

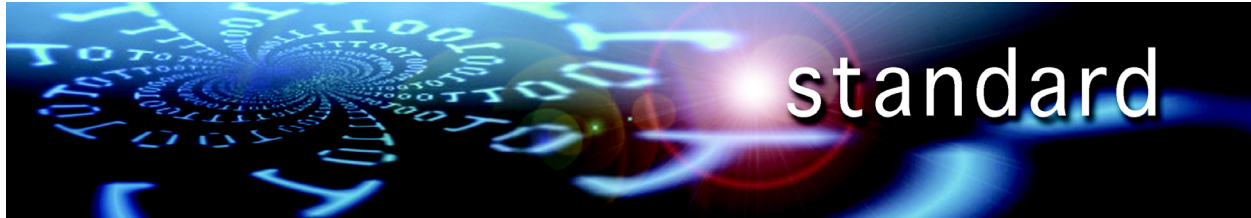


advancedtelevisionssystemscmmittteeinc

ATSC Standard: Advanced Common Application Platform (ACAP)

Document A/101, 2 August 2005

Advanced Television Systems Committee, Inc.
1750 K Street, N.W., Suite 1200
Washington, D.C. 20006



The Advanced Television Systems Committee, Inc., is an international, non-profit organization developing voluntary standards for digital television. The ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

Specifically, ATSC is working to coordinate television standards among different communications media focusing on digital television, interactive systems, and broadband multimedia communications. ATSC is also developing digital television implementation strategies and presenting educational seminars on the ATSC standards.

ATSC was formed in 1982 by the member organizations of the Joint Committee on InterSociety Coordination (JCIC): the Electronic Industries Association (EIA), the Institute of Electrical and Electronic Engineers (IEEE), the National Association of Broadcasters (NAB), the National Cable Television Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). Currently, there are approximately 140 members representing the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

ATSC Digital TV Standards include digital high definition television (HDTV), standard definition television (SDTV), data broadcasting, multichannel surround-sound audio, and satellite direct-to-home broadcasting. Contact information is given below.

Mailing address	Advanced Television Systems Committee, Inc. 1750 K Street, N.W., Suite 1200 Washington, D.C. 20006
Telephone	202-872-9160 (voice) 202-872-9161 (fax)
Web site	http://www.atsc.org
E-mail	standards@atsc.org

The revision history of this document is given below.

A/101 Revision History

A/101 approved	2 August 2005
Editor's Note: Table 10.3, "Semantics of the HTTPSPProfileBody," includes a <td> value for ProfileIdTag. Assignment of this value is in process. Once the value is available, this document will be editorially updated.	

Table of Contents

1. SCOPE	15
1.1 Purpose	15
1.2 Application	16
2. GENERAL CONSIDERATIONS	16
2.1 Format	16
2.2 Inclusion of GEM	16
2.3 Inclusion of OCAP	16
2.4 Addition of Non-ACAP Interfaces	16
2.5 Application Areas	16
2.6 Profiles	16
3. DEFINITIONS AND ABBREVIATIONS	17
3.1 Definitions from GEM	17
3.2 Definitions Introduced by OCAP	17
3.3 Definitions Introduced by ACAP	17
3.4 Abbreviations from GEM	17
3.5 Abbreviations Introduced by ACAP	17
3.6 Conformance Keywords	17
3.6.1 Section and Data Structure Syntax Notation	17
3.6.2 Informative Sections	18
4. REFERENCES	18
4.1 Normative References	18
4.2 Informative References	20
4.3 Reference Acquisition	21
4.3.1 ATSC Standards	21
4.3.2 CableLabs Specifications	21
4.3.3 CEA Standards	21
4.3.4 DVB Standards	21
4.3.5 ECMA Standards	21
4.3.6 ETSI Standards	21
4.3.7 IETF Standards	21
4.3.8 ISO Standards	21
4.3.9 SCTE Standards	21
4.3.10 W3C Standards	21
5. ARCHITECTURE	22
5.1 Support for ACAP-J Applications	22
5.2 Support of ACAP-X Applications	22
6. COMMON CONTENT FORMATS	22
6.1 General	22
6.2 Static Formats	23
6.3 Broadcast Streaming Formats	23
6.3.1 Video	23
6.3.2 Audio	23

6.3.3 Closed-Captioning	23
7. ACAP-J APPLICATIONS AND ENVIRONMENT	24
7.1 Behavior	24
7.1.1 Application Model	24
7.1.2 Destruction of Applications	24
7.2 Facilities	25
7.2.1 Java Content	25
7.2.1.1 Additional Java APIs	25
7.2.1.1.1 Closed Captioning	25
7.2.1.1.2 Locators	25
7.2.1.1.3 Events	25
7.2.1.1.4 Content Identification API	26
7.2.1.1.5 Extended SI API	26
7.2.1.2 Inter-Environment DOM Integration	26
7.2.1.3 Java Class Files	27
7.2.1.4 Integration of the JavaTV SI API	27
7.2.1.5 Addition of Non-ACAP Interfaces	27
7.2.1.6 Void	27
7.2.1.7 Semantics of java.io.File.lastModified() for Broadcast Carousels	27
7.2.2 Font Index Content	27
7.2.3 Archive Content	27
8. ACAP-X APPLICATIONS AND ENVIRONMENT	28
8.1 Behavior	28
8.1.1 Application Behavior	28
8.1.1.1 Clarifications	28
8.1.2 Resource Identifier Schemes	29
8.1.2.1 Restrictions	29
8.1.2.1.1 ecmaScript Scheme	29
8.1.2.1.2 lid Scheme	29
8.1.2.1.3 tv Scheme	29
8.1.2.2 Extensions	29
8.1.2.2.1 acap Scheme	29
8.1.2.2.2 exit Scheme	29
8.1.3 Event Processing	29
8.1.3.1 Restrictions	29
8.1.4 Trigger Processing	30
8.1.4.1 Restrictions	30
8.1.4.2 Extensions	30
8.1.4.2.1 Environment Triggers	31
8.1.4.2.2 org.atsc.trigger.start Trigger Types	31
8.1.4.2.3 Application Triggers	31
8.2 Facilities	31
8.2.1 Application Metadata Content	32
8.2.1.1 Modifications	33
8.2.1.1.1 Content Type	33
8.2.1.1.2 Document Type Definition	33

8.2.1.1.3 Document Type Declaration	33
8.2.1.2 Extensions	33
8.2.1.2.1 entity Element	33
8.2.1.2.2 initial Entity Type	33
8.2.1.2.3 permissionRequest entity Type	33
8.2.1.2.4 signature Entity Type	33
8.2.1.2.5 identifier Element	34
8.2.1.2.6 Permission Capability	34
8.2.1.2.6.1 type Parameter	35
8.2.1.2.6.2 target Parameter	35
8.2.1.2.6.3 actions Parameter	35
8.2.2 Graphics Content	35
8.2.2.1 Extensions	35
8.2.2.1.1 Image/MPEG Formats	35
8.2.3 Non-Streaming Video Content	35
8.2.3.1 Extensions	36
8.2.3.1.1 Video “Drip Feed” Format	36
8.2.4 Non-Streaming Audio Content	36
8.2.4.1 Extensions	36
8.2.4.1.1 MPEG Audio Layers	36
8.2.5 Streaming Video Content	36
8.2.6 Streaming Audio Content	36
8.2.7 Font Content	36
8.2.8 Archive Content	36
8.2.9 Markup Content	36
8.2.9.1 Restrictions	37
8.2.9.1.1 Resource Content Type References	37
8.2.9.1.2 Resource Access	37
8.2.9.1.3 Document Type Declaration	37
8.2.9.1.4 Namespace Declarations	38
8.2.9.1.5 legacy Application	38
8.2.9.1.6 intrinsic event Attributes	38
8.2.9.1.7 name Attribute	39
8.2.9.1.8 a (anchor) Element	39
8.2.9.1.9 frame Element	39
8.2.9.1.10 object Element	39
8.2.9.1.10.1 Active Content Object Element	39
8.2.9.1.10.2 Trigger Object Element	40
8.2.9.1.11 script Element	40
8.2.9.2 Extensions	40
8.2.9.2.1 Document Type Declaration	40
8.2.9.2.2 cite Attribute	41
8.2.9.2.3 event Attributes	41
8.2.9.2.4 longdesc Attribute	42
8.2.9.2.5 a (anchor) Element	42
8.2.9.2.5.1 Application Replacement and Launching	42

8.2.9.2.5.2 Service Selection	43
8.2.9.2.5.3 Service Component Selection	43
8.2.9.2.6 area Element	43
8.2.9.2.7 meta Element	43
8.2.9.2.7.1 Classpath Metadata Item	43
8.2.9.2.8 object Element	43
8.2.10 Stylesheet Content	44
8.2.10.1 Restrictions	44
8.2.10.1.1 Resource Content Type References	44
8.2.10.1.2 Media Types	44
8.2.10.1.3 Properties	44
8.2.10.1.3.1 atsc-nav-index Property	44
8.2.10.1.3.2 atsc-nav- <code>{left,right,up,down}</code> Properties	44
8.2.10.1.4 Property Values	45
8.2.10.1.4.1 <code><color></code> Property Value	45
8.2.10.2 Extensions	45
8.2.10.2.1 Font Face Rule	45
8.2.10.2.2 Viewport Rule	45
8.2.10.2.2.1 Viewport Descriptors	46
8.2.10.2.2.1.1 initial-container Descriptor	46
8.2.10.2.2.1.2 region Descriptor	47
8.2.10.2.2.1.3 resolution Descriptor	48
8.2.10.2.3 Media Types	48
8.2.10.2.4 Properties	48
8.2.10.2.4.1 acap-dynamic-refresh Property	49
8.2.10.2.4.2 crop Property	49
8.2.10.2.4.3 font Property	50
8.2.10.2.4.4 nav-index Property	50
8.2.10.2.4.5 nav- <code>{left,right,up,down}</code> Properties	51
8.2.10.2.4.6 opacity Property	51
8.2.10.2.5 Property Value Types	52
8.2.10.2.5.1 <code><color></code> Property Value Type	52
8.2.11 Script Content	52
8.2.11.1 Restrictions	52
8.2.11.1.1 HTML Module Objects	53
8.2.11.1.1.1 HTMLDocument Object	53
8.2.11.1.1.2 HTMLFormElement Object	53
8.2.11.1.1.3 HTMLImageElement Object	53
8.2.11.1.1.4 HTMLObjectElement Object	53
8.2.11.1.2 StyleSheets Module Objects	53
8.2.11.1.3 Event Types	54
8.2.11.1.3.1 HTML Event Types	54
8.2.11.1.4 Environment Module Objects	55
8.2.11.1.4.1 Navigator Object	55
8.2.11.2 Extensions	55
8.2.11.2.1 Event Module Objects	55

8.2.11.2.1.1 ApplicationEvent Object	55
8.2.11.2.1.1.1 ApplicationEvent::detail	55
8.2.11.2.1.1.2 ApplicationEvent::initApplicationEvent	55
8.2.11.2.1.2 TimerEvent Object	56
8.2.11.2.1.2.1 TimerEvent::detail	56
8.2.11.2.1.2.2 TimerEvent::interval	56
8.2.11.2.1.2.3 TimerEvent::initTimerEvent	56
8.2.11.2.1.3 TriggerEvent Object	56
8.2.11.2.1.3.1 TriggerEvent::timeType	57
8.2.11.2.1.3.2 TriggerEvent::time	57
8.2.11.2.1.3.3 TriggerEvent::detail	57
8.2.11.2.1.3.4 TriggerEvent::initTriggerEvent	57
8.2.11.2.2 Event Types	57
8.2.11.2.2.1 HTML Event Types	58
8.2.11.2.2.1.1 org.atsc.document.domstable Event	58
8.2.11.2.2.1.2 org.atsc.document.load Event	58
8.2.11.2.2.1.3 org.atsc.document.unload Event	58
8.2.11.2.2.2 Application Lifecycle Event Types	59
8.2.11.2.2.2.1 org.atsc.application.started Event	59
8.2.11.2.2.2.2 org.atsc.application.suspending Event	59
8.2.11.2.2.2.3 org.atsc.application.resumed Event	60
8.2.11.2.2.2.4 org.atsc.application.terminating Event	60
8.2.11.2.2.3 Timer Event Types	61
8.2.11.2.2.4 Trigger Event Types	61
8.2.11.2.3 Environment Module Objects	61
8.2.11.2.3.1 Window Object	61
8.2.11.2.3.1.1 TimerDispatcher::startTimer	62
8.2.11.2.4 Inter-Environment Bridge	62
8.2.11.2.4.1 Packages Object	62
8.2.11.2.4.2 Package Object	62
8.2.11.2.4.3 Java Class Object	62
8.2.11.2.4.4 Java Method Object	63
8.2.11.2.4.5 Behavior of Java Objects in ECMAScript	63
8.2.11.2.4.6 Explicit Method Selection	64
8.2.11.2.4.7 Method Signature Matching	64
8.2.11.2.4.8 Subclassing	64
8.2.11.2.4.9 Exceptions	65
8.2.11.2.4.10 Security	65
8.2.11.2.4.11 Unicode Escapes	65
8.3 ACAP-X Security Specifics	65
8.3.1 Cookie Access	65
8.3.2 Inter-Environment Bridge Access	66
8.3.3 Runtime Code Extension Access	66
8.4 ACAP-X Transport Specifics	66
8.4.1 ACAP-X Transport Binding	66
8.4.1.1 Bounded Resource Encapsulation	66

