		

NOTE: Rows shaded in grey (levels 2, 3 and 6) are not supported in the present document.

The security levels for the various file systems shall be as defined in table 15.35.

Table 15.35: File system security levels

File system source	Requests for MHEG code	Requests for content
CI	1	1
DSM	1	1
http	5	4
https	5	4
hybrid	Level of underlying file system applies	Level of underlying file system applies

Interpretation: Authentication shall be assumed for content obtained from the carousel. It is expected that all application code obtained from the Internet will be signed; content obtained from the Internet may be signed. Internet code and content that fails verification shall not be loaded. IP delivered streamed content shall not be signed.

15.13.3 Hash files

Each directory containing files that the author wishes to be authenticated shall contain a single hash file. The hash file shall contain a hash value for each of the authenticated files in the directory. Any files that are not to be authenticated shall be listed in the hash file but shall have no hash value.

An entire directory can also be marked as non-authenticated.

The hash computation considers the content of the files rather than transport specific information. As a result, the authentication is independent of the underlying transport protocol.

15.13.4 Signatures

The hash files described in clause 15.13.3 provide a concise summary of the content in a directory. In order to prove that content has not been tampered with, this summary shall be signed with the private key of a signatory that is trusted by the receiver. Such signatures shall be stored in one or more signature files within each directory to be authenticated.

NOTE: Since public key signature mechanisms can only sign small quantities of data, in fact the hash file to be signed is first hashed again to produce a small, fixed size digest. It is the digest that is then encrypted with the signatory's private key to create the signature.

Different signatories can sign different parts of an interactive service. Some parts of a service can also be signed by more than one signatory. Therefore, there may be more than one signature file present in a directory. A receiver need only ensure that files needing to be authenticated have been signed by at least one signatory that is trusted.

Each signature file shall:

- reference a certificate containing the public key required to decode the signature;
- identify the hash algorithm used;
- contain the value of the signature.

15.13.5 Application signing certificates

A certificate is a signed file that binds the identity of a signatory to the signatory's public key. Certificates are used to create a 'chain of trust' up to an entity that is implicitly trusted by the receiver.

Receivers shall use certificates to check the signatures described in clause 15.13.4.

A certificate file contains a chain of one or more certificates ending with a self-signed certificate. Application signing certificates placed in the carousel shall be deemed valid.

Figure 15.4 illustrates how a chain of certificates and a signature shall be used.

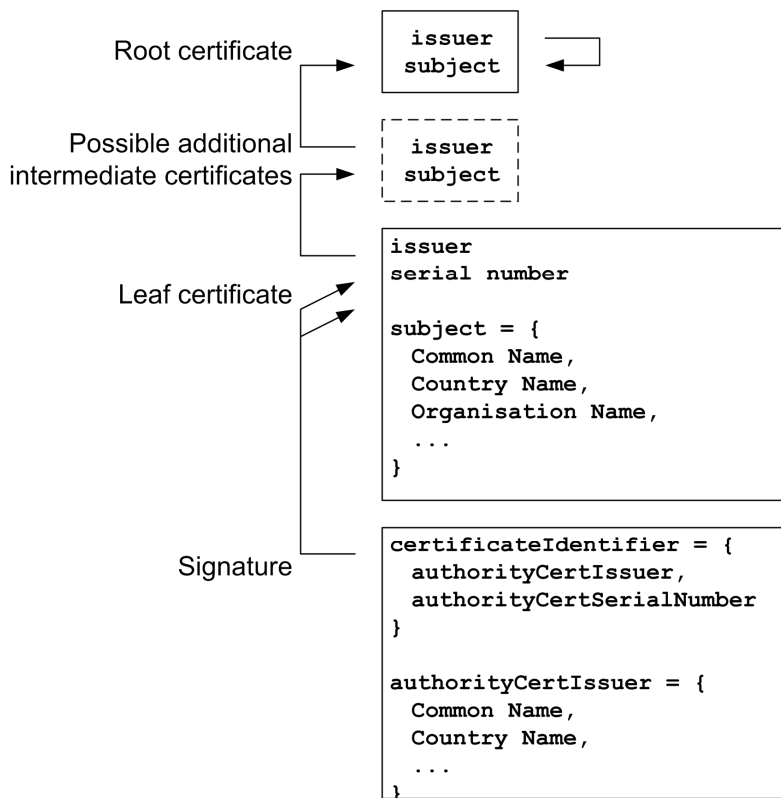


Figure 15.4: Chain of certificates

15.13.6 Authentication process

15.13.6.0 Introduction

This clause defines the process by which receivers verify the authenticity of files. This description references the syntax defined in clause 15.13.7. The process shall only be used for accesses to file systems for which the authentication level requires it. The process has two parts: verification of hash files and verification of signatures. It leads to one of a number of outcomes, the effects of which are defined by the authentication level in force. The possible outcomes are:

- file authentication successfully verified;
- file verified as being unsigned;
- verification failure.

15.13.6.1 Hash file verification

The hash file verification process is as follows:

- 1) retrieve the hash file from the same directory as the requested file (the 'current' directory);
- 2) if no hash file exists, return a verification error;
- 3) if the hash file has `digest_count <> 0`, perform the following steps:
 - a) if the requested file is not listed in the hash file, return a verification error;
 - b) if the file is listed and has a zero digest type, the file shall be verified unsigned subject to signature verification; proceed to step 5;
 - c) if the file is listed and has a non-zero digest type, perform a consistency check of the file contents against the corresponding digest value;
 - d) if the file contents are not consistent with the digest, return a verification error;
 - e) otherwise the file shall be verified subject to signature verification; proceed to step 5;
- 4) if the hash file has `digest_count = 0`, perform the following steps:
 - a) if the pathname encoded in the hash file does not contain at least one path element, return a verification error;
 - b) if the pathname elements in the hash file do not match the path to the current directory, return a verification error;
 - c) otherwise, the file (along with all the files in the directory) shall be verified unsigned subject to signature verification; proceed to step 5;
- 5) proceed to signature verification for the hash file.

15.13.6.2 Signature verification

The process of signature verification is as follows:

- 1) retrieve a signature file from the current directory;
- 2) search the available certificates provided in the root directory of the DSM-CC carousel for one that matches the issuer and serial number information in the signature file;
- 3) if no matching certificate is found, verification for this signature has failed; repeat from step 1 for the next signature file (if any);
- 4) use the certificate to verify that the signature correctly signs the hash file;
- 5) if the signature is incorrect, verification for this signature has failed; repeat from step 1 for the next signature file (if any);

- 6) if none of the signatures present can be verified correctly, return a verification error.

15.13.6.3 Optimizations

Signature verification can be time consuming. If the receiver has previously determined that the signature is authentic and has confirmed that the relevant certificate and signature files have not changed then the signature file need not be re-verified.

Implementers should use caution when considering other optimizations without diligent security audit.

15.13.6.4 Certificate caching

Subject to the constraints defined below, receivers may cache certificates to avoid loading them from the carousel each time a signature requires verification and to allow a cache of signature verifications to be maintained for each certificate.

Receivers shall verify that any cached copy of a broadcast certificate is consistent with the file being broadcast each time the application auto-boot process begins as described in clause 9.3. This verification process shall include performing a complete binary comparison of the broadcast file with the cached copy and shall be completed before any use of the cached certificate is made following the start of the auto-boot process.

In addition, receivers shall not allow a cached copy of a certificate to be used more than four hours after that certificate was last verified as being up to date. For the purpose of these additional checks, it shall be sufficient to verify that the module containing the certificate and the module containing the service gateway have not changed.

Where a signature to be checked references a certificate that is not available from cache, receivers shall check for the certificate in the carousel even if previous attempts to load the certificate have failed.

15.13.6.5 Handling race conditions

Hash verification and signature verification require data from two or more files to be compared. Since these files may be dynamic, verification can fail if attempted at the same time as the files are being updated.

If verification fails, receivers shall retry at intervals of 0,5 seconds as long as one of the relevant files continues to change. If, upon retrying, the files contain the same content as on the previous attempt, verification shall have failed and no further attempt shall be made.

To minimize the possibility of such retries, receivers should retrieve the files needed for verification within as short a period of time as possible. Similarly, authors should aim to update all the relevant files at the same time.

15.13.6.6 Redirection

When a file request results in one or more redirections, authentication shall be performed on the final resource loaded. I.e. the 'current' folder, and the source of the hash and certificate files used for authentication shall be the one indicated in the final redirection header as the source of the file.

15.13.7 Authentication file formats

15.13.7.1 Hash file

15.13.7.1.0 Hash file structure

The hash file normally lists all of the files present in the relevant directory except itself. Signature files may be listed, in which case they shall have a `digest_type` of `non-authenticated`. Those elements to be authenticated are associated with hash codes. Subdirectories may be listed but such entries are ignored.

The directory that contains the hash file can be marked as unsigned in its entirety by setting `digest_count` to zero. In this case, the hash file shall include part of the pathname of the directory, as described below.

The syntax of the hash file is shown in table 15.36.

Table 15.36: Hash file syntax

Syntax	Bits	Type
Hashfile () {		
digest_count	16	uimsbf
for(i=0; i<digest_count; i++) {		
digest_type	8	uimsbf
name_count	16	uimsbf
for(j=0; j<name_count; j++) {		
name_length	8	uimsbf
for(k=0; k<name_length; k++) {		
name_byte	8	bslbf
}		
}		
for(j=0; j<digest_length; j++) {		
digest_byte	8	bslbf
}		
}		
if (digest_count == 0) {		
pathname_length	16	uimsbf
for(i=0; i<pathname_length; i++) {		
pathname_byte	8	bslbf
}		
}		
Other data may follow but can be ignored by implementations conforming to the present document.		

digest_count: This 16 bit value identifies the number of digest values in this hash file.

digest_type: This 8 bit value identifies the digest computation rules and the digest algorithm, if any, used for the associated objects. Table 15.37 lists the allowed values for this field. The digest computation rules are defined in clause 15.13.7.1.3.

Table 15.37: digest_type_values

value	digest length	algorithm
0	0	Non-authenticated
1	n/a	Reserved
2	20	Digest computation rules without prefix and with SHA-1 as defined in FIPS-180-1 [49]
3	20	Digest computation rules with prefix and with SHA-1 as defined in FIPS-180-1 [49]
other values		Reserved for future use

name_count: This 16 bit value identifies the number of object names associated with the digest value. The value of this field shall be equal to one.

name_length: This 8 bit value identifies the number of bytes in the object name.

name_byte: This 8 bit value holds one byte of the object name.

Each name shall be the name of an object in the directory that contains the hash file. So, file names are the names of files in the directory and directory names are the names of direct subdirectories of the directory. No path information shall be included in the name.

The names carried by this field are binary identical to the payload part of names in the file system. So, any name matching process can be binary and ignorant of character encoding, letter case etc. Also, terminating null characters are not considered to be part of the file name.

digest_length: This integer value gives the number of bytes in each digest value. It depends upon the digest type, see table 15.37. Receivers shall support all digest algorithms.

digest_byte: This 8 bit value holds one byte of the digest value. See clause 15.13.7.1.3.

pathname_length: This integer value gives the number of bytes in the path name.

pathname_byte: This 8 bit value holds one byte of the pathname.

15.13.7.1.1 Hash file pathname matching

The pathname bytes are mandatory if `digest_count` is zero.

A pathname is illustrated by the following pattern:

- `scheme://hostname/path element/path element...`

For the purposes of the following text "scheme" and "hostname" shall be treated identically to other path elements. Also, the scheme separator "://" shall be treated the same as the normal directory separator "/" (0x2F). So, the pattern above shall be considered as if it were the following:

`path element/path element/path element/path element...`

There shall be at least one path element.

".." is not a valid path element and shall not be present.

Trailing directory separator slashes are optional and shall be ignored by receivers.

A hash file shall be authenticated by the matching all of the pathname from the hash file with the rightmost path elements of the path to the hash file.

When matching pathnames from the hash file the pathname shall be compared using a binary comparison of complete path elements.

EXAMPLE: The file: "<http://www.server.com/applications/app1/auth.hash>" could contain a pathname of any of the following:

- `app1`
- `applications/app1`
- `www.server.com/applications/app1`
- <http://www.server.com/applications/app1>

All of these validate the location of the hash file with varying degrees of precision.

Whereas the hash file: "<http://www.server.com/auth.hash>" could only contain one of the following:

- `www.server.com/`
- `http://www.server.com/`

15.13.7.1.2 Hash file location and naming conventions

A hash file shall be mandatory for each directory containing objects that need to be authenticated.

The name of the hash file shall be:

`auth.hash`

There shall only be one instance of hash file per directory that contains authenticated resources. See also clause 15.13.8.

15.13.7.1.3 Digest value computation rules

The digest value is computed over the object named in the hash file. The relevant data for each object shall depend on the value of `digest_type` and is specified in table 15.38.

In the present document, the method for calculating a digest of a directory is undefined.

Table 15.38: Data required for digest value computation

Digest type	entry_type	Relevant data
1	Reserved	
2	not applicable	The entire content of the file.
3	1	A prefix concatenated with the entire content of the file. The prefix is made of the entry_type encoded as a 32 bit uimsbf and concatenated with the file length in bytes encoded as a 32 bit uimsbf.
NOTE: All other values of entry_type are reserved for future use.		

EXAMPLE: Consider a directory that contains file1 and file 2. A digest of type 2 shall be computed for file1 and a digest of type 3 shall be computed for file2:

The digest of file1 is simply SHA-1(contents of file1).

The digest of file2 is SHA-1 ((uimsbf 32) 1 + (uimsbf 32) FileLength(file2) + contents of file2).

15.13.7.1.4 Special authentication rules

The following rules shall also be observed:

- Each hash file shall either provide a complete list of all the files named in the directory that are to be accessible, except itself and the signature file, mentioning each name exactly one time, or shall list no files.
- Where the hash file contains a set of names, the set of names shall act as a filter for the set of objects that can be accessed. In this case, attempts to access a file that is not named in the hash file and is not the hash file itself shall fail as if a hash could not be verified.