8.4.1.2 Unbounded Resource Encapsulation	66
8.4.1.3 Trigger Encapsulation	67
9. MONITOR APPLICATION SUPPORT (INFORMATIVE)	67
10. TRANSPORT AND SIGNALING	68
10.1 Introduction	68
10.1.1 Notation	68
10.2 Carousel	68
10.2.1 NSAP Address	68
10.2.2 Content Type and Timestamp Inheritance	69
10.2.3 Application Transport Over HTTP	69
10.2.3.1 HTTP Profile	69
10.2.3.2 HTTPS Profile	71
10.2.4 Time Stamp Descriptor	71
10.2.5 Usage of Private Data for non-ACAP Extensions	72
10.2.6 Data Broadcast Descriptor	72
10.3 Application Signaling	72
10.3.1 Application Content Types	73
10.3.2 Application Protocol ID	73
10.3.3 Signaling of Profiles and Versions Required by Applications	74
10.3.4 ACAP-X Extensions	74
10.3.4.1 ACAP-X Application Descriptor	74
10.3.4.2 ACAP-X Application Location Descriptor	74
10.3.4.3 ACAP-X Application Boundary Descriptor	75
10.4 Object Carousel Protocol (Informative)	76
10.4.1 Message Template	76
10.4.1.1 Interoperable Object Protocol	76
10.4.1.2 Interoperable Object References	76
10.4.1.2.1 Network Service Access Point Address	76
10.4.2 Service Gateway Message	78
10.4.2.1 Message Schema	79
10.4.2.2 Message Descriptors	79
10.4.2.2.1 Label Descriptor	79
10.4.2.2.2 Time Stamp Descriptor	79
10.4.3 Directory Message	79
10.4.3.1 Message Schema	79
10.4.4 Message Descriptors	80
10.4.4.0.1 Label Descriptor	80
10.4.4.0.2 Time Stamp Descriptor	80
10.4.5 File Message	80
10.4.5.1 Message Schema	80
10.4.5.2 Message Descriptors	80
10.4.5.2.1 Content Type Descriptor	80
10.4.5.2.1.1 Descriptor Schema	80
10.4.5.2.1.2 Descriptor Semantics	81
10.4.5.2.2 Time Stamp Descriptor	82
10.4.5.2.2.1 Descriptor Semantics	83

10.4.6 Stream Message	84
10.4.7 Stream Event Message	84
10.4.7.1 Stream Event Concepts	84
10.4.7.2 Message Schema	84
10.4.7.3 Message Semantics	85
10.4.7.4 Message Descriptors	85
10.4.7.4.1 Stream Event Descriptor	85
10.4.7.4.2 NPT Reference Descriptor	85
10.5 Data Carousel Protocol (Informative)	85
10.5.1 The Message Template	85
10.5.1.1 Message Header	85
10.5.1.2 Section Format	85
10.5.2 Download Info Indication Message	86
10.5.2.1 Message Schema	86
10.5.2.2 Method Structures	86
10.5.2.3 Message Descriptors	86
10.5.2.3.1 Compressed Module Descriptor	86
10.5.2.3.2 Label Descriptor	87
10.5.2.3.3 Caching Priority Descriptor	87
10.5.3 Download Server Initiate Message	87
10.5.3.1 Message Schema	87
10.5.3.2 Method Structures	87
10.5.3.3 Group Link Descriptor	87
10.5.3.3.1 Subgroup Association Descriptor	87
10.5.3.4 Download Data Block Message	87
10.5.3.5 Download Cancel Message	88
10.5.3.5.1 Message Schema	88
10.5.3.5.2 Message Semantics	88
10.6 Transport Protocol (Informative)	88
10.6.1 Introduction	88
10.6.2 Program Map Table	88
10.6.2.1 Deferred Association Tags Descriptor	89
10.6.2.2 Carousel Identifier Descriptor	89
10.6.2.3 Application Signaling Descriptor	89
10.6.2.4 Data Broadcast Id Descriptor	90
10.6.3 Application Information Table	90
10.6.3.1 Generic Application Descriptor Sequence	92
10.6.3.1.1 Transport Protocol Descriptor	92
10.6.3.1.1.1 Descriptor Schema	92
10.6.3.1.1.2 Object Carousel Selector Structure	93
10.6.3.2 Download Info Indication Location Descriptor	93
10.6.4 Application Specific Descriptor Sequence	93
10.6.4.1 Application Descriptor	93
10.6.4.2 Application Name Descriptor	93
10.6.4.3 Application Icon Descriptor	93
10.6.4.4 Prefetch Descriptor	94

10.6.4.5 Download Info Indication Location Descriptor	94
10.6.5 Application Representation Specific Descriptor Sequences	94
10.6.5.1 ACAP-J Application Descriptors	94
10.6.5.1.1 ACAP-J Application Descriptor	94
10.6.5.1.2 ACAP-J Application Location Descriptor	94
10.6.5.2 ACAP-X Application Descriptors	95
10.6.5.2.1 ACAP-X Application Descriptor	95
10.6.5.2.2 ACAP-X Application Location Descriptor	95
10.6.5.2.3 ACAP-X Application Boundary Descriptor	96
11. INTERACTION CHANNEL	96
11.1 Interaction Channel Protocols	96
11.1.1 Network Specific Protocols	96
11.1.2 Internet Protocol	97
11.1.3 User Datagram Protocol (UDP)	97
11.1.4 Transmission Control Protocol (TCP)	97
11.1.5 Hyper-Text Transfer Protocol (HTTP)	97
11.1.6 Domain Name Service (DNS)	97
12. SECURITY	97
12.1 Introduction	97
12.2 ACAP Trust Model	97
12.2.1 General Rules	97
12.2.2 Applications Received Over a Terrestrial Interface	98
12.2.3 Applications Received Over a Cable Interface	98
12.3 Security Policy for Applications	98
12.4 ACAP Extensions to GEM Security Model	99
12.4.1 ACAP Signing Framework	99
12.4.1.1 General Principles	99
12.4.1.2 Authentication of ACAP-X Applications	99
12.4.2 ACAP Extensions to Security Policies for Applications	100
12.4.2.1 ACAP Permission Request File	100
12.4.2.1.1 General Principles	100
12.4.2.1.2 DTD definition	100
12.4.2.1.3 ACAP Permission Request File Name and Location	101
12.4.2.2 Cable Specific Security Access Policy	101
12.4.2.2.1 Monitor Application Features Access Policy	101
12.4.2.2.1.1 Applications not Signed by ACAP Signing Framework	101
12.4.2.2.1.2 Applications Signed by the ACAP Signing Framework	101
12.4.2.2.1.3 Privileged Monitor Application API access	101
12.4.2.3 ACAP Security Policy for Applications	102
12.4.2.3.1 Cookie Permission	102
12.4.2.3.1.1 Untrusted Applications	102
12.4.2.3.1.2 Trusted Applications	102
12.4.2.3.1.3 Permission Request Syntax	102
12.4.2.3.2 Runtime Code Extension Permission	102
12.4.2.3.2.1 Untrusted Applications	103
12.4.2.3.2.2 Trusted Applications	103

12.4.2.3.2.3 Permission Request Syntax	103
12.4.2.3.3 Inter-Environment Bridge Permission	103
12.4.2.3.3.1 Untrusted Applications	103
12.4.2.3.3.2 Trusted Applications	103
12.4.2.3.3.3 Permission Request Syntax	103
12.5 Security over the Interaction Channel	104
12.6 Platform Minima	104
12.7 ACAP Security Operational Model	104
13. GRAPHICS REFERENCE MODEL	104
14. SYSTEM INTEGRATION	104
14.1 Void	104
14.2 Resource Reference and Locators	104
14.2.1 ACAP URI Scheme	104
14.2.1.1 Scheme Definition	104
14.2.1.1.1 Additional Restrictions	106
14.2.1.2 Extended ACAP URI Scheme for ACAP-X	107
14.2.1.3 Referencing Specific Entities	108
14.2.1.3.1 Program Streams	108
14.2.1.3.2 Program Elements	108
14.2.1.3.3 Files and Directories	108
14.2.1.3.4 Resolution of Locator Elements	108
14.2.1.3.4.1 Contextual	109
14.2.1.3.4.2 Universally Resolvable	109
14.2.1.3.4.3 Environment Specific	110
14.2.1.3.4.4 Physical Constructs	111
14.3 Persistent Local Storage	111
14.4 SCTE 90-1 Exceptions	111
15. MINIMUM RECEIVER REQUIREMENTS	112
15.1 General	112
15.2 User Input	112
15.3 Graphics	112
16. DETAILED PLATFORM PROFILE DEFINITIONS	112
17. CONFORMANCE	113
17.1 Compliance with GEM	113
17.1.1 GEM Errata	114
17.1.2 Modifications to MHP Definitions of Functional Equivalents	114
17.1.2.1 Application Icons Descriptor	114
Annex A: Content Identification API	115
A1. PACKAGE ORG.ATSC.SI	115
A1.1 Description	115
Annex B: Document Type Definitions (Normative)	121
B1. SCOPE	121
B2. ACAP PERMISSION REQUEST FILE DOCUMENT TYPE DEFINITION	121

B2.1 acap-permission-1.dtd	121
B3. ACAP-J FONT INDEX FILE DOCUMENT TYPE DEFINITION	124
B3.1 acap-j-font-index-1.dtd	124
B4. ACAP-X APPLICATION METADATA DOCUMENT TYPE DEFINITION	125
B4.1 acap-x-metadata-1.dtd	125
B5. ACAP-X MARKUP DOCUMENT TYPE DEFINITION	129
B5.1 acap-x-xdml-1.dtd	129
B5.2 acap-x-xdml-model-1.ent	134

Index of Tables and Figures

Table 7.1 ACAP-J Content Types	25
Table 8.1 ACAP-X Content Types	32
Table 8.2 Markup Resource Content Type References	37
Table 8.3 Stylesheet Resource Content Type References	44
Table 8.4 ECMAScript Internal Properties for Java Entities	63
Table 8.5 ACAP-X Trigger Event Transport Binding	67
Table 10.1 Specifier and Service Location	68
Table 10.2 Semantics of the HTTPProfileBody	70
Table 10.3 Semantics of the HTTPSProfileBody	71
Table 10.4 application_type Extensions	73
Table 10.5 protocol_id Extension	73
Table 10.6 ACAP-X Application Descriptor	74
Table 10.7 ACAP-X Application Location Descriptor	75
Table 10.8 ACAP-X Application Boundary Descriptor	76
Table 10.9 Network Service Access Point Address	77
Table 10.10 Network Service Access Point Address Fields	77
Table 10.11 Specifier Type Assignments	77
Table 10.12 Organization Unique Identifier Assignments	78
Table 10.13 ACAP Carousel Location	78
Table 10.14 Program Map Table	89
Table 10.15 Application Content Types	90
Table 10.16 Application Information Table	90
Table 10.17 Application Type Assignments	92
Table 10.18 Protocol Id Assignments	93
Table 10.19 ACAP-J Application Descriptor	94
Table 10.20 ACAP-J Application Location Descriptor	95
Table 12.1 Application Name for Different Application Types	101
Table 14.1 ACAP URI Contextual Constructs	109
Table 14.2 ACAP URI Universally Resolvable Constructs	109
Table 14.3 ACAP URI Environment Specific Constructs	110
Table 14.4 ACAP URI Physical Layer Constructs	111
Table 16.1 Detailed Platform Profile Definitions	113
Figure 5.1 ACAP-J System Architecture.	22
Figure 5.2 ACAP Application and System Software.	23
Figure 10.1 Content Type Inheritance.	82
Figure 10.2 Content Type Inheritance conflict.	83
Figure 11.1 Interaction channel network protocols.	96

ATSC Standard:

Advanced Common Application Platform (ACAP)

1. SCOPE

1.1 Purpose

This document defines the Advanced Common Application Platform, henceforth referred to as ACAP. ACAP is applicable for specifications and standards based on the ACAP APIs, content formats, and semantic guarantees.