These rules shall apply regardless of the value of digest_type associated with the object.

- If the digest_count is equal to 0, every entry in the directory shall be non-authenticated subject to the encoded pathname elements matching those of the directory (see also clause 15.13.7.1.1).
- If any name_length is equal to zero the associated digest shall be ignored.
- Where the hash file is present in a DSM-CC carousel, Stream or StreamEvent objects shall specify a digest_type of zero.
- Subdirectories may be listed in the hash file but shall be ignored if present.
- Any item that cannot be successfully parsed due to an unsupported digest_type, or any unexpected, corrupt or missing data shall be ignored and no subsequent entries in the hash file shall be considered.

15.13.7.2 Signature file

15.13.7.2.1 Description

The signature file is a file containing one digital signature. It contains the following ASN.1 [7] DER structure:

```
Signature ::= SEQUENCE {
  certificateIdentifier      AuthorityKeyIdentifier,
  hashSignatureAlgorithm    OBJECT IDENTIFIER,
  signatureValue            BIT STRING }
```

certificateIdentifier: The certificateIdentifier field shall be as defined in the Recommendation ITU-T X.509 [35] extension for the AuthorityKeyIdentifier field. It identifies the certificate that carries the certified public key that is used to check the signature.

```
AuthorityKeyIdentifier ::= SEQUENCE {
  keyIdentifier              [0] KeyIdentifier OPTIONAL,
  authorityCertIssuer        [1] GeneralNames OPTIONAL,
  authorityCertSerialNumber  [2] CertificateSerialNumber OPTIONAL }
```

Implementations need not use the optional `keyIdentifier` element of the `AuthorityKeyIdentifier`. The `AuthorityKeyIdentifier` structure shall contain both the `authorityCertIssuer` and `authorityCertSerialNumber` elements.

The `authorityCertIssuer` shall contain the field "directoryName"; this field shall be equal to the `issuerName` of the certificate that carries the public key used to check the signature.

The `authorityCertSerialNumber` shall be equal to the `serialNumber` of the certificate that carries the public key used to check the signature.

hashSignatureAlgorithm: The `hashSignatureAlgorithm` field identifies the hash algorithm that is used. Note that the encryption algorithm used to compute the signature itself is already described in the `SubjectKeyInfo` field of the certificate that certifies this key, and thus only the identification of the hash algorithm shall be used here. The supported algorithm shall be SHA-1.

```
sha-1 OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) oiw(14)
    secsig(3) algorithm(2) 26 }
```

signatureValue: The RSA signature generation process and the encoding of the result are described in detail in IETF RFC 3447 [53].

15.13.7.2.2 Signature file location and naming conventions

The signature file is located in the directory whose hash file it signs. There can be several signature files, as there can be several entities that sign the structure. See clause 15.13.6.

The name of a signature file shall be:

```
auth.sig.<x>
```

where the `<x>` is a textual representation of an integer decimal number without leading zeroes. The range of values represented in any single directory shall start with 1 and increment in steps of 1. The first unused integer value in the ascending sequence indicates the end of the range.

The purpose of this `x` is to allow the hash file of an authenticated directory to be signed by more than one entity. It is independent of the `x` value in the file name of the certificate files; receivers shall use the certificate issuer and serial number values to identify the corresponding certificate. See clauses 15.13.7.3.5 and 15.13.6.

15.13.7.2.3 Supported algorithms

Signing the hash file is a two-step process:

- 1) first a hash is computed over the contents of the hash file;
- 2) the resulting hash value is then encrypted using an encryption algorithm.

The hash algorithm shall be SHA-1.

The encryption algorithm used to compute the signature is indicated in the certificate that carries this key. The encryption algorithm shall be RSA.

15.13.7.3 Certificate file

15.13.7.3.1 Description

The certificate file contains all of the certificates in a certificate chain up to, and including, the root certificate. The leaf certificate is placed first in the file. The last certificate in the file is the root certificate. The encoding of the certificate is defined in Recommendation ITU-T X.509 [35]. The profile of Recommendation ITU-T X.509 [35] for use in authenticating applications is specified by clause 12.5 of ETSI TS 101 812 [36]. All references to 'MHP terminals' or 'decoders' shall be treated as references to a receiver conforming to the present document. Further, the rules for encoding the `organizationName` in clause 12.5.6 of ETSI TS 101 812 [36] shall be ignored.

The syntax of the certificate file is shown in table 15.39.

Table 15.39: Certificate syntax

Syntax	Bits	Type
<pre> Certificatefile () { certificate_count for(i=0; i<certificate_count; i++) { certificate_length certificate() } } </pre>	16	uimsbf
	24	uimsbf

certificate_count: This 16-bit integer carries the number of certificates in the certificate file. Receivers shall support counts up to and including 5.

certificate_length: This 24-bit integer specifies the number of bytes in the certificate. This length value shall be no more than 65 535.

certificate(): This field carries a single 'Certificate' data structure as defined by Recommendation ITU-T X.509 [35]. See IETF RFC 2459 [33] and clause 12.5 of ETSI TS 101 812 [36].

15.13.7.3.2 ASN.1 encoding

The basic specification of the ASN.1 DER encoding used in IETF RFC 2459 [33] is given in the ASN.1 specifications, Recommendation ITU-Ts X.680 [6] and X.690 [7]. However, IETF RFC 2459 [33] defines some extensions that are required to implement the present document.

15.13.7.3.3 Supported algorithms

There are various algorithm identifiers in the certificate structure. The Object Identifier (Recommendation ITU-T X.509 [35]) of the algorithm used in the SubjectPublicKeyInfo structure shall be RSA.

The values for AlgorithmIdentifier used both in the certificate structure and in the TBSCertificate structure that are supported in the present document are listed in clause 12.5.1 of ETSI TS 101 812 [36].

Receivers shall support certificate key lengths up to and including 2 048 bits.

15.13.7.3.4 Name matching

The only allowed encoding of attributes of distinguished names shall be UTF8String.

NOTE: The use of this encoding allows name matching to be a binary comparison.

15.13.7.3.5 Certificate file location and naming conventions

A certificate chain is a hierarchy of certificates that enable the implementation to verify the validity of the key used to check a signature. For consistency, the file shall carry all of the certificate chain up to, and including, the root certificate.

The certificate file that leads to the public key of a signature shall be placed in the root directory of the "Authentication Source" (see clause 8.1.5.4).

The name of a certificate file shall be:

auth.cert.<x>

where the <x> is a textual representation of an integer decimal number without leading zeroes. The range of values represented in any single directory shall start with 1 and increment in steps of 1. The first unused integer value in the ascending sequence indicates the end of the range.

The purpose of this x is to allow multiple signatories to sign files. When trying to find a certificate to verify a signature, the receiver shall try all certificates in turn until one is found that matches the certificate issuer and serial number values in the signature. See clause 15.13.6.

Certificates shall be considered to be the same if they have bitwise identical contents.

15.13.7.3.6 Authentication rules

Where multiple signature files authenticate an application, the receiver shall check each one before deciding that a file fails authentication.

15.13.8 Example of creating an application that can be authenticated

15.13.8.1 Scenario (informative)

This clause is informative and gives an example of how a file system carrying an application can be organized.

In this example, the file system carries single signed application app1.

The application is comprised of the files app1/a and app1/scene.mhg, which are both signed. The file app1/data2/foo is also signed. The file app1/logo.png and the contents of the directory app1/data1 are marked as non-authenticated.

The file structure is shown in figure 15.5.

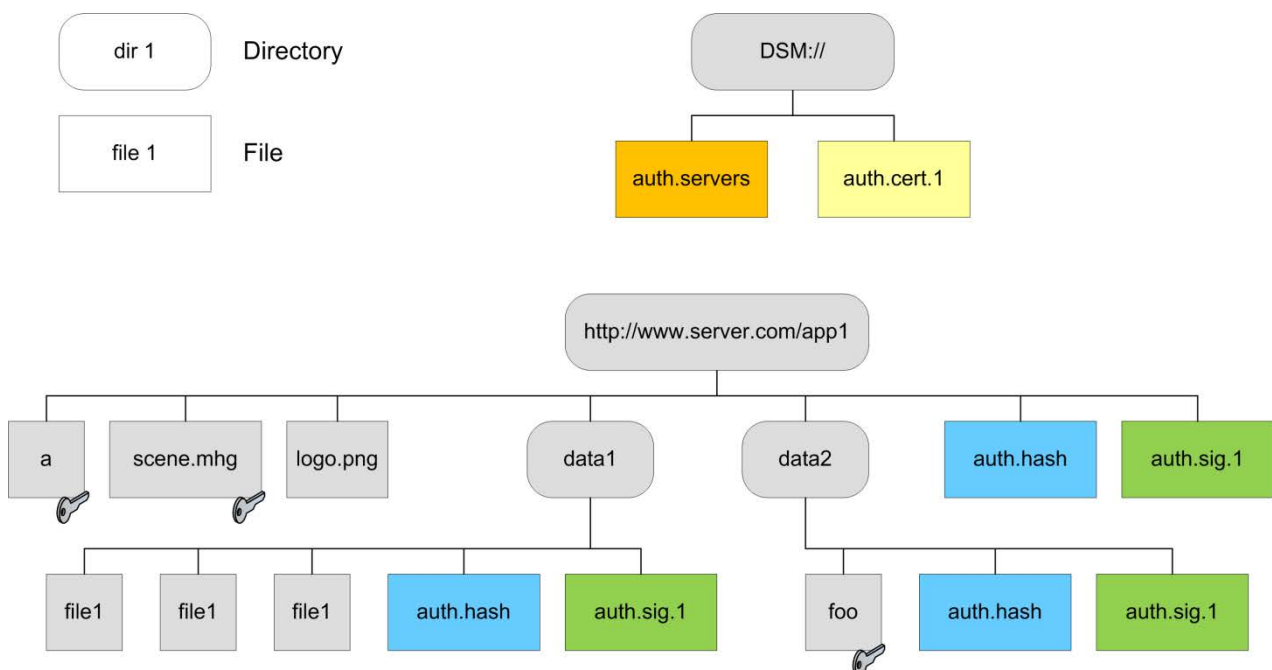


Figure 15.5: Example application file structure

15.13.8.2 Hash and signature computations

15.13.8.2.1 Computation of the hashes of the app1 directory

- 1) initialize the SHA-1 algorithm;
- 2) apply the SHA-1 algorithm to the contents of the file a to compute H1 then apply the SHA-1 algorithm to the contents of the file scene.mhg to compute H2;
- 3) construct the contents of the hash file for the app1 directory as shown in table 15.40;

Table 15.40: Hash file for app1 directory

Field	Comment
3	Three digests
2	Type of digest algorithm = SHA-1
1	Entries over which an SHA-1 hash has been computed
a	Name of entry
H1	SHA-1 hash of file a
2	Type of digest algorithm = SHA-1
1	Entries over which an SHA-1 hash has been computed
scene.mhg	Name of entry
H2	SHA-1 hash of file scene.mhg
0	Type of digest algorithm = non-authenticated data
1	Number of entries
logo.png	Name of entry
<no digest in this case>	

- 4) create the hash file app1/auth.hash with the above contents.

15.13.8.2.2 Computation of the hashes of the app1/data1 directory

- 1) construct the contents of the hash file for the directory app1/data1 as shown in table 15.41;

Table 15.41: Hash file for app1/data1 directory

Field	Comment
0	No digests
app1/data1	Pathname

- 2) create the hash file app1/data1/auth.hash with the above contents.

15.13.8.2.3 Computation of the hashes of the app1/data2 directory

- 1) initialize the SHA-1 algorithm and compute the SHA-1 hash H3 using the contents of the file app1/data2/foo;
 2) construct the contents of the hash file for the directory app1/data2 as shown in table 15.42;

Table 15.42: Hash file for app1/data2 directory

Field	Comment
1	One digest
2	Type of digest algorithm = SHA-1
1	Entries over which an SHA-1 hash has been computed
foo	Name of entry
H3	SHA-1 hash of file foo

- 3) create the hash file app1/data2/auth.hash with the above contents.

15.13.8.2.4 Computation of the signatures

- 1) initialize the SHA-1 algorithm and compute the SHA-1 hash H4 using the contents of the file app1/auth.hash;
 2) ASN.1 encode the following structure (DigestInfo):
- DigestAlgorithm: SHA-1
 - Digest: H4
- 3) RSA-encrypt the result of step (2) with the private key corresponding to the public key that can be found in the leaf certificate in the file DSM://auth.cert.1. In this example this has serial number 0123456;

- 4) ASN.1 encode the following structure:
 - AuthorityCertIssuerName: Name of the CA
 - AuthorityCertSerialNumber: 0123456
 - HashSignatureAlgorithm: SHA-1
 - SignatureValue: result of step (3)

- 5) put this structure into the signature file app1/auth.sig.1.

This procedure is repeated for the app1/data1 and app1/data2 directories to create the files app1/data1/auth.sig.1 and app1/data2/auth.sig.1.

15.14 Controlling access to Internet servers

15.14.1 Overview

This clause describes a mechanism to restrict the set of Internet servers that an application can access under normal operating conditions. The mechanism limits the extent to which applications could, accidentally or otherwise, send sensitive information to unauthorized servers.

The present document provides no explicit signalling to control access to Internet servers beyond removal of the broadcast carousel or server list file.

15.14.2 Restricting access to Internet servers

15.14.2.0 Overview

Access to the Internet shall be restricted to specified directories on servers that are listed in a 'server list' file. If no such file is present, the receiver shall not establish HTTP or TLS connections to any Internet server.

Before connecting to any Internet server following a request from an application, receivers shall verify that the connection being made is permitted by the server list file.

If the connection is not permitted, the operation shall fail as if the connection could not be established. A ContentRefError, GroupIDRefError, or StreamRefError event shall be generated (as appropriate). The engine events ICLocalError, ICNetworkError and ICRemoteError shall not be generated.

Receivers shall also verify that the connection is permitted when making a POST request, although no authentication is required for the data returned in the response (clause 11.10.12.2).

Where a file request results in redirection the server list file shall not be used to verify subsequent connection.

The behaviour for caching the server list file shall be the same as that for the application signing certificate (see clause 15.13.6.4).

15.14.2.1 Server list file

15.14.2.1.1 Location

The name of the server list file shall be:

auth.servers

Receivers shall look for this file in the root directory of the "Authentication Source" (see clause 8.1.5.4).

The server list file shall specify the hosts for which interaction channel access is permitted and whether interaction channel authentication is required for content from each host.

15.14.2.2 File format

The server list file shall contain zero or more entries, each separated by a newline (0x0A) character. Empty lines and lines beginning with '#' (0x23) shall be ignored. The format of each entry is:

```
[!]source://servername[:portnumber][/path_elements][/]
```

where:

- **source** shall be http or https;
- **servername** shall be the host name or IPv4 address (dotted decimal) of a server for which interaction channel access is to be permitted;
- **portnumber** shall be the optional IP port number (decimal) to which access is to be permitted;
- the optional **path_elements** field shall consist of one or more path elements such that access is restricted to those directories on servername whose path begins with path_elements;
- the **source** field may be optionally prepended with an "!". If the source field is prepended with an "!", then content (but not MHEG code) shall be loaded from matching hosts by ignoring the "!" and loading content as if the authentication level were "1" (i.e. authentication is not required) for any subsequent redirection. If the source field is not prepended by an "!" or the source field is prepended with a "!" but the content is MHEG code, then authentication shall take place according to the authentication level as described in clause 15.13.2.

Receivers shall determine whether URLs match by performing a binary comparison between the first parts of the string requested by the application and the whole entry in the server list file appended with a directory separator ("/) if the entry does not end with one.

15.14.2.3 Example

An entry for http://www.server.com/mheg shall implicitly be http://www.server.com/mheg/ and shall allow interaction channel access for the /mheg directory on the HTTP server with domain name www.server.com. It shall not permit access to any of the following:

- a file named /mheg;
- a directory named /mhegstuff;
- a TLS connection to a server called www.server.com;
- An entry for !http://www.server.com allows interaction channel access to content on the HTTP server with domain name www.server.com without requiring a valid hash file to be present. It would not allow access to MHEG code without a valid hash file.

15.15 Transport Layer Security

15.15.1 Overview

Where applications need to send or receive data over the Internet securely, secure connections can be established using the TLS (Transport Layer Security) protocol as described in IETF RFC 2246 [37]. Certificates for authentication of TLS servers shall be provided in the broadcast carousel. TLS connections shall be requested by applications by using the "https:" URI scheme in place of "http:". See clause 16.3.2.3.4.

15.15.2 TLS features that need not be implemented

Receivers need not implement the following features of TLS:

- the functionality of being a server for the TLS protocol;
- compliance with SSL 3.0;
- TLS client authentication.

15.15.3 TLS cipher suites

The minimum set of TLS cipher tools that shall be implemented are:

- RSA
- MD5
- SHA-1
- DES
- AES

Table 15.43 identifies which methods shall be implemented in a receiver (see IETF RFC 2246 [37] and IETF RFC 3268 [38] for definition of the terms).

Table 15.43: TLS cipher suite methods

CipherSuite	Key Exchange	Cipher	Hash	Value (hex)	Status
TLS_NULL_WITH_NULL_NULL (see note)	NULL	NULL	NULL	00, 00	Mandatory
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5	00, 01	Mandatory
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1	00, 02	Mandatory
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA_EXPORT	DES40_CBC	SHA-1	00, 08	Mandatory
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA-1	00, 09	Mandatory
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA-1	00, 0A	Mandatory
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	AES_128_CBC	SHA-1	00, 2F	Mandatory
TLS_RSA_WITH_AES_256_CBC_SHA	RSA	AES_256_CBC	SHA-1	00, 35	Mandatory
NOTE: This cipher suite is only used by a TLS implementation during the negotiation of a connection. It is not required to be enabled as a cipher suite that is available for a negotiated connection.					

Note data encryption can be very time consuming for a server. The ciphers TLS_RSA_WITH_NULL_MD5 and TLS_RSA_WITH_NULL_SHA provide integrity checking but without confidentiality (i.e. data are in the clear). They are useful for applications in which data does not need to be encrypted but in which data integrity is very important. For these applications, the server will only have to compute an HMAC for every message exchanged.

15.15.4 Server authentication for TLS

15.15.4.0 Overview

Before the TLS connection can be established, the receiver shall ensure that the certificate list sent by a server contains at least one trusted certificate. In the computer environment, this is simply done by checking the list of certificates against a root certificate that is resident in the computer.

In the MHEG environment, an application author knows which servers an application will connect to and can provide a copy of the appropriate certificate with the application. To allow the receiver to check the certificate chain, at least one certificate from it shall be placed in the root directory of the DSM-CC carousel. The receiver shall check the signatures on all certificates in the chain from the leaf certificate to a certificate that is present in the carousel.

When the certificate chain sent by the TLS server cannot be verified against any of the TLS certificates present in the DSM-CC carousel, the receiver shall not use the connection to the TLS server.

Receivers shall ignore any 'CRL distribution point' fields during TLS server authentication.

15.15.4.1 Certificate files

15.15.4.1.1 Location

The TLS certificates shall be located in the root directory of the "Authentication Source" (see clause 8.1.5.4).

The name of a certificate file shall be:

auth.tls.<x>

where the <x> is a textual representation of an integer decimal number without leading zeroes. The range of values represented shall start with 1 and increment in steps of 1. The first unused integer value in the ascending sequence indicates the end of the range.

Receivers shall check each certificate file in turn until a certificate is found that matches a certificate from the certificate chain presented by the server.

15.15.4.1.2 File format

Each TLS certificate file shall contain a single certificate. The file format is otherwise the same as that used by certificate files for application authentication, see clause 15.13.7.3.

15.15.4.1.3 Certificate caching

Receivers may cache certificates for up to 24 hours to avoid loading them from the carousel each time a TLS session is established.

Where a signature to be checked references a certificate that is not available from cache, receivers shall check for the certificate in the carousel even if previous attempts to load the certificate have failed.

Receivers shall discard all cached certificates each time the auto-boot process begins (see clause 9.3.4.2).

15.16 HTTP profile for delivering encrypted streams

15.16.0 Scope

This clause describes additional behaviour in the HTTP profile to support the delivery of encrypted streams over HTTP. This behaviour is required for receivers that implement ICEncryptedStreamExtension.

15.16.1 Additional response headers

15.16.1.0 Usage of additional response headers

Receivers shall handle the following response headers in the manner described in this clause:

- X-Keys
- X-KeyLocation

If these headers are present in a Redirection response (3xx) they shall be applied to any media ultimately obtained from the destination of the redirect. If a header is received more than once, then the last value obtained shall be acted upon.

If the media is encrypted then the specified decryption keys shall be used to decrypt the media as described in clause 15.16.3.

15.16.1.1 X-Keys

When the X-Keys header is present it indicates that the transport stream is encrypted. This header contains the keys needed to decrypt the transport stream. Clause 15.16.3 describes how these keys should be used.

15.16.1.2 X-KeyLocation

When the X-KeyLocation header is present it indicates that the transport stream is encrypted. The value of this header is the URL from which the keys needed to decrypt the transport stream can be obtained. Clause 15.16.3 describes how that file is used. The scheme of the URL shall be either http or https as defined in IETF RFC 1738 [27].

15.16.2 Redirects

When a receiver encounters a redirect response when accessing a media URL it shall follow the redirect. Any of the additional headers, specified in clause 15.16.1, which it encounters in a redirect response should be obeyed, unless they are overridden by another header obtained by a later response.

Although media shall not be streamed over a TLS connection, it is permitted for an https URL to be used initially, allowing decryption keys to be returned together with a redirect to an http URL.

When a receiver follows a 301 (Moved Permanently) redirect it shall not re-request the same URL which returned the redirect unless the MHEG application resets the Content attribute of a Stream object to that location.

15.16.3 Encrypted content

15.16.3.0 Introduction

The presence of an X-Keys or an X-KeyLocation header in a response indicates that encryption has been applied to the payload of transport stream packets. This header may be present in a redirect response (see clause 15.16.2). The receiver shall load the specified keys into the transport stream decryption unit before commencing playback. Receivers shall support loading keys from either header. If both headers are present receivers may choose to use either one. If a receiver is unable to retrieve the file indicated by the X-KeyLocation header, or the file cannot be parsed, then the KeyFileError engine event shall be raised.

When the X-KeyLocation header is present the server list file shall not be used to verify any subsequent connection required to retrieve the key file. The content of the X-KeyLocation file is not required to be authenticated.

Receivers shall not implement any mechanisms designed to expose the keys to the user, to MHEG applications or to other devices.

15.16.3.1 Encryption scheme

Where content is encrypted it shall be done so using Transport Stream level AES encryption, as described here. This is compatible with the specification in section 6.4 of ISO/IEC 62455:2011 [52]. The encryption algorithm shall be AES-128 in CBC mode, described in FIPS PUB 197:2001 [50] with the termination mode as described in section 4.2 and 4.3 of ANSI/SCTE 52 2003 [51] (note that the DES encryption referred to in that standard shall not be used). In the present document, all encrypted PIDs shall use the same odd key, even key and initialization vector. The same initialization vector shall be used for all packets, regardless of whether the odd or even key was used.

The decryptor is re-initialized with the specified IV before the decryption of every TS packet.

The transport_scrambling_control bits in the transport packet header of each packet indicate whether the packet has been encrypted with the odd or even key, or is in the clear as follows:

Table 15.44

transport_scrambling_control value	Meaning
00	Not encrypted
01	Not used in this profile.
10	Encrypted with Even Key
11	Encrypted with Odd Key

15.16.3.2 Format of key file or X-Keys header

For each PID to which encryption has been applied the following information is contained within the key file or header:

- PID
- IV
- Odd Key
- Even Key

The PID is encoded as a decimal number in ASCII. All other values are binary, encoded as Base64. The contents of the key file and the header have exactly the same format.

```

KeyFile ::= KeyString
KeyHeader ::= 'X-Keys:' <space> KeyString
KeyString ::= PIDString [ ':' KeyString ]
PIDString ::= PID ':' IV ':' OddKey ':' EvenKey
PID ::= <integer encoded as decimal ascii value>
IV ::= <128bit binary value encoded as Base64>
OddKey ::= <128bit binary value encoded as Base64>
EvenKey ::= <128bit binary value encoded as Base64>

```

NOTE: This format allows the key and IV data to be specified separately for each PID. However, in this profile, receiver behaviour is undefined if different PIDs have different data specified.

16 Name mapping

16.1 Names within the receiver

See table 16.1.

Table 16.1: 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 14.2.6.
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 14.2.6.
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/eu1	Font attribute of Application class or OriginalFont attribute of Text class	This identifies the in-built receiver font described in clause 13.3.2 rendered with the default metrics.
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 rendered using the font metrics of v7.51 of the font.
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 13.8.1.
rec://htext/bot		
pst://<name>	Name space for true persistent storage	See clause 14.7.

16.2 MHEG-5 component tags

See clause 15.3.

16.3 Namespace mapping

16.3.0 Introduction

For an application to start, at least one of the file systems available to the receiver shall have been mounted. The file systems are listed in clause 8.1.6. Data may be retrieved from one or more unambiguous namespaces.

The high-level API differentiates between three types of retrieved data:

- objects that comply with the high-level API definition, i.e. MHEG-5;
- the content (such as bitmaps or text) of those objects; and
- streams (such as video and audio).

For accessing the application data on the server side, the DSM-CC Directory, File and Stream objects shall be used. Note that the server, in this context, need not be a physical server, but could be implemented, for example, as a broadcast carousel in a pure-broadcast topology.

Each file accessed by an MHEG engine is either a Scene object, an Application object, or the content data of an Ingredient object. Each Scene object, Application object and content data shall be stored in a separate file.

16.3.1 MHEG-5 object references

16.3.1.0 Mapping rules

MHEG-5 objects can be exchanged in two ways. Application and Scene objects shall be exchanged as files, which may be retrieved from any mounted file system. All other objects are classed as **Ingredients** and shall be exchanged within another high-level API object, i.e. within an **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 files, the following additional rules are defined:

- 1) For any **GroupIdentifier**, the mapping rules defined in clause 16.3.2 apply. The **GroupIdentifier** shall be a string which, when expanded according to those rules, gives the pathname (URI) of the file.
- 2) Specifically, each **Application** and **Scene** object shall have in its **GroupIdentifier** a byte string which maps on the name of the file which contains that object. These objects shall have their **ObjectNumber** set to 0.
- 3) Each application shall have exactly one **Application** object. That object shall be contained in a file. Only one **Application** object shall be contained in each such file. See clause 9.3.4.2.
- 4) **Ingredient** objects may either:
 - a) leave out the **GroupIdentifier**, in which case it is assumed to be a string which maps on the name of a file which contains the object (**Application** or **Scene**) of which this object is a part; or
 - b) fill in the **GroupIdentifier** with such a string.
- 5) **Ingredient** objects shall have their **ObjectNumber** set to a value which is unique within that **Group**.

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** should always 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 shall always refer to a DSM-CC File object.

16.3.2 Mapping rules for GroupIdentifier and ContentReference

16.3.2.1 Void

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.

Additionally in receivers that implement **InteractionChannelExtension** http and https URIs (see clause 15.6), and the hybrid file system (see clause 15.12), both provide case sensitive file names.

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 17.8 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

16.3.2.3.0 Syntax

GroupIdentifiers and ContentReferences are composed of the following four components in sequence:

- source;
- path origin;
- path;
- filename.

Table 16.2 specifies the allowed contents of each of these components and their meaning.