The reader's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

This standard is firstly intended to be used by entities writing terminal specifications and/or standards based on ACAP. Secondly, it is intended for developers of applications that use the ACAP functionality and APIs. ACAP aims to ensure interoperability between ACAP applications and different implementations of platforms supporting ACAP applications.

Note: This standard defines the interfaces visible to applications. Application developers should not assume that any related interface is available unless it is specifically listed. Terminal standards or implementations may have other interfaces present.

An ACAP Application is a collection of information which is processed by an application environment in order to interact with an end-user or otherwise alter the state of the application environment.

ACAP Applications are classified into two categories depending upon whether the initial application content processed is of a procedural or a declarative nature. These categories of applications are referred to as procedural (ACAP-J) and declarative (ACAP-X) applications, respectively. An example of an ACAP-J application is a Java TV™ Xlet composed of compiled Java™ byte code in conjunction with other multimedia content such as graphics, video, and audio. An example of an ACAP-X application is a multimedia document composed of XHTML markup, style rules, scripts, and embedded graphics, video, and audio.

Note: An ACAP application need not be purely procedural or declarative. In particular, an ACAP-J application may reference declarative content such as graphic content or may construct and cause the presentation of markup content. Similarly, ACAP-X applications often make use of script content, which is procedural in nature. Furthermore, an ACAP-X application may reference an embedded Java TV Xlet.

Application environments are similarly classified into two categories depending upon whether they process procedural or declarative applications. These categories are referred to as ACAP-J

and ACAP-X environments, respectively. An example of an ACAP-J environment is a Java Virtual Machine and its associated Application Programming Interface (API) implementation. An example of an ACAP-X environment is an XHTML multimedia document browser, also known as a user agent.

1.2 Application

The architecture and facilities of the ACAP Standard are intended to apply to broadcast systems and receivers for terrestrial (over-the-air) broadcast and cable TV systems. In addition, the same architecture and facilities may be applied to other transport systems (such as satellite).

2. GENERAL CONSIDERATIONS

2.1 Format

ACAP is primarily based on GEM [1] and DASE [5], and includes additional functionality from OCAP [4]. GEM provides a framework for the definition of a GEM Terminal Specification. This document builds on GEM by adding specific elements in order to offer a higher degree of interoperability among different environments based on digital TV standards from the ATSC and the Society of Cable Telecommunications Engineers (SCTE).

2.2 Inclusion of GEM

This document includes GEM [1] in its entirety, except as explicitly modified by this standard.

2.3 Inclusion of OCAP

This document includes OCAP [4] in its entirety, except as explicitly modified by this standard for ACAP terminals operating in a terrestrial broadcast environment. To be fully compliant with this standard, equipment that is capable of operating in a cable environment shall also be fully compliant with OCAP [4].

2.4 Addition of Non-ACAP Interfaces

Terminal specifications based on ACAP may add public interfaces, provided that they are added in a namespace that does not conflict with ACAP. ACAP terminal specifications and ACAP terminals shall not require that such extension interfaces be called by ACAP applications in order to enable behavior that is normatively required by this standard.

2.5 Application Areas

This standard supports the same application areas as GEM [1] Section 0.5.

2.6 Profiles

The informative text referenced from GEM [1] Section 0.6 describes the GEM approach to profiles. The profiles defined in this standard are modeled on a similar scheme. This standard defines two profiles, an ACAP-J Profile and a combined ACAP-J and ACAP-X Profile. They are detailed in Section 16, "Detailed Platform Profile Definitions."

3. DEFINITIONS AND ABBREVIATIONS

3.1 Definitions from GEM

The definitions from GEM [1] Chapter 3 apply to this standard.

3.2 Definitions Introduced by OCAP

The definition of CableCARD™ from OCAP [4] applies to this standard. For this standard, the terms CableCARD™ and POD shall be considered to be interchangeable and equivalent.

3.3 Definitions Introduced by ACAP

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

ACAP Application – An application that is written only to the interfaces and semantic guarantees defined in ACAP. A suitably signaled ACAP application will run on any terminal that complies to an ACAP terminal specification.

ACAP Terminal – A terminal or other device that conforms to an ACAP Terminal Specification.

ACAP Terminal Specification – An ACAP terminal specification is a specification that includes all normative and selected optional elements of its underlying ACAP specification, and provides additional specifications as required.

Trusted Application – An application that is eligible to be trusted and to which is granted access to some sensitive resources.

3.4 Abbreviations from GEM

The definitions from GEM [1] Chapter 4 apply to this standard.

3.5 Abbreviations Introduced by ACAP

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

ACAP – Advanced Common Application Platform

ACAP-J – ACAP Procedural (Java)

ACAP-X – ACAP Declarative (XHTML)

3.6 Conformance Keywords

As used in this document, the conformance keyword *shall* denotes a mandatory provision of the standard. The keyword *should* denotes a provision that is recommended but not mandatory. The keyword *may* denotes a feature whose presence does not preclude compliance, that may or may not be present at the option of the application or the system implementer.

If the term *is* appears in a definition in a normative section of this standard, then it is to be construed as a normative definition; i.e., it has mandatory force for interpreting compliance with the definition.

3.6.1 Section and Data Structure Syntax Notation

This document contains symbolic references to syntactic elements. These references are typographically distinguished by the use of a different font (e.g., *restricted*), may contain the

underscore character (e.g., `sequence_end_code`) and may consist of character strings that are not English words (e.g., `dynrng`).

The formats of sections and data structures in this document are described using a C-like notational method such as that employed in ISO/IEC 13818-1 [22].

3.6.2 Informative Sections

There are no mandatory requirements in sections marked as informative.

4. REFERENCES

4.1 Normative References

The following standards and other references contain provisions, which through reference in this text, constitute provisions of this standard. A simple statement that the content of the reference document is a provision of this standard shall be an alternative form. At the time of publication, the editions indicated were valid. All references are subject to revision; users of this standard are therefore encouraged to investigate the possibility of applying the most recent edition of the standards and other references listed below. References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions shall not apply.
- For a non-specific reference, the latest version shall apply.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

The following comments apply to particular sources of documents:

- 1) Where the reference is to an ISO specification, it is considered to be a "non-specific" reference; additionally, officially published amendments and corrigenda are considered to automatically update the referenced document.
- 2) Where an ISBN number is provided for a referenced document, it is considered to be "specific reference".
- 3) References to RFCs are considered to be "specific references." An RFC being indicated obsoleted by another RFC is not considered significant.
- 4) ETSI specifications are available from the ETSI server at: <http://www.etsi.org>. However, the ETSI server provides the current edition of the specification and in every case this standard makes "specific" references which in the future may not be the current reference.
- 5) SCTE 90-1 is available from <http://www.scte.org>. (Informative note: The most recent version of the CableLabs OCAP 1.0 specification is available from <http://www.opencable.com/specifications>).

Note: The extent to which all or part of the following references are normative is specified at the locations in the main body of this standard where they are used. Listing a reference here does not imply that all of a reference is required.

Reference	Edition	Description	Notes
[1] GEM	1.0.2	Digital Video Broadcasting (DVB), Globally Executable MHP version 1.0.2, available as ETSI TS 102 819 V 1.3.1	4
[2] MHP 1.0	1.0.3	Digital Video Broadcasting (DVB), Multimedia Home Platform version 1.0.3, available as ETSI TS 101 812 V 1.3.1	4
[3] MHP 1.1	1.1.1	Digital Video Broadcasting (DVB), Multimedia Home Platform version 1.1.1, available as ETSI TS 102 812 V1.2.1	4
[4] OCAP 1.0	90-1 2004	ANSI/SCTE 90-1 2004, SCTE Application Platform Standard, OCAP 1.0 Profile	5
[5] A/100-1	1.0	DASE-1 Part 1: Introduction, Architecture, and Common Facilities, A/100-1, ATSC	
[6] A/52		Digital Audio Compression (AC-3) Standard, Rev. B," 14 June 2005	
[7] A/53		ATSC Digital Television Standard, Revision D, with Amendment 1, 19 July 2005	
[8] ANSI/SCTE 43	2004	Digital Video System Characteristics Standard for Cable Television	
[9] ISO 15706		Information and documentation - International Standard Audiovisual Number (ISAN)	
[10] ISO 20925-1	(work in progress)	Information and documentation - Identifier for versions of audiovisual works (V-ISAN) - Part 1: Format and use.	
[11] ISO/IEC 13818-1:2000/PDAM 4	(work in progress)	Generic Coding of Moving Pictures and Audio: Systems Amendment 4: ISAN and V-ISAN use in the content labeling descriptor	
[12] OP-SC		OC-SP-SEC-I05-040831 OpenCable Security Specification	
[13] Void		Void	
[14] Void		Void	
[15] Void		Void	
[16] Void		Void	
[17] EN 301 192	1.3.1	Digital Video Broadcast Specification for Data Broadcasting	
[18] ETR 162	Edition 1	Digital Video Broadcasting (DVB);Allocation of Service Information (SI) codes for DVB systems	
[19] ETS 300 468	Edition 2	Digital Video Broadcasting (DVB);Specification for Service Information (SI) in DVB systems	
[20] ISO 639.2	1.0	Code for the Representation of Names of Languages: Part 1	
[21] ISO 8859-1	1.0	Information Technology: 8-Bit Single-Byte Coded Graphic Character Sets: Part 1: Latin Alphabet No. 1	
[22] ISO 13818-1	Second Edition (2000)	Information Technology: Generic Coding of Moving Pictures and Associated Audio Information: Systems	
[23] ISO 13818-6	First Edition (1998)	Information Technology: Generic Coding of Moving Pictures and Associated Audio Information: Extensions for Digital Storage Media Command and Control	
[24] PNG	1.0.1	Portable Network Graphics	
[25] RFC 1738		Universal Resource Locators (URL)	
[26] RFC 1950		ZLIB Compressed Data Format Specification (Version 3.3)	
[27] RFC 1951		DEFLATE Compressed Data Format Specification (Version 1.3)	
[28] RFC 2045		Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies	
[29] RFC 2396		Uniform Resource Identifiers (URI): Generic Syntax	
[30] ANSI/SCTE 40	2004	Digital Cable Network Interface Standard	
[31] ANSI/SCTE 54	2004	Digital Video Service Multiplex and Transport System Standard for Cable Television	
[32] TR 101 162	1.0	Digital Broadcasting Systems for Television, Sound, and Data Services: Allocation of Service Information (SI) Codes for Digital Video Broadcasting (DVB) Systems	
[33] CSS-BOX	Working Draft	CSS3 Module: The Box Model, W3C	