Table 16.2: Definition of reference components

Source	Path origin	Path (see note)	Filename	Meaning
"DSM:"				The service gateway of the "Current carousel". See clause 8.1.5.2.
"~"				For receivers not implementing InteractionChannelExtension this is shorthand for "DSM:". For receivers implementing InteractionChannelExtension this is shorthand for "hybrid:".
"CI:"				The root of the file system provided by a CI module while application MMI session is open after a RequestStart (see clause 14.10.3.2) has been sent by the module. See clause 14.10.
empty string ""				Shorthand for "Current carousel" (see clause 8.1.5.2). Resolves to a source 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. The text "dir" is one or more characters long. 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. See clause 16.3.2.1 for limitations. 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.

Additional components for receivers implementing InteractionChannelExtension are shown in table 16.3.

Table 16.3: Additional components for receivers implementing InteractionChannelExtension

Source	Path origin	Path (see note)	Filename	Meaning
"http:"				Denotes that the file reference is an http URI, to be retrieved by the IC file system, (see clause 15.6) or, delivered via the Interaction Channel in the case of a stream reference.
"https:"				Denotes that the file reference is an https URI, to be retrieved by the IC file system, (see clause 15.6) or delivered via the Interaction Channel in the case of a stream reference.
"hybrid:"				The root of the hybrid file space, (see clause 15.12).

16.3.2.3.1 Resolution of file references

The process for resolving file references shall be equivalent to:

- 1) Expand any shorthand components:
 - an empty source shall expand to the string represented by the "Current Source" (see clause 8.1.5);
 - the path origin "/" shall expand to a path such as "//weather/today/";
 - the source and path origin together may be "DSM://weather/today/".

Therefore:

"/cloud" may become "DSM://weather/today/cloud".

- 2) Allow each "../" component to consume the path component preceding it.

Therefore:

"/../tomorrow/app" may 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 shall operate left to right;
- there may be many "../" components in a path and these need not be contiguous;
- the reference shall be 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.

"../" shall be resolved before the indirection through any file system. Additionally the file path shall never contain "../" when presented to the file system.

16.3.2.3.3 URI Syntax

A file reference with a source of "http:" or "https:" shall conform to the syntax for an http or https URI, even if this does not match the four components described above. In particular, the part of the URI corresponding to Path Origin shall be "/" and not "".

16.3.2.3.4 Non-equivalence of URIs with different sources

No assumption can be made that URIs that are equivalent in their path but different in the source, reference the same resource. In particular "http" and "https" URI schemes define separate namespaces (http://example.org/ and https://example.org/ are not, or may not be, equivalent).

16.3.2.3.5 Maximum reference length

The following limitations apply to resolved file references:

- DSM-CC reference shall be at most 64 bytes.
- Interaction Channel references shall be at most 1 024 bytes.
- CI references shall be at most 64 bytes.
- Hybrid File System references have no restrictions but references to the underlying file system will have limitations.

This limitation applies to the length of the reference after complete resolution (i.e. the final "DSM://weather/tomorrow/app" form) and after any hybrid mapping has been applied.

Implementers should be aware that this limitation does not apply to a partially resolved reference (e.g. "DSM://weather/today/./tomorrow/app"). These may require significantly more memory.

16.3.2.3.6 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.

Additionally, for receivers that implement InteractionChannelExtension the following clarifications shall apply:

- "http:" is 0x687474703A.
- "https:" is 0x68747470733A.
- "hybrid:" is 0x6879627269643A.

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 shall not be permitted as it cannot be distinguished from the meaning of "/" defined above.

16.3.2.5 Reserved characters in HTTP URIs

For ContentReferences or GroupIdentifiers that contain an HTTP URI (or for references in the hybrid file system that resolve to an HTTP URI), receivers shall not perform percent-encoding; it is the responsibility of application authors to ensure that URIs are correctly encoded. This shall also apply to the URI to which data is sent using the ReturnData resident program (see clause 11.10.12.2).

16.3.3 URL formats for access to broadcast services

16.3.3.1 Numerical format

DVB TS 101 812 [36] 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 shall not be supported.

The values of <original_network_id>, <transport_stream_id> and <service_id> shall be represented as hexadecimal strings without any "0x" prefix (for example, "4d2e").

See ETSI TR 101 211 [i.3]. Services are unique within the scope of a single original network ID. So, transport stream ID need not 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.

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 cannot 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 should be used unless otherwise stated in a localized profile of the present document:

```
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 clause 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.

17.2 Avoiding confusion with navigator functions

17.2.1 Use of user inputs (mandatory)

17.2.1.1 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.

17.2.1.2 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`.

17.2.1.3 Leaving

An exception to the above is that through use of "Persistent storage" (see clause 14.6) 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.

Applications shall provide a means of returning to the "selected" state. Such means should be clear and simple for the viewer to use.

17.2.2 Broadcast-triggered native applications

Receivers supporting `NativeApplicationExtension` may give control of one or more keys to a broadcast-triggered native application instead of an MHEG 5 application, or may allow such a native application to overlay an MHEG-5 application. MHEG 5 application authors should be aware that a broadcast-triggered native application may be signalled by SI on the current service, or may be present on the new service following a service tune.

The effect on a running MHEG 5 application can be controlled through use of the `SetBroadcasterInterruptions` resident program. See clause 11.10.10.5.

Following a service tune the effect can be controlled by the `broadcaster_interrupt_flag`. See clause 11.10.8.4.

17.3 Use of the "Text" and "Cancel" functions

17.3.0 Historical meaning of specific input 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 "interacting" 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).

17.3.1 The traditional "teletext" key

17.3.1.0 Use of the "Text" key

In figure 17.1 "Text" toggles between conventional TV and a full-screen MHEG-5 application. This is very similar to the behaviour of the analogue TV's "teletext" button.

An available elaboration is that a single "teletext" service can be shared by a number of TV services.

17.3.1.1 Entering

In this case the first `Scene` of the "Auto boot broadcast application" (see clause 8.1.3) 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.

17.3.1.2 Leaving

If the application is part of the service and the "Text" user input event is activated a second time, then the "return" may 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 17.1.

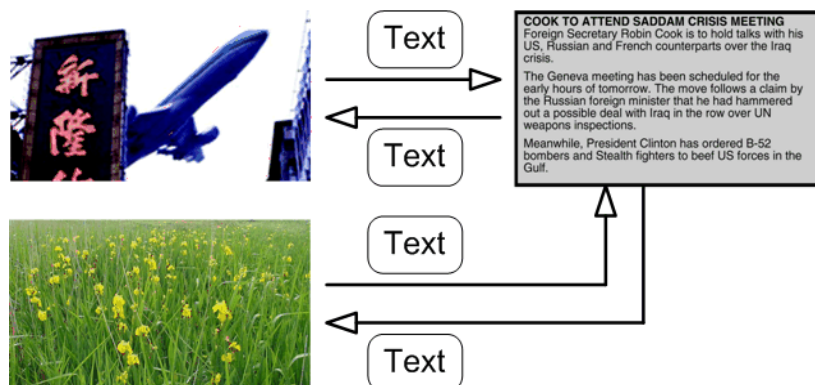


Figure 17.1: "Text" accesses "teletext"

17.3.2 Accessing additional programme information

In figure 17.2 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).

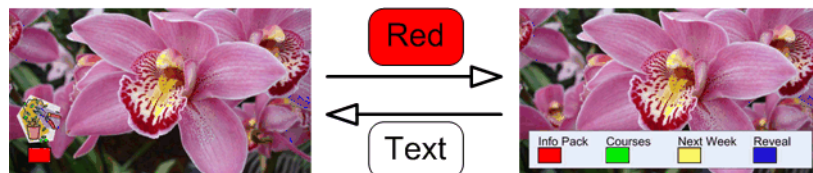


Figure 17.2: "Text" provides further information

Exactly how such visual prompts are used is the broadcaster's choice. For example, this graphic might be visible throughout the programme. Alternatively, it may be shown for short periods e.g. after the programme is first selected and at the end of each "item" within the programme.

17.3.3 "Text" has no effect

In figure 17.3 the TV service has no MHEG-5 application. In this case "Text" should not have any effect. However, the receiver may optionally provide a response.

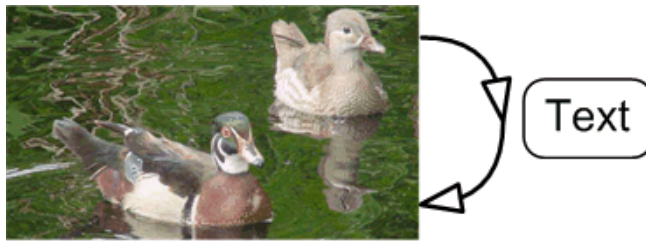


Figure 17.3: TV with no MHEG-5 application

In figure 17.4 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.

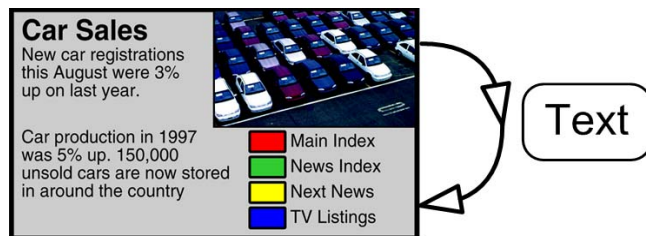


Figure 17.4: MHEG-5 service has no TV alternate

In the case shown in figure 17.4 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.

17.3.4 On-screen prompts

The "CancelKeyFunction" user input event should be 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.

17.4 Use of stream decoders

17.4.1 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.

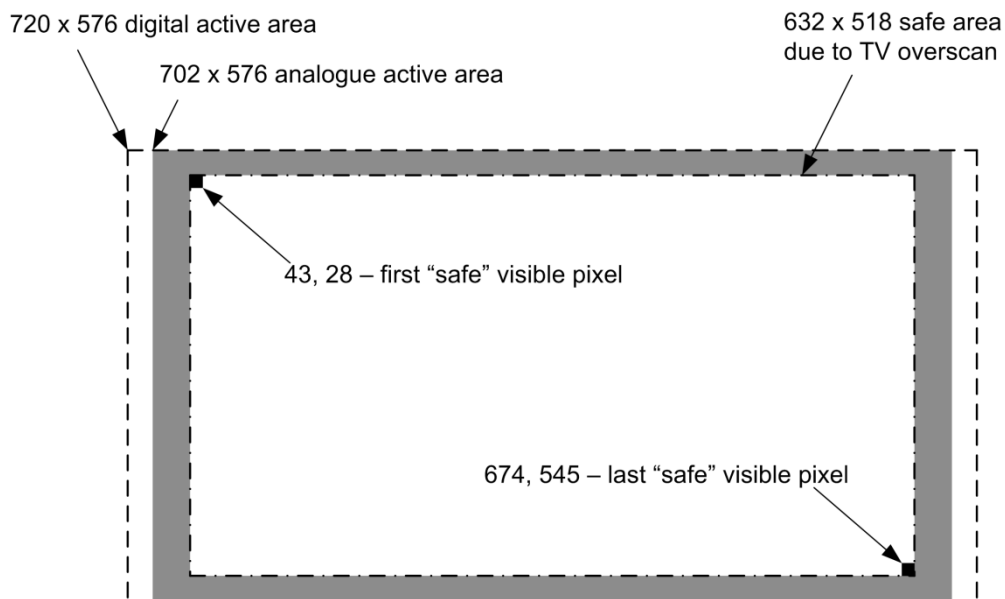
To obtain deterministic behaviour, authors shall not build applications that attempt to activate more than one of each type of decoder, e.g. applications shall Stop a running Stream object using MPEG video before Running a Bitmap object with MPEG I-frame content (content hook 2 or 7).

See clause 14.8.2.

17.4.2 Visible area

Typically a 5 % border region can be lost due to monitor overscan. This leaves a central 632 x 518 area of the graphics plane which should 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 17.5.



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.

Figure 17.5: Authoring "safe" area

17.4.3 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 shall 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 can support simultaneous subtitles and MHEG-5 presentation.

The author may choose to suppress the subtitles for compositional reasons. Calling the resident program `SetSubtitleMode(false)` shall ensure that subtitles are not visible even on receivers capable of simultaneous presentation. `SetSubtitleMode(true)` shall re-enable presentation if it has been requested by the viewer (see clause 14.3.3).

17.4.4 Accuracy of video positioning

Authors should note that some platforms have limited video positioning accuracy. The degree of error may be affected by the position and by the scaling factor employed but may be up to 2 pixels horizontally or vertically in the source video or I-frame.

If greater accuracy than this is required then authors should consider using the `VideoToGraphics ResidentProgram`. The return values from this should be accurate to within 2 destination graphics pixels.

17.4.5 Defensive behaviour (mandatory)

Applications shall take steps to prevent inappropriate behaviour if the receiver cannot provide stream components when requested, for example if the service in question is badly specified or not known to the receiver. This may be achieved by checking service references using the `SI_GetServiceIndex` resident program or by appropriate handling of `StreamPlaying` and `StreamStopped` events. Suitable behaviour could be the display of an appropriate apology message.

17.5 Aspect ratio

17.5.1 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 17.2.1), This means that such an application shall begin with a Scene whose AspectRatio is undefined.

17.5.2 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 should 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.

17.5.3 I-frames

I-frames shall be combined with other MHEG-5 graphics without any decoder format conversion. 4:3 I-frames shall 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.

17.5.4 Presentation of 16:9 AspectRatio Scenes on 4:3 displays

Application authors should note that predictable representation of 16:9 Scenes on 4:3 displays is unlikely to be possible.

17.5.5 Presentation of 4:3 AspectRatio Scenes over HDMI

Application authors should note that predictable representation of 4:3 Scenes over HDMI is unlikely to be possible.

17.6 PNG bitmaps

17.6.1 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 can produce pictures of much lower quality when an interlace method (such as Adam7) is used.

17.7 Missed events

Applications cannot be guaranteed to receive all events. For example, stream events and timer events can be missed while an application is paused as a consequence of losing priority access to the display. See clause 14.9.

17.8 File naming

17.8.1 Name length (mandatory)

The length of file references shall be limited. See clause 11.11.5.

17.8.2 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 should be 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).

17.8.3 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.

17.8.4 File names in persistent storage (mandatory)

The <name> part of the file name "ram://<name>" (see table 16.1) 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.

Applications that make use of persistent storage should ensure that filenames do not clash by choosing a prefix that is specific to the application provider.

17.8.5 File names in true persistent storage (mandatory)

The <name> part of the file name "pst://<name>" used to access files in true persistent storage (see clause 14.7) shall be managed to avoid accidental file name collision between the applications of different service providers.

The first 3 characters of <name> shall be allocated to a broadcaster/application author by the DTG.

17.9 Text encoding

17.9.1 Mark-up

Mark-up coding for Text objects is described in clause 13.6 is functionally equivalent to a small subset of HTML recoded to improve transmission efficiency. A simple translation process between the two formats is possible.

See table 17.1.

Table 17.1: Text object mark-up codes

HTML mark-up	Broadcast mark-up codes
(no equivalent)	0x09 (see note 1)
(no equivalent)	0x20 (see note 2)
 	0xC2 0xA0
 	0x0D
<P>	
</P>	0x0D 0x0D
 [text in bold] 	0x1B 0x42 0x00 [text in bold] 0x1B 0x62
 [coloured text] 	0x1B 0x43 0x04 0xrr 0xgg 0xbb 0xtt [coloured text] 0x1B 0x63
<	0x3C
>	0x3E
& (see note 3)	0x26
 [anchor text] 	0x1B 0x41 0xnn tag_bytes [anchor text] 0x1B 0x61
<body	0x1B 0x44 0xnn body_attr_bytes 0x1B 0x64
bgcolor=colour	HyperText object BackgroundColour
text=colour	HyperText object TextColour
link=colour	body_attr_bytes anchor_colour (see table 13.20)
vlink=colour	body_attr_bytes visited_anchor_colour (see table 13.20)
alink=colour>	body_attr_bytes active_anchor_colour (see table 13.20)
NOTE 1: Tab characters have meaning, see clause 13.5.9.	
NOTE 2: All space characters are significant.	
NOTE 3: All HTML "named character entities" can be directly represented with a simple character code where they exist in the font used to render the text object.	

17.9.2 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 13.9.

17.9.3 Width of row of characters

Certain characters can 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 the present document. 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.

17.10 Reference checking

17.10.1 Application design issues when checking references

17.10.1.0 Introduction

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.

17.10.1.1 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.

17.10.1.2 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.

17.10.2 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 11.10.16, 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 (see clause 11.10.9.1)
//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
// expected 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 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
// this implements:
// if( fork returned OK &&
//     the correct reference has been checked &&
//     the file referenced is available )
// {
//     TransitionTo( the next Scene )
// }
{:Link 14
  :EventSource 3
  :EventType TestEvent
  :EventData true
  :LinkEffect {
    :TransitionTo(:IndirectRef 2)
  }
}

```

17.11 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 shall 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.

17.12 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.

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) cannot be guaranteed to be detected by the receiver for outstanding events. Modification may result in the receiver waiting indefinitely for an event that will never be broadcast.

17.13 User input events

17.13.1 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" (see clause 11.6.1.2) shall be generated only when there is an active Scene object. Authors might wish to create an application which does not transition to a Scene until a "Register 3 group" (see clause 11.6.1.2) key is pressed - this is not possible. Instead the application should transition to a simple transparent Scene and wait for an EngineEvent or UserInputEvent related to the required key.

17.13.2 Use of user input related engine events

An application may be driven from two classes of user input related asynchronous events:

- UserInputEvents can be generated by the current Scene when there is no active interactible;
- EngineEvents (TextKeyFunction, RedKeyFunction, etc.(see table 11.10)) can be generated by the current application. They shall be generated even if there is an active interactible (however, see clause 17.13.1).

When both are generated the EngineEvent shall be generated before the UserInputEvent.

17.14 Undefined behaviour (mandatory)

17.14.0 Avoid use of features with undefined behaviour

Some aspects of MHEG-5 engine behaviour are undefined both in the ISO/IEC 13522-5 [14] (MHEG-5) specification, and in the present document. Authors shall avoid reliance on any one implementation and the behaviour observed on that platform.

17.14.1 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 the ISO/IEC 13522-5 [14] (MHEG-5) specification and has led to receiver implementations varying in behaviour. Some receivers queue all of the events, and others 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.

17.14.2 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.

17.15 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 (ISO/IEC 13522-5 [14]) (MHEG-5) specification 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 shall be ignored.

17.16 Catching failure of TransitionTo

17.16.1 Background

It is good authoring practise to use LockScreen prior to a transition to another Scene (using TransitionTo).

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 shall continue to process actions in the current `Scene`. If the screen is still locked the user would perceive that the receiver has locked-up.

NOTE: Authors should write defensively to ensure that the application will continue to interact with the user if an expected transition fails.

Typically this defensive coding shall include placing one (or more) `UnlockScreen` elementary actions following the `TransitionTo` elementary action.

17.17 Elements of the object carousel

17.17.1 DIIs and elementary streams

The following recommendations are made to aid good receiver performance:

- Ensure that eight or fewer DIIs should be monitored to achieve any single presentation.
- Ensure that the DSM-CC sections that require monitoring for any single presentation should be distributed over four or fewer elementary streams.

17.17.2 Directory structure (mandatory)

- The maximum allowed number of bindings in each directory or service gateway message shall be 512.
- 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.

17.17.3 Timeouts (mandatory)

Timeouts are encoded for both DII messages and Modules (see clauses 15.2.2.5 and 15.2.5). These values have no defaults and shall be explicitly encoded when constructing the carousel. The encoded value shall 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.

17.17.4 Examples of object carousels

Figure 17.6 illustrates an object carousel that is distributed over three elementary streams belonging to the same service.

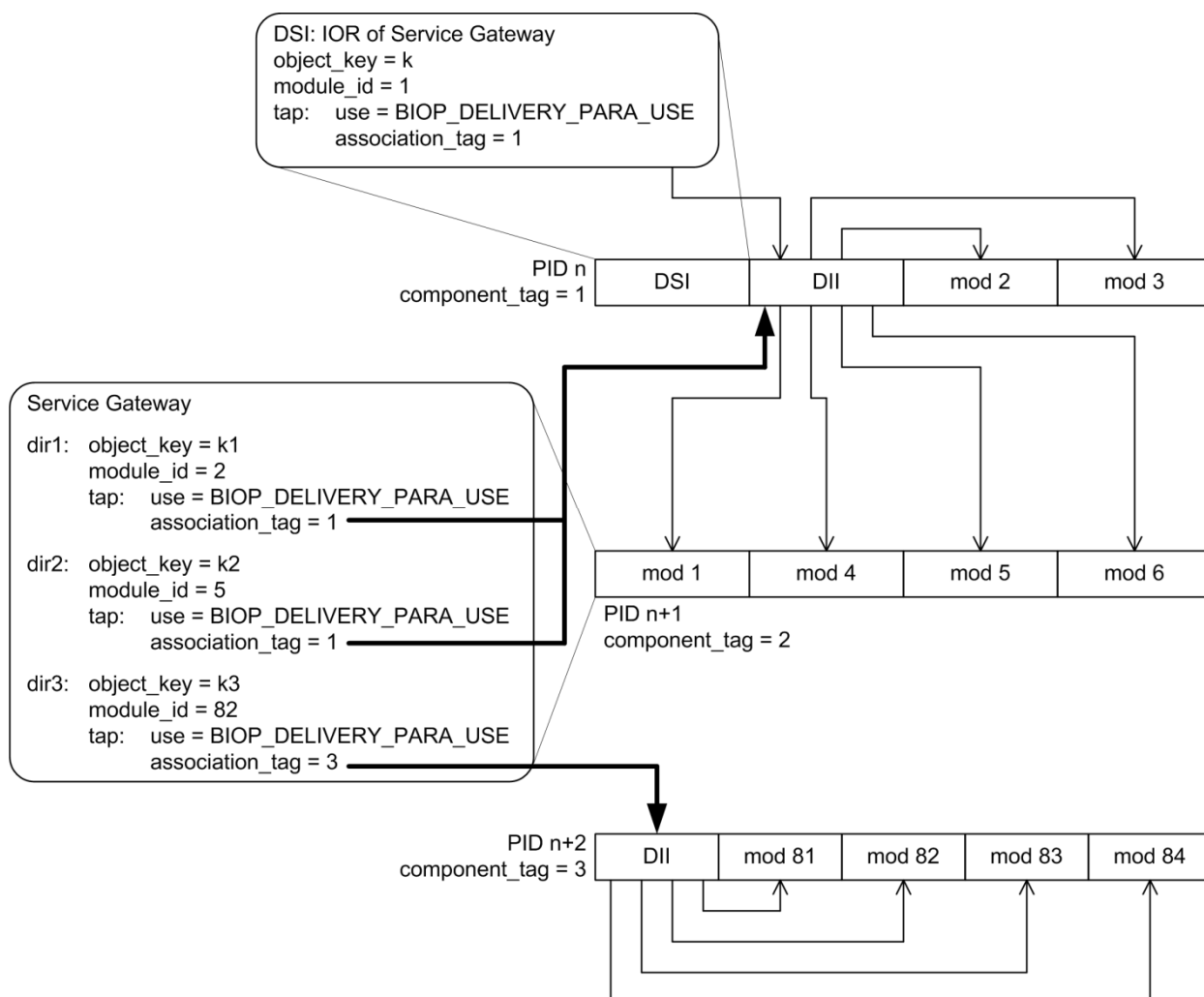


Figure 17.6: First example carousel

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 shall be "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 17.7. 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.

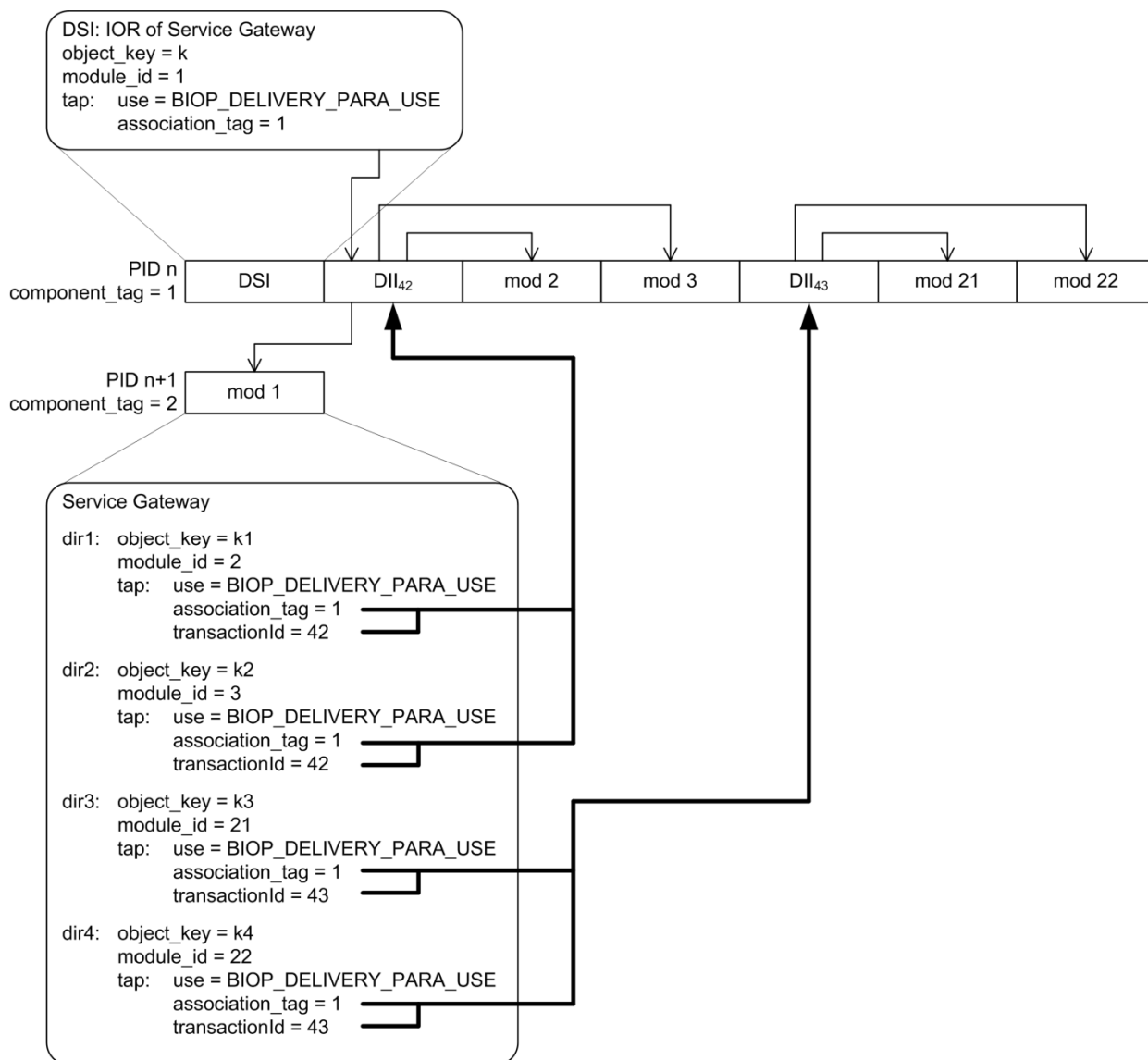


Figure 17.7: Second carousel example

17.18 Possible uses of persistent storage (informative)

17.18.1 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.

17.18.2 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).

17.18.3 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.

17.18.4 Persistent store size

Receivers shall only guarantee that 1 024 bytes of persistent storage is available. Consequently, applications should be authored to write at most 1 024 bytes of data, unless they are coded defensively to fall back to an alternative option should a `StorePersistent` elementary action fail.

17.18.5 Interaction channel

Authors should be aware that data written to persistent storage (including true persistent storage) may be read by any application. Such data could potentially be sent over the interaction channel (if the receiver supports `InteractionChannelExtension`).