Reference	Edition	Description	Notes
[34] CSS-COLOR	Working Draft	CSS3 Module: Color, W3C	
[35] CSS-TV	Candidate Recommendation	CSS TV Profile 1.0, W3C	
[36] CSS-UI	Working Draft	CSS3 Module: Basic User Interface, W3C	
[37] A/100-8	1.0	DASE-1 Part 8: Conformance, A/100-8, ATSC	
[38] A/100-2	1.0	DASE-1 Part 2: Declarative Applications and Environments, A/100-2, ATSC	
[39] RFC 2616		Hypertext Transfer Protocol – HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999	
[40] A/96		ATSC Interaction Channel Protocols	
[41] Void		Void	
[42] Void		Void	
[43] A/100-4	1.0	ATSC DASE-1 Part 4, A/100-4, “Application Programming Interface,” 9 March 2003	
[44] A/100-3	1.0	ATSC DASE-1 Part 3, A/100-3, “Procedural Applications and Environment,” 9 March 2003	
[45] ANSI/SCTE 65	2002	Service Information Delivered Out-Of-Band for Digital Cable Television	
[46] Void		Void	
[47] A/65		ATSC A/65B, “Program and System Information Protocol for Terrestrial Broadcast and Cable, Rev. B,” 18 March 2003	
[48] CSS	Recommendation	Cascading Style Sheets, Level 2, W3C	
[49] DOM2 EVENTS	Recommendation	Document Object Model (DOM) Level 2 Events, W3C	
[50] ECMASCRIPT		ECMAScript Language Specification, 3 rd Ed., ECMA-262, ECMA	
[51] MIME-MEDIA		Multimedia Internet Mail Extensions (MIME) Part Two: Media Types, RFC2046, IETF	
[52] Void		Void	
[53] CEA-708-B		Digital Television (DTV) Closed Captioning	
[54] HTML	4.01	HTML 4.01 Specification, W3C	
[55] ISO/IEC 11172-3	1993	Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s – Part 3: Audio	
[56] ETR 154	3.0	Digital Video Broadcasting (DVB); Implementation Guidelines for the use of MPEG-2 Systems	
[57] T3-548		Technology Group Report T3-548: “ATSC Usage of the MPEG-2 Registration Descriptor,” 9 October 2001	
[58] T3-549		Technology Group report T3-549: “Collision Avoidance for Private Fields and Ranges,” 9 October 2001	

4.2 Informative References

The following standards and other references may provide valuable information to the reader, but are not required when complying with this standard.

The following informative references apply to this standard.

Reference	Edition	Description	Note
[59] A/95		Transport Stream File System, 25 February 2003	
[60] OC-SP-HOST2.0-CFR-I06-050708		OpenCable Host Device 2.0 Core Functional Requirements	
[61] T3-575		ATSC Code Point Registry	
[62] GEM Annex C	1.0.2	Annex C of Digital Video Broadcasting (DVB), Globally Executable MHP version 1.0.2, available as ETSI TS 102 819 V 1.3.1	4

4.3 Reference Acquisition

4.3.1 ATSC Standards

Advanced Television Systems Committee (ATSC), 1750 K Street N.W., Suite 1200 Washington, DC 20006 USA; Phone: +1 202 872 9160; Fax: +1 202 872 9161; <http://www.atsc.org/>.

4.3.2 CableLabs Specifications

Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027-9750, USA; Phone: +1 303 661 9100; Fax: +1 303 661 9199; <http://www.cablelabs.com>.

4.3.3 CEA Standards

Consumer Electronics Association (CEA), 2500 Wilson Boulevard, Arlington, VA 22201-3834, USA; Phone: +1 703 907 7625; Fax: +1 703 907 7693; <http://www.ce.org>.

4.3.4 DVB Standards

Digital Video Broadcast (DVB) Project, c/o European Broadcasting Union (EBU), 17a Ancienne Route, CH-1218 Grand Saconnex, Geneva, Switzerland; Phone: +41 22 717 27 14; Fax: +41 22 717 27 27; <http://www.dvb.org>.

4.3.5 ECMA Standards

ECMA, 114, rue du Rhône, CH-1204 Geneva, Switzerland; Phone: +41 22 849 60 00; Fax: +41 22 849 60 01; <http://www.ecma.ch/>.

4.3.6 ETSI Standards

ETSI Secretariat, 650, route des Lucioles, 06921 Sophia-Antipolis Cedex, France; Phone: +33 (0)4 92 94 42 00; Fax: +33 (0)4 93 65 47 16; <http://www.etsi.org/>.

4.3.7 IETF Standards

Internet Engineering Task Force (IETF), c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 20191-5434, USA ; Phone : +1 703 620 8990; Fax +1 703 758 5913; <http://www.ietf.org>.

4.3.8 ISO Standards

International Organization for Standardization (ISO), 1 rue de Varembe, Case postale 56, CH-1211 Geneva 20, Switzerland; Phone: +41 22 749 01 11; Fax: +41 22 733 34 30; <http://www.iso.org>.

4.3.9 SCTE Standards

Society of Cable Telecommunications Engineers (SCTE), 140 Philips Road, Exton, PA 19341, USA; Phone: +1 610 363 6888; Fax: +1 610 363 5898; <http://www.scte.org/>.

4.3.10 W3C Standards

World Wide Web Consortium (W3C), Massachusetts Institute of Technology, Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139, USA; Phone: +1 617 253 2613; Fax: +1 617 258 5999; <http://www.w3.org/>.

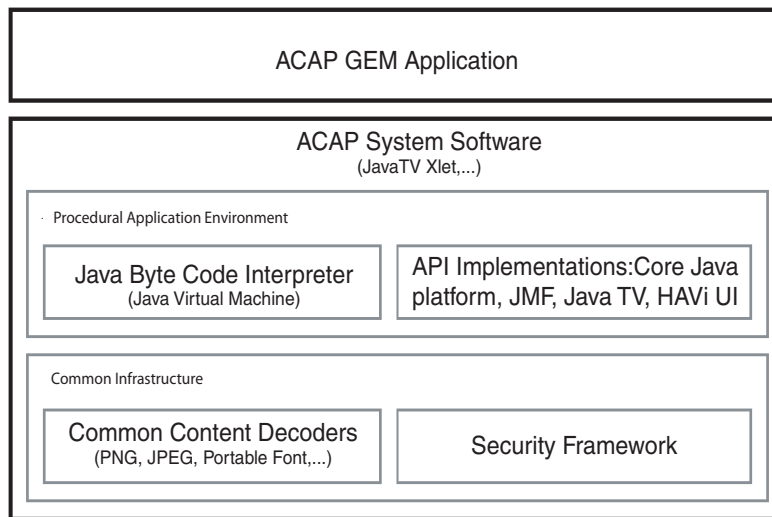


Figure 5.1 ACAP-J System Architecture.

5. ARCHITECTURE

The architecture for ACAP is as specified in the MHP definition of the functional equivalent named “Arch” as specified in GEM [1] Clause 15.6. (See Section 17.1, “Compliance with GEM.”)

5.1 Support for ACAP-J Applications

Where only ACAP-J applications are supported, the conceptual arrangement of the application and system software are as shown in Figure 5.1.

5.2 Support of ACAP-X Applications

Where the optional ACAP-X applications are supported, the system application and systems software conceptual arrangement is as illustrated in Figure 5.2.

6. COMMON CONTENT FORMATS

6.1 General

Chapter 7 of GEM [1] shall apply.

In this standard, support for PNG shall have an additional requirement beyond those requirements inherited from GEM, specifically that the tRNS chunk shall be supported for images where it is present and when it specifies one of the color types defined by PNG to be allowed to contain this chunk.

Note: The extent to which an ACAP terminal device can reproduce transparent colors is subject to those approximations defined by MHP [2], Section 15.1 “PNG Restrictions”. The above language does not modify these restrictions or impose additional requirements on the graphics hardware of an ACAP terminal device.

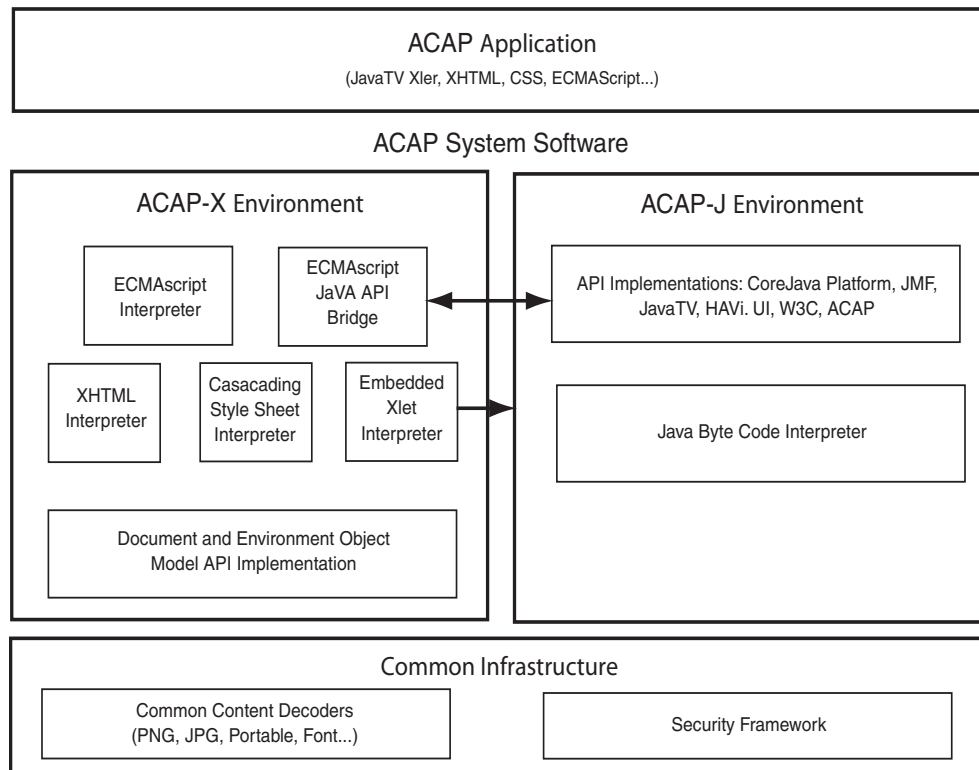


Figure 5.2 ACAP Application and System Software.

6.2 Static Formats

MPEG-1 Audio Layer 3 elementary stream data shall be supported as defined by ISO/IEC 11172-3 [55], as further constrained by ETR 154 [56].

6.3 Broadcast Streaming Formats

6.3.1 Video

Video streamed over a terrestrial network shall be as defined by ATSC A/53D [7]. Video streamed over a cable network shall be as defined by SCTE 43 [8].

6.3.2 Audio

Audio streamed over either terrestrial or cable networks shall be Dolby AC-3 data as defined by ATSC A/52A [6] and ATSC A/53D [7].

6.3.3 Closed-Captioning

The terminal device shall extract NTSC closed captioning information when present in the MPEG-2 Picture Level user_data as specified in Section 4 of CEA-708-B [53] or as specified in CEA-608-B and transported according to SCTE 21 or SCTE 20. This will include all data of cc_type 00 and 01, as defined in Part 15 of 47 CFR and CEA-608-B. The Terminal device shall reconstruct line 21 VBI (both field 1 and field 2) according to CEA-608-B on all NTSC analog video outputs. In the case where a MPEG Picture Level user_data field includes data formatted and

transported according to SCTE 21 or SCTE 20, the terminal device may use closed captioning data recovered from either method.

Note: Other closed captioning and extended data structures may be present in the MPEG-2 Picture Level `user_data`.

If the terminal device provides analog component or uncompressed digital video outputs, render and display of the NTSC caption data shall be provided according to Part 15 of 47 CFR.

The terminal device shall extract the Digital Television closed captioning (DTVCC) information when present in the MPEG-2 Picture Level `user_data`, as specified in Section 9 of CEA-708-B [53] and delivered according to ATSC A/53D [7] (with `cc_type` set to '10' or '11'). This caption data shall be passed through to a DTV display via the IEEE-1394 interface.

If the terminal device provides an NTSC analog video output, and the network stream dual carries CEA-608-B caption data transported according to SCTE 21 or SCTE 20, then the terminal device shall reconstruct line 21 VBI in the NTSC analog video output as required by Part 15 of 47 CFR.

The terminal device shall process the `caption_service_descriptor`, when present, as defined in ATSC A/65B [47] and carried in either the PMT or EIT of the in-band MPEG-2 transport stream.

The closed captioning representation shall be drawn over any graphics of ACAP applications, which shall not hide the captioning text at any time. The closed captioning layer is logically in the `HGraphicsDevice`. Actual implementation is implementation-dependent.

In all cases, reconstruction of line-21 for analog NTSC outputs and pass-through of the content advisory information in a compressed SPTS on the IEEE-1394 interface is required.

Note: Products implementing this standard may need to comply with a number of other specifications or regulations for the support of closed captioning which are outside the scope of this standard.

7. ACAP-J APPLICATIONS AND ENVIRONMENT

This section defines the content of ACAP-J applications and the behavior and facilities required or permitted by an ACAP-J environment.

The definition of an ACAP-J application and environment is based on a combination of GEM [1], OCAP [4], and DASE [5]. When normative material is incorporated into this standard from GEM [1] and OCAP [4], the terms GEM Application and OCAP Application should be read in this standard as ACAP-J application; and OCAP Execution Engine should be read as an ACAP-J environment.

The content (MIME media) type used to label an ACAP-J application as an aggregate entity shall be `application/acap-j`.

7.1 Behavior

7.1.1 Application Model

Chapter 9 of GEM [1] shall apply with the DVB-J model applying to ACAP-J applications.

7.1.2 Destruction of Applications

ACAP terminals shall implement the facility described in Section 13.2.1.8.2 of OCAP 1.0 [4] for destruction of applications.

7.2 Facilities

The content (MIME media) types specified in Table 7.1 may be used by an ACAP-J application and shall be supported by an implementation of an ACAP-J environment. In this table, the last column specifies zero or more file name extensions that should be used with files of this content type. The extension <x> designates a numeric value starting from zero (0) with no leading zeros. The extension N/A indicates that no extension is applicable since resources of this type are not named or do not appear in the broadcast file system, or the content type describes a collection of resources. The extension others indicates that other, non-specific extensions are permitted.

Table 7.1 ACAP-J Content Types

Content Type	See Section	Extensions
application/acap-certificate	12.4.1	.<x>
application/acap-digest	12.4.1	.hashfile
application/acap-j	7	N/A
application/acap-j-fontindex	7.2.2	.fontindex
application/acap-permission	6	.perm
application/acap-signature	12.4.1	.<x>
application/font-tdpfr	6	.pfr
application/java	7.2.1	.class
application/zip	7.2.3	.zip
audio/ac3	6.3.2	N/A
audio/mpeg	6	.mp2
image/jpeg	6	.jpg; jpeg
image/mpeg	6	.mpg
image/png	6	.png
text/dvb.utf8	6	.txt; others
video/dvb.mpeg.drip	6	.drip
video/mpeg	6.3.1	N/A
video/mpv	6.3.1	N/A

An ACAP-J application shall contain at least one resource of content type `application/java`. The presence of resources of other content types in an ACAP-J application is strictly optional.

7.2.1 Java Content

7.2.1.1 Additional Java APIs

7.2.1.1.1 Closed Captioning

The `org.ocap.media` package as defined in Annex S of OCAP 1.0 [4].

7.2.1.1.2 Locators

The `org.ocap.net.OcapLocator` class as defined in Annex I of OCAP 1.0 [4]. In ACAP terminals, `OcapLocator` shall support the syntax defined in Section 14.2.1.

7.2.1.1.3 Events

The `org.ocap.ui.event` package as defined in Annex E of OCAP 1.0 [4].

7.2.1.1.4 Content Identification API

An object which implements the `javax.tv.service.guide.ProgramEvent` interface shall also implement the `org.atsc.si.ContentIdentifications` interface. An array of objects which implement `org.atsc.si.ContentIdentification` or the appropriate subinterface of `ContentIdentification` shall be returned by the method `getIdentifiers()` from the `ContentIdentifications` interface. In the case where the underlying program event does not contain content identifiers, the `getIdentifiers()` method shall return an empty array. An object which implements the `org.atsc.si.ISANIdentification` interface shall be a member of the array of `ContentIdentification` returned by the `getIdentifiers()` method when the underlying program event is identified with an ISAN identifier. An object which implements `VISANIdentification` shall be a member of the array of `ContentIdentification` returned by the `getIdentifiers()` method when the underlying program event is identified with a V-ISAN identifier. The format of the string returned by the `getISANIdentifier` method of `ISANIdentification` shall be conformant with ISO 15706 [9]. The format of the string returned by the `getVISANIdentifier` method of `VISANIdentification` shall be conformant with ISO 20925-1 [10].

Note: Content identification values defined by ISO 15706 [9] and ISO 20925-1 [10] are carried by means of ISO/IEC 13818-1:2000/PDAM 4 [11].

The Content Identification API is defined in Annex A, “Content Identification API.”

7.2.1.1.5 Extended SI API

The `org.ocap.si` package as defined in Annex T, Section T.3 of OCAP 1.0 [4].

When the POD device is absent and the host device is connected to and operating on a cable network (i.e., the primary display is derived from a cable input source), any unbound applications that run, shall not have access to service information provided per ANSI/SCTE 65 2002. In all other cases, applications shall have access to service information as specified in Annex T, section T.3 of OCAP 1.0 [4].

7.2.1.2 Inter-Environment DOM Integration

If an ACAP System supports an ACAP-X environment, then it shall support the following additional packages, as further restricted below, in the ACAP-J environment:

- `org.atsc.dom`
- `org.atsc.dom.environment`
- `org.atsc.dom.events`
- `org.atsc.dom.html`
- `org.atsc.dom.views`
- `org.w3c.dom`
- `org.w3c.dom.css`
- `org.w3c.dom.events`
- `org.w3c.dom.html2`
- `org.w3c.dom.views`

Support for `org.atsc.dom` and its sub-packages as listed above shall adhere to A/100-4 [43], Sections 4.3 through 4.7, and shall be governed by the semantics defined by A/100-2 [38], Section 5.3.1.2, as further constrained by Section 8.2.11.

Support for `org.w3c.dom` and its sub-packages as listed above shall adhere to A/100-3 [44], Section 5.1.1.2.6, except that support for `org.w3c.dom.css` shall be limited to the following interfaces:

- `org.w3c.dom.css.CSSStyleDeclaration`
- `org.w3c.dom.css.ElementCSSInlineStyle`

Furthermore, the following methods of `org.w3c.dom.css.CSSStyleDeclaration` shall not be used by an ACAP Application and need not be implemented by an ACAP System:

- `getPropertyCSSValue(String)`
- `getParentRule()`

If one of these methods is implemented by an ACAP System and invoked by an ACAP Application, then a runtime exception shall be raised.

7.2.1.3 Java Class Files

An ACAP-J application shall use and an ACAP-J environment shall support Java Class Files as defined by MHP [2], Section 11, and as required by GEM [1]. The content (MIME media) type used to label Java Class Files shall be `application/java`.

7.2.1.4 Integration of the JavaTV SI API

As defined in Annex T, Section T.2.1.2.1 of OCAP 1.0 [4].

7.2.1.5 Addition of Non-ACAP Interfaces

As discussed in Section 2.4, “Addition of Non-ACAP Interfaces,” terminal specifications based on ACAP may add extensions to ACAP, provided that they are added in a namespace that does not conflict with ACAP. In the case of the ACAP-J environment, any such extensions must be done in a Java package that does not conflict with one specified by ACAP.

7.2.1.6 Void

7.2.1.7 Semantics of `java.io.File.lastModified()` for Broadcast Carousels

GEM [1] clause 11.5.1 allows GEM terminal specifications to define signalling that provides a value for the method `java.io.File.lastModified()`. Section 10.4.2, “Time Stamp Descriptor,” defines a time stamp descriptor. If a time stamp descriptor is available for a file, the value in the time stamp descriptor shall be reported from `java.io.File.lastModified()`. If one is not available, then the value returned from this method is undefined.

7.2.2 Font Index Content

An ACAP-J application may use and an ACAP-J environment shall support Font Index Files as defined by MHP [2], Annex D.2.2.2, and required by GEM [1].

The content (MIME media) type used to label Font Index Files shall be `application/acap-j-fontindex`.

7.2.3 Archive Content

An ACAP-J application may use and an ACAP-J environment shall support ZIP Archive Files as defined by MHP [2], Section 11.3.1.4, and as required by GEM [1].

The content (MIME media) type used to label a ZIP Archive File shall be `application/zip`.

8. ACAP-X APPLICATIONS AND ENVIRONMENT

This section defines the content of ACAP-X applications and the behavior and facilities required or permitted by an ACAP-X environment.

The definition of an ACAP-X application and environment is based on DASE Declarative Applications and Environment as defined by A/100-2 [38] and other parts of ATSC Standard A/100. When normative material is incorporated into this standard from the ATSC Standard A/100, the terms *DASE Declarative Application* should be read in this standard as *ACAP-X application*; *DASE Declarative Environment* should be read as *ACAP-X environment*; and *DASE System* should be read as a combination of both *ACAP-J* and *ACAP-X environments*.

The content (MIME media) type used to label an ACAP-X application as an aggregate entity shall be application/acap-x.

8.1 Behavior

This section describes certain behavioral aspects of ACAP-X applications and implementations of an ACAP-X environment.

8.1.1 Application Behavior

This section specifies restrictions and extensions on use and support for resource identifier schemes with respect to A/100-1 [5] and A/100-2 [38].

The description of application processing, decoding, and presentation specified in A/100-2 [38], Sections 4.1 through 4.3, shall apply to ACAP-X Applications and implementations of an ACAP-X environment with the following exceptions:

- The *Application Lifecycle*, including the application state model, shall adhere to GEM [1], Section 9.3, “DVB-HTML Model,” with the term *ACAP-X* substituted for the term *DVB-HTML*.

Note: The application lifecycle of a DASE Declarative Application is effectively replaced by DVB-HTML application lifecycle.

- Transitions in the lifecycle state model of an ACAP-X Application shall generate application lifecycle events as defined by Section 8.2.11.2.2.2, “Application Lifecycle Event Types.”
- The *Application Display Model* shall adhere to GEM [1] Section 13, “Graphics Reference Model.”
- Xlets embedded in ACAP-X applications shall be supported as defined in MHP 1.1 [3], Section 9.6.

8.1.1.1 Clarifications

Regarding the interpretation of GEM [1], Section 9.3, “DVB-HTML Model,” an ACAP-X application that is initially signaled for prefetching rather than auto-start shall remain in the `LOADING` state until an `org.acap.trigger.start` environment trigger is received, at which point the application shall be transitioned to the `ACTIVE` state. See Section 8.1.4.2.2, “`org.atsc.trigger.start` Trigger Types,” for further information.

8.1.2 Resource Identifier Schemes

This section specifies restrictions and extensions on use and support for resource identifier schemes with respect to A/100-1 [5] and A/100-2 [38].

An ACAP-X application may use and an ACAP-X environment shall support the archive scheme defined by A/100-1 [5], Section 5.1.2.3.1.1, and the *acap* and *exit* schemes as described below.

8.1.2.1 Restrictions

8.1.2.1.1 *ecmascript* Scheme

The *ecmascript* scheme as specified by A/100-1 [5], Section 5.1.2.3.1.2, shall not be used by an ACAP-X application and need not be supported by an ACAP-X environment.

8.1.2.1.2 *lid* Scheme

The *lid* scheme as specified by A/100-1 [5], Section 5.1.2.3.1.3, shall not be used by an ACAP-X application and need not be supported by an ACAP-X environment.

8.1.2.1.3 *tv* Scheme

The *tv* scheme as specified by A/100-1 [5], Section 5.1.2.3.1.4, shall not be used by an ACAP-X application and need not be supported by an ACAP-X environment.

8.1.2.2 Extensions

8.1.2.2.1 *acap* Scheme

An ACAP-X application may use and an ACAP-X environment shall support the *acap* scheme defined by Section 14.2.1.2, “Extended ACAP URI Scheme for ACAP-X.”

8.1.2.2.2 *exit* Scheme

An ACAP-X application may use and an ACAP-X environment shall support the *exit* scheme, the format of which consists of a scheme component and does not include a scheme specific component. The semantics of activating a URI that uses this scheme shall cause the ACAP-X application to be terminated.

Note: The *exit* scheme does not identify a resource per se, but denotes a semantic action.