17.19 Use of true persistent storage

True persistent storage is not intended to store transient or frequently changing data and as a result is limited to a maximum of 64 writes per file/service per 28 day period. Typically, applications will store a unique ID assigned by a server. This can be used as a key by the application to reference a database on the server to retrieve personalized data.

Authors should write their applications in such a way that it is not possible to use up their 64 writes per file/service per 28 day period (e.g. add a date stamp to the file to enable a check to only write the file once a day). The author could also add defensive code into their application, such that if the 64 writes are used up, the application falls back to an alternative non-receiver storage solution, so as not to degrade the user experience. Applications shall not stop working if they have run out of 'writes', even if it is caused by unusual user behaviour (e.g. a user changing their login multiple times).

17.20 Hints and tips for good authoring

17.20.0 Introduction

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.

17.20.1 Structure of the file system

17.20.1.1 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 can be found more quickly when searching.

17.20.1.2 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.

17.20.2 Structure of the object carousel

17.20.2.1 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).

17.20.2.2 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 can cause other commonly used objects in the same module to be flushed from the receiver. For example, this can 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.

17.20.2.3 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.

17.20.2.4 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 the present document will define the appropriate behaviour.

17.20.2.5 Carriage of content for HD receivers in DSMCC carousels

Application authors may wish to provide high resolution versions of certain graphics for use by HD-capable receivers. Such images will not be accessed by standard definition receivers. Application authors can minimize the impact of such content on the caching of carousel data by SD-only receivers by placing it in modules that contain only content for HD receivers.

17.20.2.6 Carriage of "auth" files

Application authors should minimize the number of `auth.tls.<x>` and `auth.cert.<x>` files to ensure speedy access to HTTP data, and should not author applications that rely on excessive numbers of such files, (e.g. 1 000+). Ideally, all `auth.servers`, `auth.tls.<x>` and `auth.cert.<x>` files should be placed in the same carousel module.

17.20.3 Use of memory

17.20.3.1 Forked resident programs

When a resident program is invoked with `Call` there shall only be 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 11.10.16 the Forked resident program shall act 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.

17.20.3.2 Original content never goes away

Note that the requirement to support cloning means that each `Ingredient` shall hold a copy of its `OriginalContent` forever and that the `OriginalContent` shall be 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.

17.20.3.3 Multiple references to the same content

If multiple `Ingredients` reference the same file normally only one copy of the content should be 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 should prevent this problem.

17.20.3.4 Simultaneous file requests

Loading of many files simultaneously places demands on receivers in terms of memory required to hold state information for the file request. Authors should endeavour to limit the number of simultaneous file requests to avoid any degradation in performance.

NOTE: Scenes with a large number of initially-active `Ingredients` with referenced content will initiate large numbers of file requests simultaneously.

Applications should not cause more than 401 simultaneous file requests to be made. However, authors should not expect even this number of requests to be possible where there are large demands on memory from other parts of the MHEG application.

17.20.4 Encoding of reserved fields

To ensure future compatibility all reserved fields shall be set to 0 unless otherwise specified.

17.20.5 Presentation of short audio clips over HDMI

Authors should be aware that audio clips shorter than a few seconds may not be presented on receiver installations connected with HDMI. This is due to the characteristics of the HDMI handshake protocol.

17.21 GetEngineSupport feature strings

17.21.1 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.

17.21.2 Engine identification

The present document provides two mechanisms for applications to obtain information about the receiver it is running on. Normally, applications should 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 11.10.14.1) provides a means for an application to find out the set of UniversalEngineProfile(N) (see clause 11.4.1) 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.

17.21.3 Extensions

A receiver may report itself as conformant with the present document even though it may not implement some extensions specified. (See clause 1.1). Therefore an application that uses functionality offered by these extensions should first check for their availability by using the appropriate GetEngineSupport string.

17.22 Appearance of Application Visible with no Scene

It is implementation dependent whether active Visible objects in an MHEG Application should be visible before the first Scene transition (but note that the presentation of any pre-existing stream components should initially continue as specified in clause 14.2.3).

Visible objects not intended for immediate presentation shall be defined 'InitiallyActive false'.

17.23 Colour representation

Application authors should be aware that attempting to use colours that are not present in the colour palette may result in inconsistent appearance between receivers. Authors should further note that blending of semi-transparency within the graphics plane may give differing results. Consequently, colours should be chosen carefully to ensure that graphics appear as intended. In the extreme case, badly chosen colours could become invisible against similarly coloured background objects.

17.24 Text Width

Text that fits in a particular size box on one receiver may not fit on another conformant receiver when text wrapping is disabled due to the way truncation is specified in the present document. This is most likely to affect centred and end justified text.

Authors shall apply the logical text width rules to ensure that text will display as intended on all conformant receivers.

17.25 Interaction Channel

17.25.1 Example of SetHybridFileSystem resident program

The use of the SHF ResidentProgram is demonstrated by the following example:

```
SHF("//i/", "http://iptv1.bbc.co.uk/content/ http://iptv2.bbc.co.uk/content/")
SHF("//i/r/", "DSM://c/r/")
SHF("//c/c2", "http://iptv1.bbc.co.uk/c2_web DSM://c/c2")
```

With these three calls to the SHF ResidentProgram, the mapping table is set up as shown in table 17.2.

Table 17.2: Mapping table, SetHybridFileSystem Example

Pathname	Mapping
//	DSM://
//i/	http://iptv1.bbc.co.uk/content/ http://iptv2.bbc.co.uk/content/
//i/r/	DSM://c/r/
//c/c2	http://iptv1.bbc.co.uk/c2_web DSM://c/c2

The hybrid file space resulting from this mapping table is shown diagrammatically in figure 17.8.

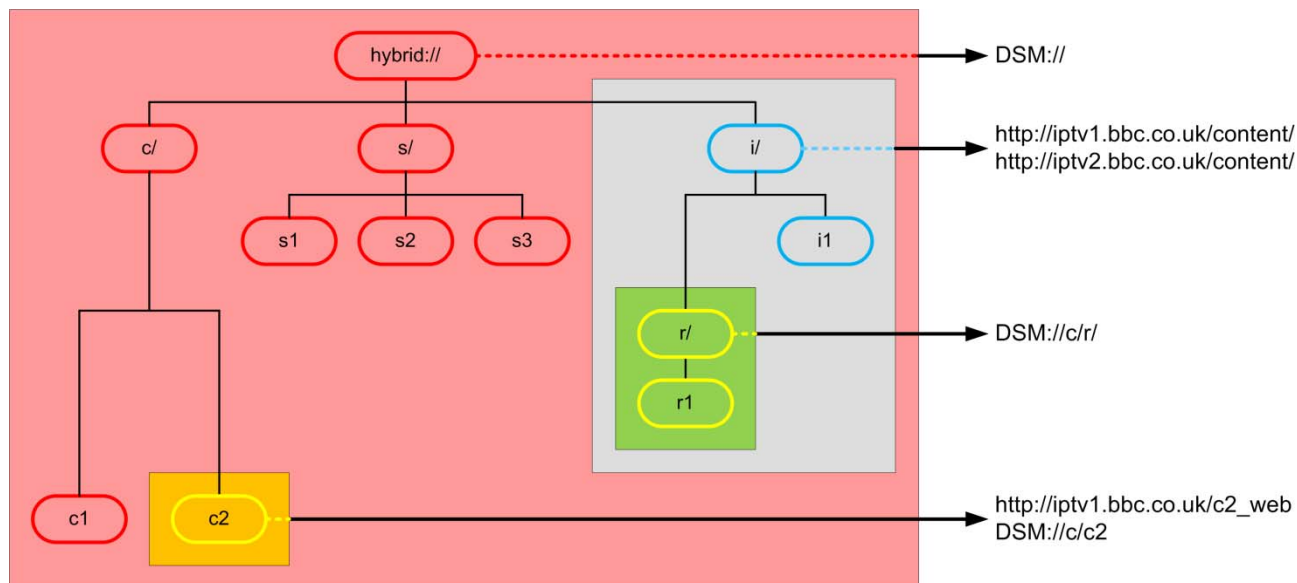


Figure 17.8: Example hybrid file space

According to this mapping table, the following filenames are resolved as indicated below. Where a filename is shown as resolving to multiple locations, the second location is used if the first location is found to be unavailable.

```
//i/i1    → http://iptv1.bbc.co.uk/content/i1
           → http://iptv2.bbc.co.uk/content/i1
//s/s1    → DSM://s/s1
//i/r/r1  → DSM://c/r/r1
//c/c1    → DSM://c/c1
//c/c2    → http://iptv1.bbc.co.uk/c2_web
           → DSM://c/c2
```

17.25.2 Example of ReturnData resident program

The ReturnData resident program can be used for any situation in which it is desirable to send data to a remote server using HTTP POST. A typical use case would be that of allowing the viewer to vote on some issue.

In this case, the application allows the viewer to choose from a number of items and then POST data of the form "vote=A", "vote=B" and so on. The application can, if its author wishes, check the response code from the server in order to act defensively should an error occur.

There is no requirement for a server to return any data. It can, however, return data (with an appropriate Content-Type header), which, regardless of the Content-Type header, shall be interpreted by the resident program as an OctetString. Typically, any returned data will be of type text/plain, perhaps a message to be displayed by the application, although arbitrary binary data (such as an image) could also be returned. Alternatively, a returned string could be cast into a ContentReference and used to retrieve further content.

Another possibility could be for regional variants of an MHEG application to send different data e.g. "vote=C®ion=wales", thereby facilitating voting by region.

EXAMPLE: Use of the RDa ResidentProgram:

```
{:ResidentPrg 100
  :InitiallyActive False
  :Name "RDa"
}
{:BooleanVar 101
  :OrigValue False
}
{:ContentRefVar 102
  :OrigValue :ContentRef "/content"
}
{:IntegerVar 103
  :OrigValue 0
}
{:OStringVar 104
  :OrigValue ""
}
...
:Call (100 101
  :GOctetString "http://www.somesite.com/cgi-bin/example.cgi"
  :GOctetString "name1"
  :GOctetString 'String wi?th var=C3=80ious %characters~'
  :GOctetString "name2"
  :GInteger 275
  :GOctetString "name3"
  :GBoolean True
  :GOctetString "name4"
  :GContentRef :IndirectRef 102
  :GOctetString "name5"
  :GObjectRef ("/scene.mhg 20")
  :GInteger :IndirectRef 103 // Response code
  :GOctetString :IndirectRef 104 // Response data
)
```

This call to the ReturnData resident program will result in the following data being sent to the server www.somesite.com

```
POST /cgi-bin/example.cgi HTTP/1.1
Content-Length: 111
Content-Type: application/x-www-form-urlencoded
[Other headers]
name1=String+wi%3Fth+var%C3%80ious+%25characters~&name2=275&name3=true&name4=%2Fcontent&name5=%2F
scene.mhg%2C20
```

If the POST request was successful, the server will reply HTTP/1.1 200 OK and the response code (returned in object 103) will have the value 200. The server may return some data such as:

```
Content-type: text/plain
Your vote has been registered
```

in which case the response data (returned in object 104) will have the value "Your vote has been registered".

17.25.3 GroupIdentifiers in the hybrid file system

When an Application or Scene is accessed through the hybrid file system, the corresponding file will be delivered by one of the other file systems available to the receiver. To ensure that the rules for mapping object references on files are not broken, application authors should avoid encoding the source in files that may be accessed through the hybrid file system. See clauses 9.3.4.2 and 16.3.1.

17.25.4 Cache priority in the IC file system

The default value of ContentCachePriority and GroupCachePriority is 127, which enables transparent caching (see clauses 15.4.3 and 15.4.4). This behaviour could degrade the performance of the IC file system in certain circumstances. When requesting files that may be delivered over the interaction channel and do not change dynamically, authors should use an even, non-zero value of ContentCachePriority or GroupCachePriority.

17.25.5 Use of Spawn with the hybrid file system

Authors should be aware that when a spawned application exits, the state of the hybrid file system that is used to restart the original application will be the state that was in force at the time of the spawn.

Consider the following sequence:

- The hybrid file system mapping table is in state 0
- Application 'A' starts
- Application 'A' sets the mapping table to state 1
- Application 'A' spawns application 'B'
- Application 'B' set the mapping table to state 2
- Application 'B' quits
- The mapping table is reset to state 1
- Application 'A' is restarted and any OnRestart actions executed

Authors (in this scenario, the author of application 'A' in particular) should note that the application object for application 'A' will be reacquired, and any OnRestart actions will be executed, in the context of state 1 of the mapping table. This may not be the same as state 0.

It is possible to configure state 1 so that the instance of application 'A' that loads after application 'B' terminates can be different from the instance that loaded originally under state 0. Application authors should avoid changing the hybrid mapping table such that the application itself is no longer available.

17.25.6 Interaction channel engine events

When an application fails to access an object or content over the interaction channel, one of three engine events may be generated (in addition to ContentRefError or GroupIDRefError).

The ICLocalError engine event indicates that no interaction channel connection is available. An application may wish to display a message asking the viewer to check that their receiver is connected and that the correct settings have been entered.

Application authors may choose to use the GetICStatus resident program and/or to monitor the ICStatusChanged engine event to modify the application behaviour to avoid attempting accesses when no connection is possible and optionally to inform the viewer of the limitations on the application when no connection is available.

The ICNetworkError engine event indicates that a remote server did not respond. The problem may lie within that part of the network under the viewer's control, or elsewhere. An application may wish to display a message asking the viewer to check that their network is correctly set up and to try again.

The ICRemoteError engine event indicates that a remote server responded but could not provide the requested resource. An application may wish to display a message explaining that this resource is temporarily unavailable.

Table 17.3 summarizes a number of situations that may occur and the Engine Events that may be generated as a result.