8.1.3 Event Processing

This section specifies restrictions on use and support for event processing behavior with respect to A/100-1 [5] and A/100-2 [38].

8.1.3.1 Restrictions

All DOM events that are emitted by an ACAP-X environment shall first be dispatched to the `EventTarget::dispatchEvent` method of the `Window` object of the affected document, and, thence, to the `HTMLDocument` object (i.e., the document node); that is, the capturing phase of downward event propagation shall begin with the `Window` object and descend from there to the affected document’s document node and thence to its descendent element nodes.

Note: See Section 8.2.11, “Script Content,” and A/100-2 [38], Section 5.3, for further information about the `Window` and `HTMLDocument` objects.

After an unload HTML event is dispatched to a `Window` object, further HTML events shall not be dispatched to the `Window` object until a new document is loaded into the `Window`.

If a new document is to be loaded into a `Window` object, then that document shall not be presented and no event shall be dispatched to it until an unload HTML event is dispatched to the `Window` to signal the unloading of the `Window`'s current document.

When dispatching an event to a `Window` object, the `Window::document` property shall reference the `HTMLDocument` object associated with the `Window`.

8.1.4 Trigger Processing

This section specifies restrictions and extensions on use and support for trigger processing behavior with respect to A/100-1 [5] and A/100-2 [38].

Note: An ACAP application indicates its interest in receiving trigger events by registering DOM Event Listeners for specific trigger event types. See Section 8.2.11.2.2.4, “Trigger Event Types,” for further information.

8.1.4.1 Restrictions

The *script* event type defined by A/100-2 [38], Section 4.5, shall not be used by an ACAP-X application, and need not be supported by an ACAP-X environment. If a *script* event type is utilized by an ACAP-X application, then it shall be ignored regardless of whether or not it is supported.

8.1.4.2 Extensions

An ACAP-X application may use and an ACAP-X environment shall support both asynchronous (“do-it-now”) and synchronized trigger events, where the concrete form of these triggers is transport dependent, but shall consist of the following information items:

- event type
- event time
- event payload

The *event type* shall take the form of a string capable of representing Unicode character data that denotes the type of trigger event.

The *event time* shall be a value that denotes one of the following: 1) “now,” or 2) a media time. In the first of these cases, the trigger is considered to be asynchronous, and the associated trigger event shall be dispatched immediately upon reception; in the last of these cases, the indicated time shall be the time at which the related media stream's play time matches the event time.

The *event payload* shall take the form of a (possibly empty) string capable of representing Unicode character data, that denotes an arbitrary, application defined trigger payload.

The concrete representation and encoding of the above information items is dependent upon the ACAP-X application transport scenario in use.

Note: See Section 8.4.1.3, “Trigger Encapsulation,” for further information on concrete representation and transport of triggers.

The set of possible trigger event types is divided into two categories: 1) environment triggers, and 2) application triggers, as defined in the following sub-sections.

8.1.4.2.1 Environment Triggers

A trigger in which the event type has the prefix `org.atsc.trigger` is defined to be an *environment* trigger. An environment trigger is dispatched to the application environment itself and not dispatched to the application.

If a trigger event type does have the prefix `org.atsc.trigger`, but the event type is not defined by this standard as an environment trigger, then the trigger shall be ignored and shall not be dispatched to the application environment.

Note: Even though environment triggers are not dispatched to an application, they do, in general, have a direct or indirect affect on a targeted application.

8.1.4.2.2 `org.atsc.trigger.start` Trigger Types

The environment trigger event type `org.atsc.trigger.start` is used to cause a prefetched application to become active. When received, it shall cause the application to be transitioned from the `LOADING` to the `ACTIVE` state, which, in turn, shall cause an `org.atsc.application.started` event to be dispatched to the targeted application. If the application is not in the `LOADING` state upon trigger reception, then this environment trigger event shall be ignored.

An event payload is not defined for use with this trigger event type, and if present, shall be ignored.

8.1.4.2.3 Application Triggers

A trigger in which the event type does not have the prefix `org.atsc.trigger` is defined to be an *application* trigger. Furthermore, an application trigger shall not have the prefix `org.atsc` unless such usage is defined by this standard or a future revision thereof. An application trigger is dispatched to the application in the form of a `TriggerEvent` object as described by Section 8.2.11.2.1.3, “`TriggerEvent` Object.”

Note: As of the initial release of the ACAP specification, no ACAP defined application trigger is specified.

8.2 Facilities

This section describes the content facilities that are either required in or available for use by an ACAP-X application and that must be supported by an ACAP-X environment.

Note: A *content facility* is a logical grouping of a set of MIME media types (content types).

The content (MIME media) types specified in Table 8.1 may be used by an ACAP-X application and shall be supported by an implementation of an ACAP-X environment. In this table, the last column specifies zero or more file name extensions that should be used with files of this content type. The extension `<x>` designates a numeric value starting from zero (0) with no leading zeros. The extension `N/A` indicates that no extension is applicable since resources of this

type are not named or do not appear in the broadcast file system or the content type describes a collection of resources.

Table 8.1 ACAP-X Content Types

Content Type	See Section	Extensions
application/acap-certificate	12.4.1	.<x>
application/acap-digest	12.4.1	.hashfile
application/acap-permission	12.4.1.2	.perm
application/acap-signature	12.4.1	.<x>
application/acap-x	8	N/A
application/acap-x-metadata	8.2.1	.xml
application/font-tdpfr	8.2.7	.pfr
application/xhtml+xml	8.2.9	.xht;.xhtml
application/zip	8.2.8	.zip
audio/ac3	8.2.6	N/A
audio/basic	8.2.4	.au
audio/mpeg	8.2.4	.mp2
image/jpeg	8.2.2	.jpg;.jpeg
image/mpeg	8.2.2	.mpg
image/png	8.2.2	.png
text/css	8.2.9.2.8	.css
text/ecmascript	8.2.111	.es
video/mng	8.2.3	.mng
video/mpeg	8.2.5	N/A
video/mpv	8.2.5	N/A

An ACAP-X application shall contain at least one resource of content type `application/xhtml+xml`. The presence of resources of other content types in an ACAP-X application is strictly optional.

When an ACAP-X application makes use of an embedded Xlet or the Inter-Environment Bridge, resources that adhere to the content types specified in Table 7.1 may also be present in an ACAP-X application's resource collection.

8.2.1 Application Metadata Content

An ACAP-X application may use and an ACAP-X environment shall support the Application Metadata Content facilities defined by A/100-1 [5], Section 6.1, as modified and extended below.

In the case that application metadata content is inconsistent with metadata information specified in application signaling, then the metadata information specified in application signaling shall be given precedence.

If an ACAP-X application does not include an application metadata resource or does not reference an application metadata resource as its root resource, then the information that is mandatory in an application metadata resource shall be implied from signaling information that accompanies the ACAP-X application. In this case, the behavior of an ACAP-X application shall be identical to an ACAP-X application that does include and reference such an implied application metadata resource as its root resource.

8.2.1.1 Modifications

8.2.1.1.1 Content Type

The content (MIME media) type used to label an ACAP-X Application Metadata Resource shall be application/acap-x-metadata.

8.2.1.1.2 Document Type Definition

The document type definition used to determine the validity of ACAP-X Application Metadata Content shall be as defined by Annex B, Section 4, “ACAP-X Application Metadata Document Type.”

8.2.1.1.3 Document Type Declaration

The formal public identifier used by a document type declaration in an ACAP-X Application Metadata resource shall be as follows:

```
"-//ATSC//DTD ACAP-X Application Metadata 1.0//EN"
```

8.2.1.2 Extensions

8.2.1.2.1 *entity* Element

In addition to those values specified by A/100-1 [5], Section 6.1.1.6.9.1, the *entitytype* attribute of the *entity* element may take the following values, as further described below:

signature

In addition, the semantics of the entity types described by A/100-1 [5], Section 6.1.1.6.9.1, are further extended as described below.

8.2.1.2.2 *initial* Entity Type

If an ACAP-X application includes an application metadata resource, then it shall be specified an *entity* element with an *entitytype* attribute with the value initial. Furthermore, the value of the *uri* attribute shall reference a resource that adheres to the ACAP-X markup content type as defined by Section 8.2.9, “Markup Content.”

8.2.1.2.3 *permissionRequest* entity Type

If an ACAP-X application includes a permission request file, then it shall be specified by an *entity* element with an *entitytype* attribute with the value permissionRequest. In this case, the value of the *uri* attribute shall reference a resource that adheres to the ACAP permission request file content type as defined by Section 12.4.2.1, “ACAP Permission Request File.”

8.2.1.2.4 *signature* Entity Type

If an ACAP-X application includes a signature file, then it shall be specified by an *entity* element with an *entitytype* attribute with the value signature. In this case, the value of the *uri* attribute shall reference a resource that adheres to the ACAP signature file content type as defined by Section 12.4.1, “ACAP Signing Framework.”

8.2.1.2.5 *identifier* Element

An ACAP-X application may specify and an ACAP-X environment shall support *param* children elements of the *identifier* element such that the *name* attributes of these *param* elements are *orgid* and *appid* and the *value* attributes of these *param* elements adhere to MHP [2], Section 14.5, “Text Encoding of Application Identifiers.”

Note: See MHP [2], Section 10.5, “Application Identification,” for further information on organization and application identifiers.

If an ACAP-X application includes an application metadata resource, then it shall specify exactly one *orgid* and exactly one *appid* using the *param* children of the *ident* element as described above.

If an ACAP-X application includes an application metadata resource, then it shall specify a value for the *uuid* attribute of the *identifier* element; however, that value may be zero (0). Notwithstanding the preceding, an ACAP-X application should specify a probabilistically unique, non-zero value for the *uuid* attribute.

Example

The following specifies an identifier element that satisfies the above requirements.

```
<identifier uuid="0">
  <param name="orgid" value="0x000023d2"/>
  <param name="appid" value="0x4020"/>
</identifier>
```

If an application metadata resource is implied and the transport scenario does not signal a UUID, then a UUID of zero (0) shall be implied.

8.2.1.2.6 Permission Capability

An ACAP-X application may use and an ACAP-X environment shall support a *permission* capability on the *cond* element in order to specify requirements or requests with respect to the granting of security permissions.

The *permission* capability admits the following parameters, as specified by child *param* elements of the *cond* element specifying this capability:

- *type*
- *target*
- *actions*

Exactly one *type* parameter shall be specified as a child of the *cond* element that specifies this capability; exactly zero or one *target* and zero or one *actions* parameters may be specified.

The semantics of a *permission* capability is as follows: if an application specifies a *permission* capability with a *qualifier* attribute whose value is required, and the ACAP System cannot or would not grant the indicated permission, then the application shall not be activated.

Note: Use of the *permission* capability does not constitute a request for authorization to access the *target* or *actions* of the named permission. A request for authorization to grant a permission is outside of the scope of the semantics of the *cond* element.

Example

The following specifies a *permission* capability on a *cond* element in order to indicate that access to read and write a file in the local filesystem is required in order to activate the application.

```
<cond qualifier="required" capability="permission">
  <param name="type" value="java.io.FilePermission"/>
  <param name="target" value="file.dat"/>
  <param name="actions" value="read,write"/>
</cond>
```

8.2.1.2.6.1 *type* Parameter

The *type* parameter of a *permission* capability shall specify either a fully qualified Java class name or a non-Java permission type name.

8.2.1.2.6.2 *target* Parameter

The *target* parameter of a *permission* capability specifies a target for the specifically named permission. The syntax and semantics of the *target* parameter are governed by the named permission.

8.2.1.2.6.3 *actions* Parameter

The *actions* parameter of a *permission* capability specifies one or more actions for the specifically named permission. The syntax and semantics of the actions parameter are governed by the named permission.

8.2.2 Graphics Content

An ACAP-X application may use and an ACAP-X environment shall support the Graphics Content facilities defined by A/100-1 [5], Section 6.2, as extended below, and shall do so in a manner consistent with GEM [1] Section 7.1.1, and with Section 17.1.1, “GEM errata,” of this document.

Note: The ACAP-X environment does not support the image/gif content type.

8.2.2.1 Extensions

8.2.2.1.1 Image/MPEG Formats

An ACAP-X application may use and an ACAP-X environment shall support the MPEG-2 I-Frame graphics content format as defined by GEM [1] Section 7.1.2. Application resources that employ this format shall be identified as image/mpeg and shall use the resource name extension “.mpg”.

An application entity identified as content type image/mpeg may be referenced in all contexts where a content type defined by A/100-1 [5], Section 6.2, “Graphics Content”, is permitted. In all other contexts, the use of content type image/mpeg is undefined.

8.2.3 Non-Streaming Video Content

An ACAP-X application may use and an ACAP-X environment shall support the *Non-Streaming Video* content facilities defined by A/100-1 [5], Section 6.3, as extended below.

8.2.3.1 Extensions

8.2.3.1.1 Video “Drip Feed” Format

The ACAP-X environment does not directly support the MPEG-2 Video “drip feed” content format as defined by GEM [1], Section 7.1.3. Nevertheless, because this non-streaming video content type is supported by the ACAP-J environment, it is possible to make use of this content type by employing appropriate ACAP-J APIs through either: 1) the Inter-Environment API Bridge, or 2) an embedded Xlet. If an ACAP-X application makes indirect use of resources of this content type, then such resources shall be identified as `video/dvb.mpeg.drip` and shall use the resource name extension “.drip”.

8.2.4 Non-Streaming Audio Content

An ACAP-X application may use and an ACAP-X environment shall support the Non-Streaming Audio content facilities defined by A/100-1 [5], Section 6.4, as extended below.

8.2.4.1 Extensions

8.2.4.1.1 MPEG Audio Layers

An ACAP-X application may use and an ACAP-X environment shall support the MPEG-1 Audio Layers 1 and 2 content format as defined by MHP [2], Section 7.1.4. Application resources that employ this format shall be identified as `audio/mpeg` and shall use the resource name extension “.mp2”.

An application entity identified as content type `audio/mpeg` may be referenced in all contexts where a content type defined by A/100-1 [5], Section 6.4, “Non-Streaming Audio Content”, is permitted. In all other contexts, the use of content type `audio/mpeg` is undefined.

8.2.5 Streaming Video Content

An ACAP-X application may use and an ACAP-X environment shall support the *Streaming Video Content* facilities defined by A/100-1 [5], Section 6.5.

8.2.6 Streaming Audio Content

An ACAP-X application may use and an ACAP-X environment shall support the *Streaming Audio Content* facilities defined by A/100-1 [5], Section 6.6.

8.2.7 Font Content

An ACAP-X application may use and an ACAP-X environment shall support the *Font Content* facilities defined by A/100-1 [5], Section 6.7.

8.2.8 Archive Content

An ACAP-X application may use and an ACAP-X environment shall support the *Archive Content* facilities defined by A/100-1 [5], Section 6.8.

8.2.9 Markup Content

An ACAP-X application may use and an ACAP-X environment shall support the *Markup Content* facilities defined by A/100-2 [38], Section 5.1, as restricted and extended below.

8.2.9.1 Restrictions

This section describes restrictions upon the markup content facility defined by A/100-2 [38], Section 5.1.

8.2.9.1.1 Resource Content Type References

References to resources from markup content attributes that take the form of a URI shall be limited to those content types checked as supported in Table 8.2. If a reference is made to a resource of a content type that is not supported in the referencing context, then the behavior of an ACAP-X environment is implementation-dependent, and shall not be relied upon by an ACAP-X application. If a content type is not listed in this table, then it shall be construed as unsupported.

Table 8.2 Markup Resource Content Type References

Element/Attribute	application/acap-j	application/acap-x	application/java	application/xhtml+xml	application/zip (note 3)	audio/ac3	audio/basic	audio/mpeg	image/{jpeg,mpeg,png}	text/css	text/ecmascript	video/mng	video/mpeg	video/mpv	Notes
a.href	✓	✓	x	✓	x	✓	x	x	x	x	x	x	✓	✓	
area.href	✓	✓	x	✓	x	✓	x	x	x	x	x	x	✓	✓	
Base.href															1
frame.longdesc	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	
frame.src	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	
img.longdesc	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	
img.src	x	x	x	x	x	x	x	x	✓	x	x	✓	x	✓	
Input.src	x	x	x	x	x	x	x	x	✓	x	x	✓	x	✓	
link.href	x	x	x	✓	x	x	x	x	x	✓	x	x	x	x	
object.archive															2
object.classid	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	
object.codebase															1
object.data	x	x	x	✓	x	✓	✓	✓	✓	x	x	✓	✓	✓	
script.src	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	
style.src	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	
Notes															
1. The value of the <i>href</i> attribute of the <i>base</i> element does not reference a resource; rather, it serves as a means for resolving relative URIs that appear as attributes of other elements.															
2. No semantics are associated with the <i>archive</i> attribute of the <i>object</i> element in this version of the ACAP specification.															
3. Use of the application/zip content type is supported only indirectly through use of the <i>archive</i> URI scheme.															

8.2.9.1.2 Resource Access

References to a streaming video or audio resources from ACAP-X markup content shall not cause tuning to occur. References that imply tuning to access a resource shall behave as if the resource were unavailable.

8.2.9.1.3 Document Type Declaration

An ACAP-X application shall not include or reference and an ACAP-X environment need not support a markup content entity whose document type declaration refers to a document type that

is not an ACAP-X Family Document Type as defined by Annex B, Section 5, “ACAP-X Markup Document Type.”

If a markup content entity of an application signaled as an ACAP-X application makes reference to a formal public identifier (FPI) in its document type declaration and that identifier references a document type that is not an ACAP-X Family Document Type, then the ACAP-X application shall be aborted.

8.2.9.1.4 Namespace Declarations

A markup content document instance in an ACAP-X application should specify a default XML Namespace Declaration using the *xmlns* attribute in the root (document) element. If no default XML Namespace Declaration is specified, then <http://www.w3.org/1999/xhtml> shall be assumed to be the default namespace.

The use of any XML namespace prefix in a markup content document instance other than the *xml* prefix shall be accompanied by an appropriate XML namespace attribute.

8.2.9.1.5 legacy Application

An ACAP-X application shall not indicate that it is a legacy application by specifying a *legacy* application parameter with the value *true* in the application metadata resource. If an ACAP-X application is marked as a legacy application, then it shall not be activated by an ACAP-X environment.

Note: See A/100-1 [5], Section 6.1.1.6.13.4, for further information on the legacy application parameter.

8.2.9.1.6 intrinsic event Attributes

An ACAP-X application shall not use and an ACAP-X environment need not support the following *intrinsic event* attributes:

- onclick
- ondblclick
- onmousedown
- onmouseup
- onmouseover
- onmousemove
- onmouseout
- onkeypress
- onkeydown
- onkeyup
- onfocus
- onblur
- onsubmit
- onreset
- onselect
- onchange
- onload
- onunload

Note: The functionality of these intrinsic event attributes may be obtained by use of DOM-2 Events functionality as specified by A/100-2 [38], Section 5.3.1.2.7.

If an ACAP-X application makes use of one of these attributes, then an ACAP-X environment shall not evaluate its script content, and may optionally present an error indication to the end-user.

8.2.9.1.7 *name* Attribute

Except for those element types that require the presence of a *name* attribute, the *name* attribute defined by A/100-2 [38], Section 5.1.1.5.2, for use with transcoded legacy documents shall not be used by an ACAP-X application and need not support the semantics for this attribute in this context in an ACAP-X environment.

8.2.9.1.8 *a* (anchor) Element

The ability for an *a* (anchor) element to target the top-level frame with video content defined by A/100-2 [38], Section 5.1.1.6.1.1.3, for use with transcoded legacy documents in order to terminate a declarative application, shall not be used by an ACAP-X application.

If an ACAP-X application does target the top-level frame with video content, then activating the anchor shall be ignored, producing no side effect.

The ability for an *a* (anchor) element to reference dynamically generated content via the *ecmascript* scheme defined by A/100-2 [38], Section 5.1.1.6.1.1.5, shall not be used by an ACAP-X application.

If an ACAP-X application does make use of this scheme, then activating the anchor shall be ignored, producing no side effect.

8.2.9.1.9 *frame* Element

The *src* attribute of a *frame* element shall be restricted to reference an application resource of content type `application/xhtml+xml`; furthermore, that resource shall be part of the current application.

If an entity of an ACAP-X application uses content type `application/xhtml+xml`, and the *src* attribute of a *frame* element references a destination resource that is not of content type `application/xhtml+xml`, then the user agent shall either: 1) ignore the reference and produce no side effect, or 2) present feedback to the end-user that the reference will/does produce no effect.

8.2.9.1.10 *object* Element

8.2.9.1.10.1 Active Content Object Element

The *object* element type may be used to reference active content objects as described by A/100-2 [38], Section 5.1.1.6.8.1, with the following restrictions.

The content type of the *object* element's immediate implementation, as referenced by the *classid* attribute, shall be restricted to `application/java` and shall adhere to those constraints defined by the ACAP-J environment.

The semantics of the *archive* attribute, if specified, shall be ignored by an ACAP-X environment.

Note: Use of the *archive* attribute is expected to be reintroduced in the future in order to support access to archive packaging of Java Xlets via a return channel.

The *codetype* attribute, if specified, shall be `application/java`.

The *height* and *width* specify the size of the Xlet's visible representation. If either height or width is zero or negative, then the ACAP-J method `javax.tv.graphics.TVContainer.getRootContainer()` shall return null for this embedded Xlet.

A *param* element with a *name* attribute of *appid* may be specified as a child of an active content *object* element, in which case the *value* attribute shall be a hexadecimal number prefixed with "0x" that corresponds to an application identifier listed in the ACAP-X application signaling. If more than one *param* element is specified, or the *appid* value does not correspond to any listed application identifier, then the embedded Xlet shall not be initialized.

Note: An *appid* parameter need not be specified. An explicit *appid* parameter is typically used to distinguish among multiple embedded Xlets for the purpose of supporting inter-Xlet communications.

8.2.9.1.10.2 Trigger Object Element

The ability for an *object* element to serve as a special trigger object defined by A/100-2 [38], Section 5.1.1.6.8.2, shall not be used by an ACAP-X application.

If an ACAP-X application does make use of this form of the *object* element, then it shall be ignored, producing no side effect.

8.2.9.1.11 *script* Element

The time of evaluation of script content contained in or referenced by a *script* element is implementation-dependent, except that it shall occur at a time no later than immediately prior to the dispatching of the `org.atsc.document.domstable` event to the Window object containing the document in which the *script* element appears.

The order of evaluation of script content contained in or referenced by more than one *script* element contained within a document shall be the same as pre-traversal element order.

In a multiple frameset document, the order of evaluation of script content contained in or referenced by *script* elements contained in documents referenced by *frame* elements is implementation-dependent. However, any script content contained in or referenced by a *script* element contained in a document containing a *frameset* element shall occur prior to the evaluation of script content contained in or referenced by *script* elements contained in documents referenced by *frame* elements that occur in the same document as the *frameset* element.

8.2.9.2 Extensions

This section describes extensions to the markup content facility defined by A/100-2 [38], Section 5.1.

8.2.9.2.1 Document Type Declaration

An ACAP-X application may use and an ACAP-X environment shall support references to document types that satisfy the definition of an ACAP-X *Family Document Type*, defined as follows:

The set of *ACAP-X Family Document Types* is defined to include the following:

- The ACAP-X Document Type (XDML) defined by Annex B in this standard.
- All future standard and non-standard (proprietary) ACAP-X Document Types such that: 1) any non-standard element type or attribute is declared to use qualified names, the prefix of the qualified name is not *acap*, and the XML namespace associated with the qualified

name is not “http://www.atssc.org/acap#markup”; 2) any alteration to an existing element’s content model is strictly backward compatible from the perspective of validating the element’s children.