Table 17.3: Engine Events Example

	Reason for failure	Content/GroupRef event generated	IC event generated
1	Server access file is not in the current DSM-CC object carousel	Y	
2	Server access file cannot be parsed	Y	
3	Access to server is not allowed	Y	
4	Host name could not be resolved	Y	Network
5	Connection cannot be created (but IC is active)	Y	Local
6	TLS connection failed for various reasons	Y	
7	Connection timed-out	Y	Network
8	Connection closed before request was sent	Y	Network
9	Server time-out during reply	Y	Remote
10	HTTP server failure (HTTP code 5xx)	Y	Remote
11	Other HTTP failures that are not 4xx	Y	Remote
12	Hash file cannot be loaded	Y	
13	Hash file cannot be parsed	Y	
14	File is not in the hash file (and digest_count is not zero)	Y	
15	Invalid pathname in hash file (and digest_count is zero)	Y	
16	File hash is invalid	Y	
17	First signature file cannot be loaded	Y	
18	Signature file(s) cannot be parsed	Y	
19	Signature doesn't match	Y	
20	First certificate file cannot be loaded	Y	
21	Certificate file(s) cannot be parsed (or certificate cannot be parsed)	Y	
22	No matching certificate is found	Y	
23	Out of memory during this process	N	

17.25.7 Presenting restricted content

Applications that present streams delivered by the Interaction Channel should consider using the PromptForGuidance resident program to verify that the viewer should be allowed to view the content. The policy for determining suitability for presentation is not defined in the present document but in general any content that is not believed to be "suitable for all" should be verified before presentation.

The PromptForGuidance resident program provides an optional OctetString that describes the reason for the restriction. The text used should be human readable, for example "Contains scenes of mild peril". Note that where parental controls are not implemented or are disabled the resident program will return immediately without presenting the text to the viewer.

The guidance mechanism for the Interaction Channel is independent of any other guidance mechanisms defined in the present document.

17.25.8 Use of remote control keys to control A/V streams

Receivers supporting ICStreamingExtension may also provide recording functionality and may have additional keys on the remote control such as play, stop, fast forward and rewind.

Users of such receivers will naturally expect these DVR keys to function when viewing content streamed over IP.

Input event register 6 allows applications to access any DVR control keys that exist on the remote control. Applications should handle these user input events in the appropriate way. However, applications should also provide alternative means of controlling playback of IP-delivered streams since not all receivers can provide these keys. Applications should query the NonLinearPlaybackKeys engine support string to check whether a minimum set of these keys is available on the receiver.

Since ICStreamingExtension does not support variable speed playback, applications should map both the 'skip forwards' and 'fast forwards' user input events to a skip function, where one is provided by the application. Similarly, 'skip backwards' and 'rewind' should both perform a skip back function if one exists.

17.25.9 Use of './' in the Hybrid File System mapping table

The interpretation of the './' form in the pathName parameter to the SHF Resident Program (see clause 11.10.13.1) is implementation specific and should be avoided.

17.25.10 Use of SetCounterPosition on IC streams

Applications may present IC Stream content from an arbitrary mid-point by calling the SetCounterPosition Elementary Action on the associated Stream object. When doing so the application should choose a CounterPosition that corresponds to an H.264 IDR Frame otherwise the stream decoder will discard any preceding content until the next IDR Frame is acquired. Further, if the stream is being started from a mid-point or after another stream has been decoding then the decoder may also need to re-acquire the stream's PSI tables.

17.25.11 CounterEndPosition of IC Streams

Following a SetData targeted to an IC Stream object the value of CounterEndPosition may be undefined in some receivers. Applications should explicitly call SetCounterEndPosition to the desired value following a ContentAvailable event.

17.25.12 Cookie support

See clause 15.7.5. Authors should not rely on a cookie being stored over a power cycle. The GetCookie and SetCookie ResidentPrograms enable cookies to be stored and retrieved by the application.

17.26 HD intelligent rendering

17.26.1 Sizing images for HD presentation

Depending on the location of an object in the SD co-ordinate system, its final size in the HD co ordinate system may vary by 1 pixel.

If an image were to be provided at an HD resolution, ideally it should be prepared for its new dimension in the HD co-ordinate system. This can be calculated based on its position in the SD co ordinate system using the co-ordinate transform given in clause 12.11.3.1. If this calculation is not performed, or if the same image is to be used in different locations in different scenes, then HD images should be prepared for the largest possible size for the HD resolution in question. The image will then either fill the image object exactly, or it will be cropped by 1 pixel at the right-hand or bottom edge such that it fits the available size. This will prevent any gaps between image objects and objects that the image is intended to touch.

17.26.2 Appearance of objects on HD resolution graphics planes

Adjacent pixels in the SD co ordinate system do not map to adjacent pixels in an HD co ordinate system. Furthermore, the addressable pixels in the HD co ordinate system are not evenly spaced. This means that objects spaced equally in the SD co-ordinate system may have spacing that varies by 1 HD pixel when rendered on an HD graphics plane. This includes lines of text within a multi-line Text object.

This effect is generally too small to be visible. However, if an application moves an object one SD pixel at a time across the screen, receivers performing HD intelligent rendering will show the object with slightly uneven motion as the object moves between addressable HD pixels.

Depending on the location of an object in the SD co ordinate system, its final size in the HD co ordinate system may vary by 1 pixel. Again, the effect of this is generally too small to be visible but an effect may be noticed on very small moving objects. Authors may hide the effect of these size changes by using a PNG image on a transparent background inside a larger MHEG Bitmap object as an alternative to small rectangle objects.

17.26.3 PNG bitmap resolution

For compatibility with receivers supporting HDGraphicsPlaneExtension, authors should avoid encoding resolution information in SD PNG images.

17.27 Non-destructive service tunes

17.27.1 Non-availability of broadcast file system during tune

During a non-destructive service tune (see clause 8.1.7) any object carousel file requests made by the running application will be queued and only resolved following completion of the tune and the successful attachment of a carousel in the new service. Hence, any visual effects employed to "distract" the viewer during the tune need to be based on content already loaded into active MHEG-5 objects or available from a file system other than the broadcast file system. Fortunately, there are a number of ways of achieving animated visual effects within this constraint, including:

- Cycling the appearance of one in a sequence of MHEG-5 objects using either the Run/Stop or DisplayStack manipulation actions.
- Executing SetData actions where the NewContent exchanged attribute shall be set or refer to included data.

17.27.2 Carousel structure

As specified in clause 8.1.7 the new Current Carousel, attached to as a result of a non-destructive service tune, need not be the same as the previous Current Carousel in the previous service at the point that the application initiated the non-destructive tune. This is true both in terms of the broadcast file system that it delivers and the encoding of the underlying Object and Data Carousel structures. It is up to the author to ensure that the new carousel delivers a broadcast file system that is appropriate for the running application. Whilst the broadcast file system delivered by this new carousel need not contain the currently running MHEG-5 Application object, it has to contain a directory structure corresponding to that in which the MHEG-5 Application object was located when launched if file references with the path origin set to '/' are used.

EXAMPLE: An application "DSM://foo/bar/a" will retrieve content with the path origin set to '/' from "DSM://foo/bar". Unless "DSM://foo/bar" exists in the target carousel, references with the path origin set to '/' will fail.

17.27.3 Behaviour of Spawn during tune

During a non-destructive service tune (see clause 8.1.7) the application stack will be reset. Consequently the behaviour of Spawn ElementaryActions shall be undefined during this period and therefore should be avoided until the "NonDestructiveTuneOK" Engine Event has been received, indicating successful completion.

17.28 Signalling MHEG Applications for all types of Terminal

If an MHEG application is present for a service, and an AIT subtable is present on that service, then MHEG applications should be signalled using both the classical signalling and the AIT signalling.

17.29 Avoid moving carousel components

When AIT signalling is used to control MHEG lifecycle there is an implied association between the object carousel's auto-boot component as identified by the transport_protocol_descriptor in the AIT and the carousel id identified by the carousel id descriptor in the PMT on the same component clause 9.3.7.4. Classical application signalling allows the auto-boot components PID to change without causing the running application to be terminated clause 9.3.7. When AIT signalling is employed, moving the auto-boot component in this way breaks the implied association between carousel id and auto-boot component unless an equivalent change occurs in the AIT. As the signalling exists in both PMT and AIT it is not possible to update both in the receiver simultaneously and receiver behaviour during this transient period is undefined.

17.30 Value of LastAnchorFired for boundary events

The effect of generating boundary events in HyperText objects has been clarified in clause 13.8.3.3. However, application authors should be aware that a small number of implementations conformant to a previous version of this specification may not exhibit this behaviour. Special provision may need to be made for these devices when using HyperText functionality.

17.31 Case sensitivity in server names

Application authors are recommended to not use mixed case in server names, instead using lower case. Use of mixed case is known to cause problems for some well known server implementations and is therefore not recommended unless the servers used are known to be unaffected. This restriction only applies to server names; URL path objects remain case sensitive.

Annex A (informative): The user experience

A.1 Introduction

A.1.0 Scope

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.



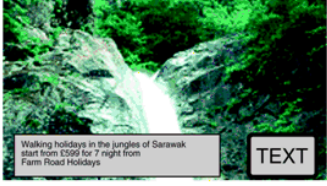
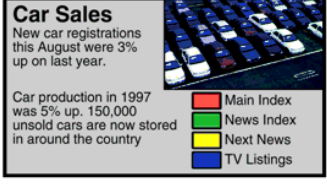

This broadcaster controlled behaviour should be seen as a concept model for how services may appear. As broadcasters develop services, and gain experience from user feed-back, the detail of the behaviour they implement is likely to evolve.

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.

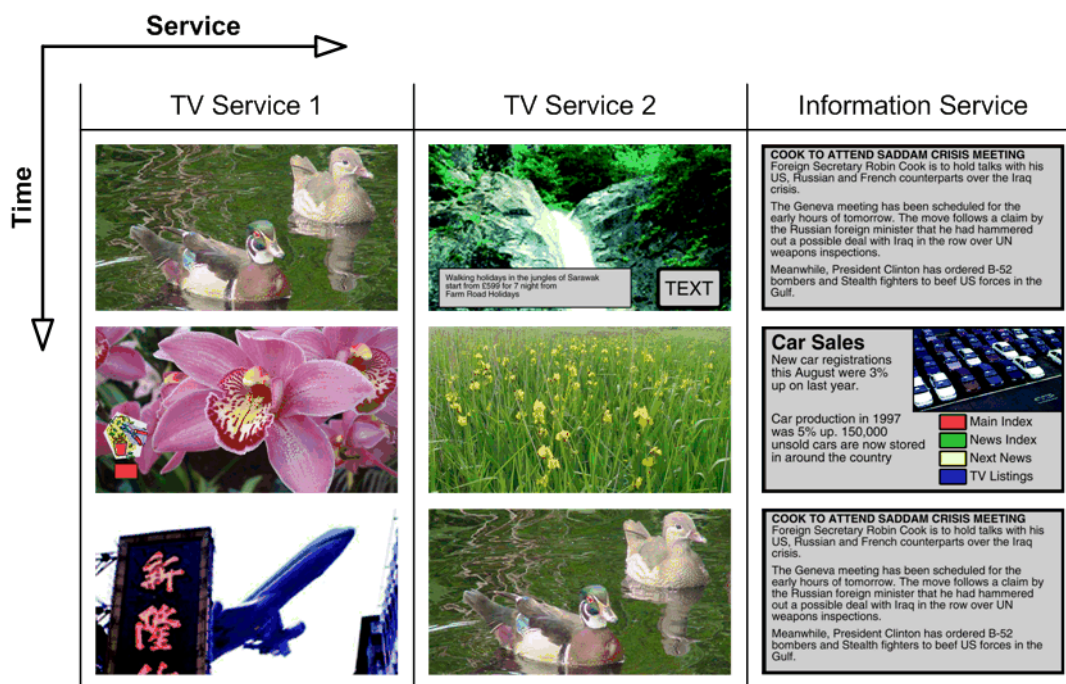


Figure A.1: What might be seen across channels and through time

A.2 User navigation

A.2.1 Channel change

A.2.1.0 Introduction

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.

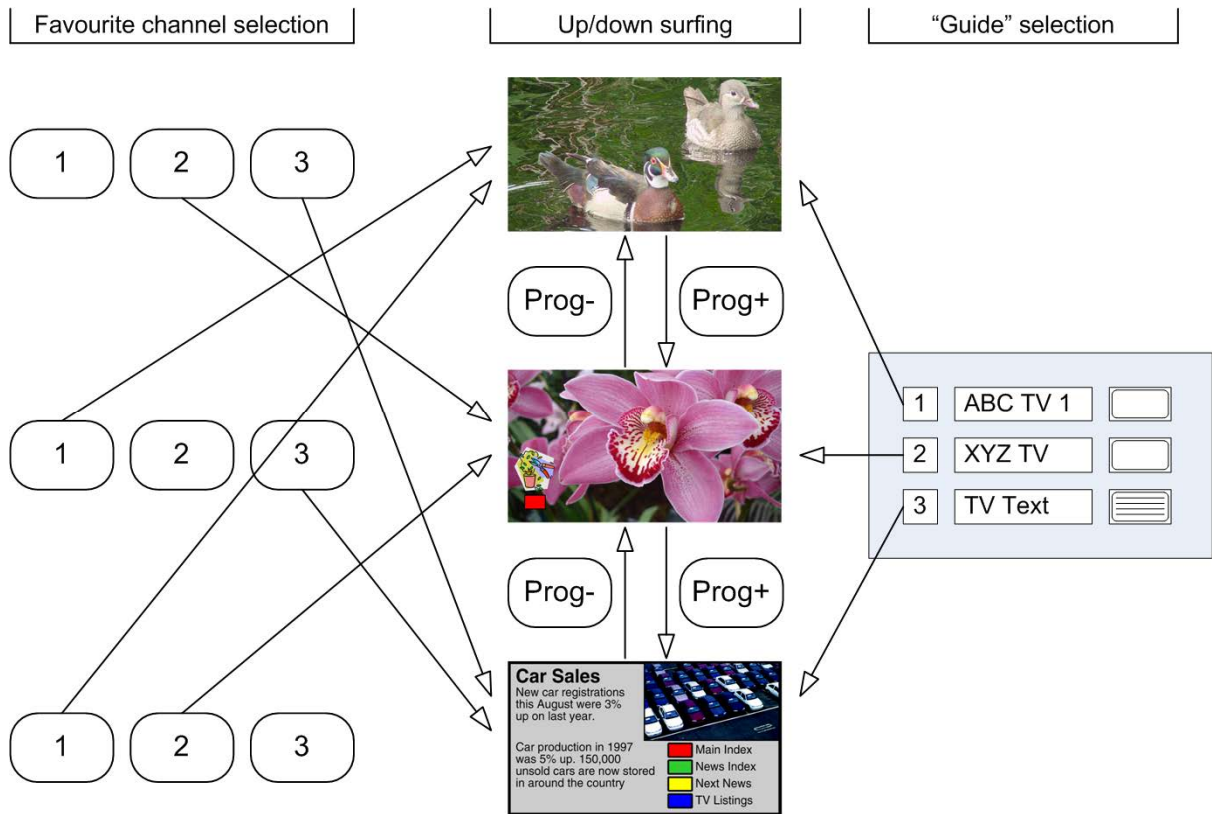


Figure A.2: Changing channels

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.

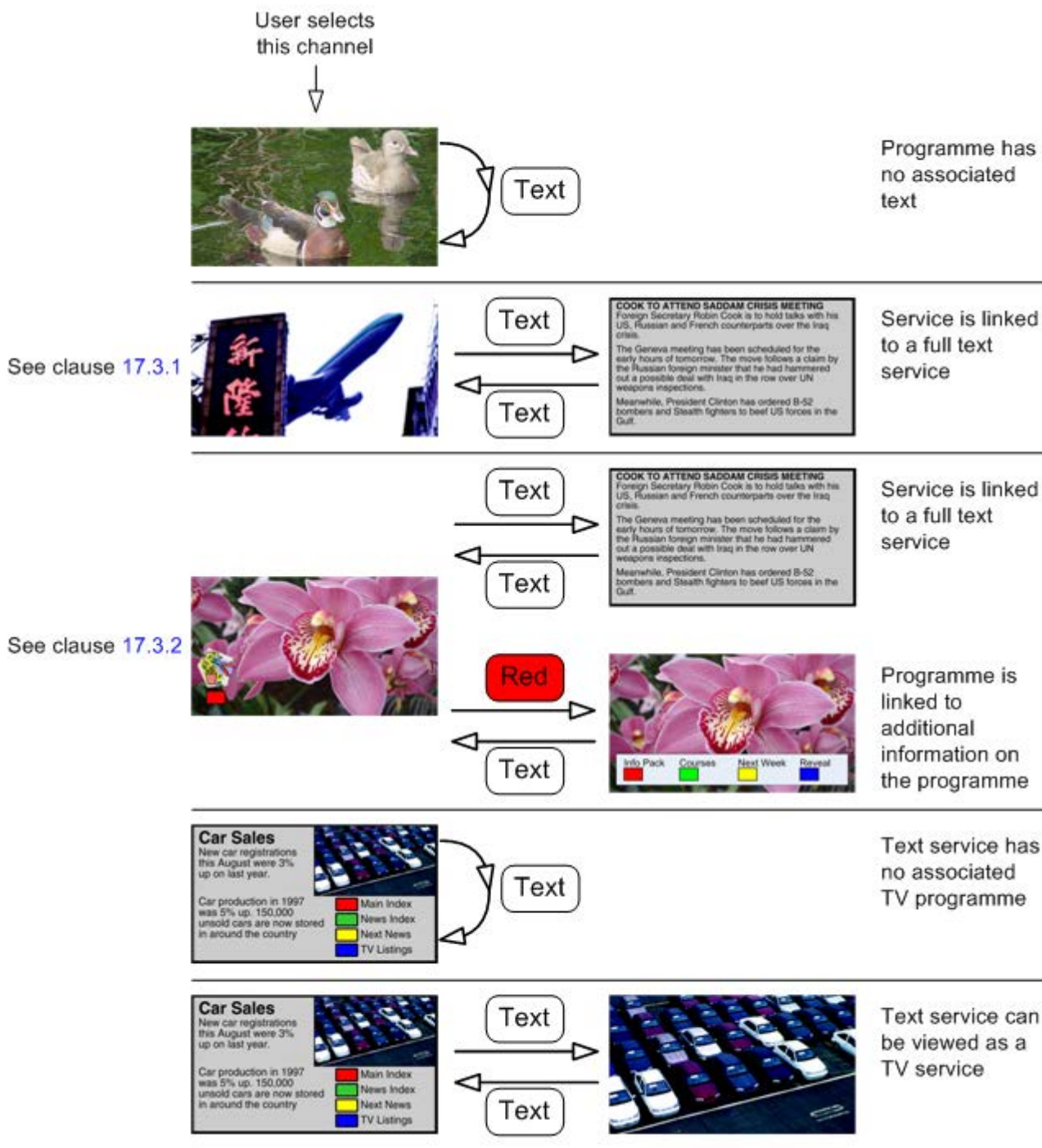


Figure A.3: Using the text button

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 should be exactly the same as if the user selected a channel from the receiver's "Guide".

See figure A.4.

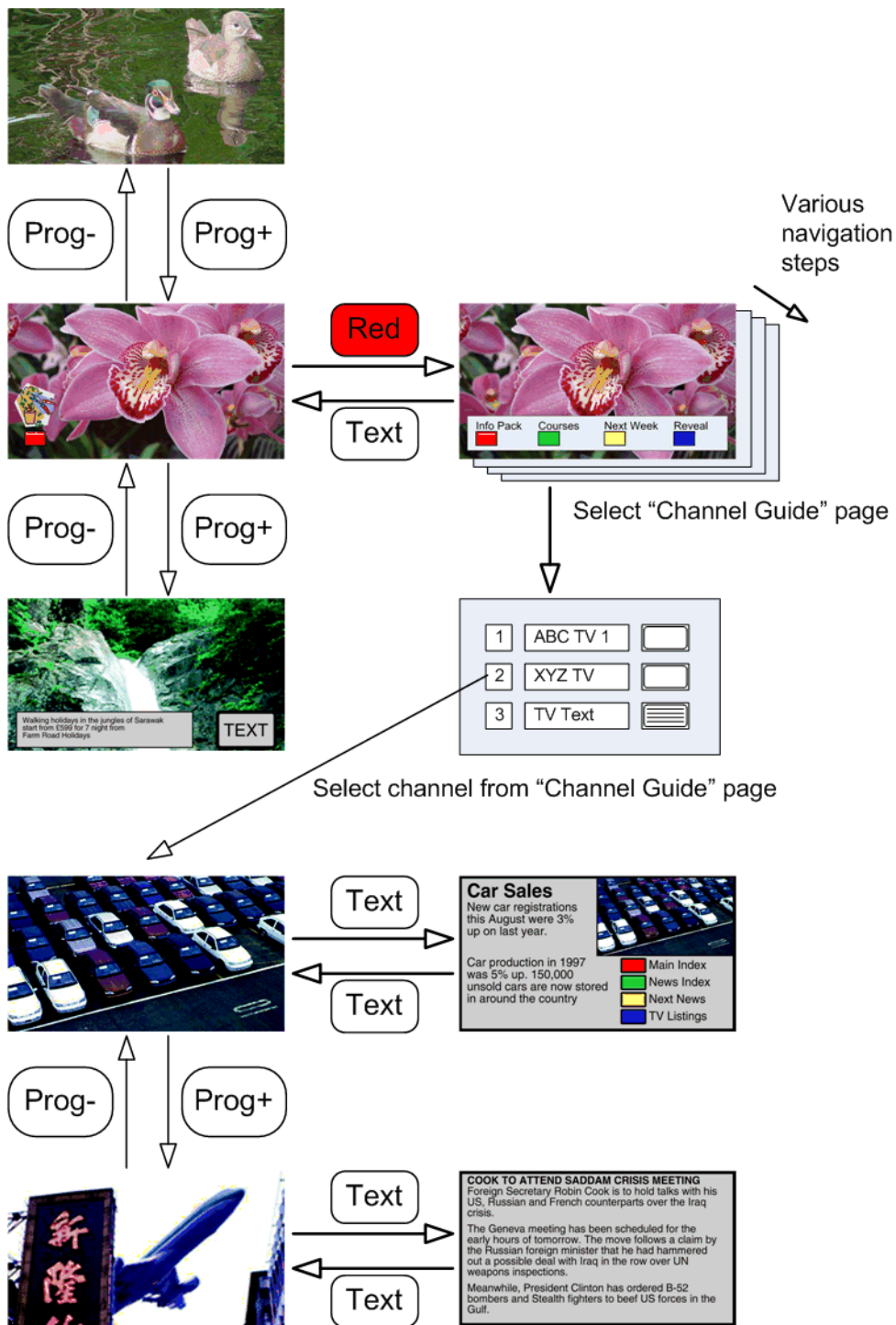


Figure A.4: Using a TV listings page to change channel

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 should return to the service they started from.

See figure A.5.

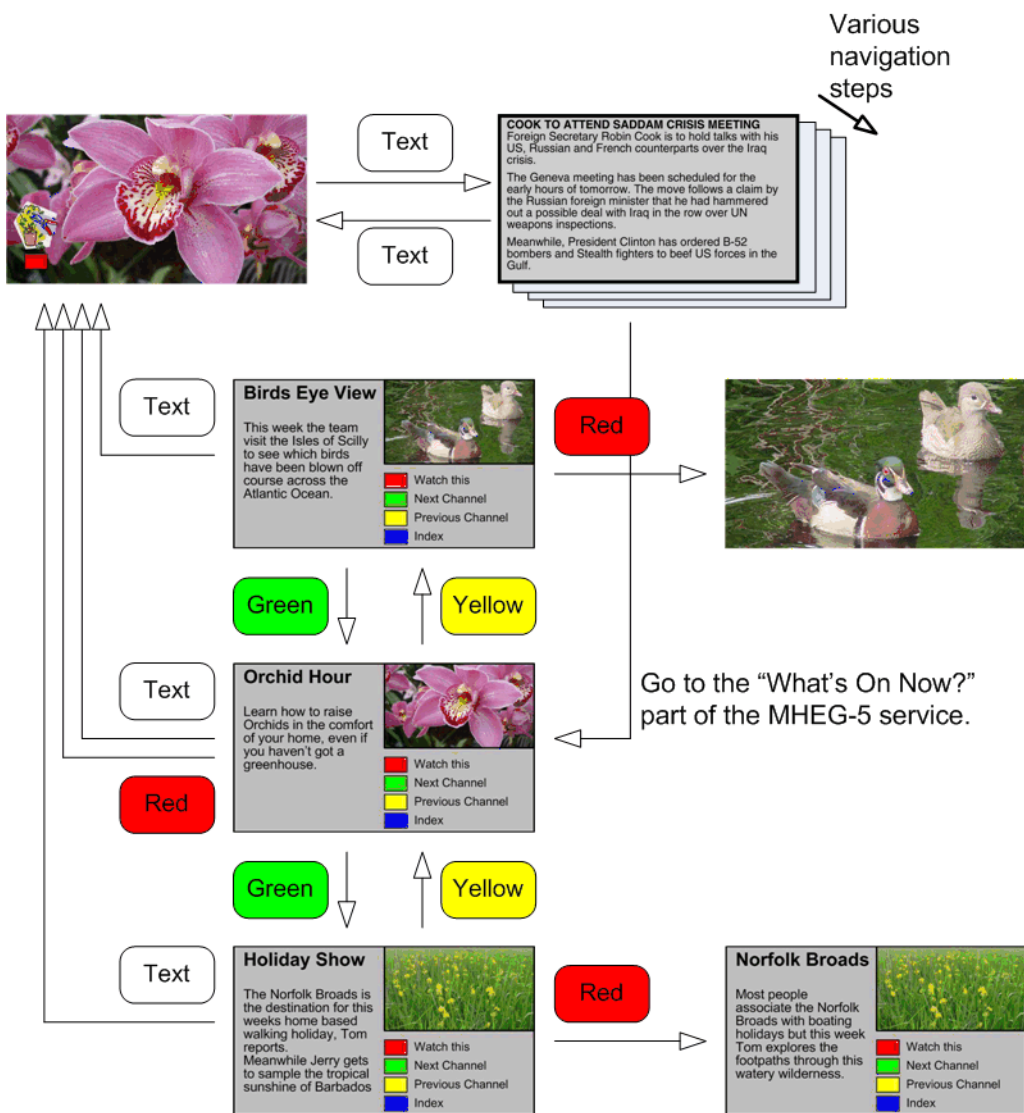


Figure A.5: A possible "What's on now?" TV browser

Annex B (normative): Allocation of codes for use with the present document

B.1 Application type codes

Table B.1 identifies the allocated values for `application_type_code` for use in the `data_broadcast_id_descriptor` with `data_broadcast_id` value 0x0106 that a receiver implementing the `ETSIEngineProfile1` shall recognize.

Table B.1: Application type codes

Description	Code Value (hex)	Code Value (dec)	Notes
NULL_APPLICATION_TYPE	0x0000	0	There is no auto-boot application. This value is intended for use where no auto-boot application is present but a <code>data_broadcast_id_descriptor</code> is needed to provide lifecycle signalling (such as following a non-destructive tune for receivers implementing <code>LifecycleExtension</code>).
UK_PROFILE_LAUNCH	0x0101	257	uk mheg launch
UK_PROFILE_BASELINE_1	0x0505	1 285	uk mheg profile 1,06

B.2 UniversalEngineProfile(N)

Table B.2: UniversalEngineProfile(N) GetEngineSupport "feature" string

N	Description
1 285	Baseline profile: the receiver is fully conformant with the present document.
NOTE:	Receivers based on the present document but which have not been verified as fully conformant shall not return true when N=1 285. Instead they shall return true when N=2.

History

Document history		
V1.1.1	November 2004	Publication
V2.1.1	March 2010	Publication
V2.2.1	March 2011	Publication
V2.3.1	March 2013	Publication
V2.4.1	April 2016	Membership Approval Procedure MV 20160617: 2016-04-18 to 2016-06-17