The formal public identifier (FPI) used to label and reference an ACAP-X Family Document Type shall adhere to one of the following forms, where *x* and *y* indicate a major and minor version number of a specific standard XDML document type defined for use by a published edition of an ACAP specification:

- -//ATSC//DTD XHTML ACAP-X XDML x.y//EN"
- -//ATSC//DTD XHTML ACAP-X XDML x.y ProprietaryNameAndVersion//EN"
- -//W3C//DTD XHTML Basic 1.0//EN"

Non-standard (proprietary) ACAP-X Family Document Types shall employ the second of the above forms and shall supply a non-empty value for *ProprietaryNameAndVersion*, which shall consist of at least two space separated tokens the last of which is a version number (preferably in *major.minor* form) and the remaining (initial) tokens designate a unique organization and or specification identity. The value(s) employed for the proprietary name should be chosen in such a manner as to minimize the probability of colliding with other names. The substring **ACAP** (in any combination of lower- and upper-case) shall not appear in the proprietary name.

Note: The value of the *ProprietaryNameAndVersion* tokens are further restricted by the syntactic constraints implied by an XML Formal Public Identifier (FPI).

8.2.9.2.2 *cite* Attribute

For those element types that admit a *cite* attribute, the value of that attribute may be a URI that references a resource of content type application/xhtml+xml, as defined by Section 8.2.9, “Markup Content.”

8.2.9.2.3 *event* Attributes

An ACAP-X application may use and an ACAP-X environment shall support the following event attributes, where the *acap* namespace prefix is bound to the namespace URI <http://www.atssc.org/acap#markup>:

- *acap:ondomstable*
- *acap:onload*
- *acap:onunload*

Note: For the purposes of defining the DTD for ACAP-X markup content and for use in actual instance documents, the namespace prefix employed by these attributes is fixed.

The values of these attributes shall adhere to the syntax of the *StatementList* non-terminal of ECMAScript [50], Section 12.1; in particular, it shall be syntactically valid for use as the content of the *Block* non-terminal.

Use of these event attributes shall have the same semantic effect as registering a DOM-2 event handler to receive the *org.atssc.document.domstable*, *load*, and *unload* event types, respectively.

Note: See Section 8.2.11.2.1.3, “TriggerEvent Object,” for further information on the *org.atssc.document.domstable*, *load*, and *unload* event types.

8.2.9.2.4 *longdesc* Attribute

For those element types that admit a *longdesc* attribute, the value of that attribute may be a URI that references a resource of content type `application/xhtml+xml`, as defined by Section 8.2.9, “Markup Content.”

8.2.9.2.5 *a* (anchor) Element

8.2.9.2.5.1 Application Replacement and Launching

An ACAP-X application may use and an ACAP-X environment shall support references to content types `application/acap-j` and `application/acap-x` as the destination of an anchor (*a*) element, where the reference identifies an ACAP-J or ACAP-X application.

If an *a* (anchor) element’s destination content type is `application/acap-j` or `application/acap-x`, and a *target* attribute is specified with a value that designates the current top level frame of the application, then anchor activation shall cause: 1) termination of the current application, and 2) instantiation of a new ACAP-J or ACAP-X application instance.

The process of instantiating an application by activation of an anchor while terminating the activating applications is referred to as *application replacement*.

If an *a* (anchor) element’s destination content type is `application/acap-j` or `application/acap-x`, and no *target* attribute is specified or a *target* attribute is specified with a value of `_blank`, then anchor activation shall cause the instantiation of a new ACAP-J or ACAP-X application instance.

The process of instantiating an application by activation of an anchor without terminating the activating applications is referred to as *application launching*.

If an *a* (anchor) element’s destination content type is `application/acap-j` or `application/acap-x`, and a *target* attribute is specified with a value other than one that designates the top level frame of the application or the target `_blank`, then an attempt to perform anchor activation shall not replace the current application or launch a new application.

Note: See Section 14.2.1, “ACAP URI Scheme,” for further information on referencing an ACAP application.

A query component may appear in a resource identifier used to reference an ACAP-J or ACAP-X application for the purpose of launching a new application or replacing the current application. The format of the optional query component shall be identical to that used by the `application/x-www-form-urlencoded` content type as described by HTML [54], Section 17.13.4, “Form Content Types.”

In case the new or replacement application is an ACAP-X application, the unordered union of name and value pairs that appear in: 1) the resource reference, 2) the application signaling, and 3) the application parameter elements of the ACAP-X application metadata resource, if specified, shall form the set of application parameters to the ACAP-X application, and shall be exposed to the application as a query component of the URL represented by the `Window::location` property of the application’s top-level `Window` object. If the same name appears in more than one of the above three locations, then the value used shall be determined according to a precedence established by the order of locations specified here, with the resource reference given the highest precedence.

The same behavior shall apply for launching or replacement with an ACAP-J application, except that: 1) no application metadata resource applies, and 2) the application parameters are exposed through the `javax.tv.xlet.XletContext.getXletProperty(XletContext.ARGS)` method as described by MHP [2], Section 11.7.1.1.

8.2.9.2.5.2 Service Selection

An ACAP-X application may use and an ACAP-X environment shall support references to content type `video/mpeg` as the destination of an anchor (*a*) element, where of the reference identifies a service (virtual channel) in an MPEG-2 transport stream.

If an *a* (anchor) element's destination content type is `video/mpeg`, then anchor activation shall cause service selection within the current service context.

Note: See Section 14.2.1, "ACAP URI Scheme," for further information on referencing a service.

Note: A side-effect of service selection may be that the activating application is terminated if the application is a service bound application and it is not signaled in the referenced service.

8.2.9.2.5.3 Service Component Selection

An ACAP-X application may use and an ACAP-X environment shall support references to content type `video/mpv` and `audio/ac3` as the destination of an anchor (*a*) element, where of the reference identifies a service component (program element) in an MPEG-2 transport stream.

If an *a* (anchor) element's destination content type is `video/mpv` or `audio/ac3`, then anchor activation shall cause service component selection within the current service.

Note: See Section 14.2.1, "ACAP URI Scheme," for further information on referencing a service component.

8.2.9.2.6 *area* Element

An ACAP-X application may use and an ACAP-X environment shall support application replacement and launching, service selection, and service component selection by means of the *area* element in an identical manner to that of the *a* (anchor) element as described in Section 8.2.9.2.5, "*a* (anchor) Element."

8.2.9.2.7 *meta* Element

The following metadata items are defined as extensions to the set of items defined by A/100-2 [38], Section 5.1.1.6.7:

- Classpath

8.2.9.2.7.1 Classpath Metadata Item

The class path to be used when loading application defined Java class files for use with embedded Xlets or the Inter-Environment Bridge may be specified by using a *meta* element with a *name* attribute with a value `Classpath`. In this case, the value of the *content* attribute shall be a form that corresponds to that specified by A/100-1 [5], Section 6.1.1.6.13.2.

If an ACAP-X application includes an application metadata resource (AMR) and the AMR specifies a *classpath* application parameter, then any value specified by a `Classpath` metadata item using a *meta* element shall be appended to the effective classpath.

8.2.9.2.8 *object* Element

If an *object* element's *data* attribute references a non-streaming audio resource as the object instance data and the *classid* attribute is not specified, then a *param* child element with the name `loop` and the value of either `true` or `false` may be specified in order to indicate that the non-streaming

audio resource is to be repeatedly presented in a loop. If a loop *param* element child with the value true is not specified, then the non-streaming audio resource shall be presented only once.

8.2.10 Stylesheet Content

An ACAP-X application may use and an ACAP-X environment shall support the *Stylesheet Content* facilities defined by A/100-2 [38], Section 5.2, as restricted and extended below.

8.2.10.1 Restrictions

The following subsections describe restrictions upon the stylesheet content facility defined by A/100-2 [38], Section 5.2.

8.2.10.1.1 Resource Content Type References

References to resources from stylesheet content rules, properties, or descriptors that take the form of a URI shall be limited to those content types checked as supported in Table 8.3. If a reference is made to a resource of a content type that is not supported in the referencing context, then the behavior of an ACAP-X environment is implementation dependent, and shall not be relied upon by an ACAP-X application. If a content type is not listed in this table, then it shall be construed as unsupported.

Table 8.3 Stylesheet Resource Content Type References

Property/ Descriptor	application/font-tdpfr	image/{jpeg,mpeg,png}	text/css	video/mng	video/mpeg	video/mpv
background-image	x	✓	x	✓	✓	✓
list-style-image	x	✓	x	✓	✓	✓
@font-face src	✓	x	x	x	x	x

8.2.10.1.2 Media Types

An ACAP-X environment is not required to support the *atsc-tv* media type as defined by A/100-2 [38] Section 5.2.1.7.1.

8.2.10.1.3 Properties

8.2.10.1.3.1 *atsc-nav-index* Property

The *atsc-nav-index* property as defined by A/100-2 [38], Section 5.2.1.8.3.2, shall not be used by ACAP Applications; rather, the *nav-index* property should be used instead, as defined by Section 8.2.10.2.4.4, “*nav-index* Property”.

8.2.10.1.3.2 *atsc-nav-{left,right,up,down}* Properties

The *atsc-nav-{left,right,up,down}* properties as defined by A/100-2 [38], Section 5.2.1.8.3.3, shall not be used by ACAP Applications; rather, the *nav-{left,right,up,down}* properties should be used instead, as defined by Section 8.2.10.2.4.5, “*nav-{left,right,up,down}* Properties.”

8.2.10.1.4 Property Values

8.2.10.1.4.1 <color> Property Value

The `atsc-rgba()` functional notation as defined by A/100-2 [38], Section 5.2.1.8.4.1, shall not be used by ACAP Applications in those contexts that take a <color> property value; rather, the `rgba()` functional notation should be used instead, as defined by Section 8.2.10.2.5.1, “<color> Property Value Type.”

8.2.10.2 Extensions

8.2.10.2.1 Font Face Rule

An ACAP-X application may use and an ACAP-X environment shall support the units-per-em font descriptor within an `@font-face` rule, as defined by CSS2 [48], Section 15.3.4.

8.2.10.2.2 Viewport Rule

An ACAP-X application may use and an ACAP-X environment shall support an `@viewport` rule, defined according to the following extensions to CSS [48], Appendix D, “The Grammar of CSS2”:

`viewport`

```
VIEWPORT_SYM S* '{' S* declaration [ ';' S* declaration ]* '}' S*
```

In this grammar production, `VIEWPORT_SYM` denotes the lexical string “@viewport” (not including quotation marks) and `declaration` is as defined by CSS [48], Appendix D, Section D.1, “Grammar.” An instance of this production may appear in the same context as the `media non-terminal` as specified by the `stylesheet` production of CSS [48], Appendix D, Section D.1.

Only one `@viewport` rule shall apply to an XHTML document instance hierarchy presented by an ACAP-X application. If multiple `@viewport` rules are present, then the priority for determining the applicable rule shall be in accordance with CSS [48], Section 6.4, “The Cascade.”

The collection of declarations contained within an `@viewport` rule constitute a *viewport descriptor set*. Only the declarations that appear in the applicable rule shall be used to determine the viewport descriptor set. Each *viewport descriptor* is expressed as a name and a value pair using the syntax of the `declaration` production of CSS [48], Appendix D, Section D.1. The following sub-sections define the permissible descriptors that may be used by an ACAP-X application.

The purpose of the `@viewport` rule and its viewport descriptor set is to describe a reference region within the *canvas*, to establish a logical (user) coordinate space within this region, and to determine the bounds of the *initial containing block* within this coordinate space.

Note: In this standard, the canvas is considered to be coterminous with the graphics plane, as defined by Section 13, “Graphics Reference Model.”

Note: See CSS [48], Section 2.3.1, “The Canvas,” Section 9.1.1, “The Viewport,” and Section 9.1.2, “Containing Blocks,” for further information on *canvas*, *viewport*, and *initial containing block*, respectively.

If no `@viewport` rule is applicable when presenting an XHTML document instance hierarchy, then the viewport and the initial containing block shall be considered to be coterminous with and have the same horizontal and vertical resolution as the graphics plane.

Example

The following specifies a viewport that occupies one-fourth (1/4) of the graphics plane and is centered within the graphics plane. The logical (user) coordinate space of the viewport is 288x360 pixels, and the initial containing block is a 200x200 pixel square centered within the viewport.

```
@viewport {
  region : inset-rect(25%,25%,25%,25%);
  resolution : 288 360;
  initial-container : inset-rect(80,44,80,44)
}
```

8.2.10.2.2.1 Viewport Descriptors

The following descriptors may be specified in an @viewport rule:

- initial-container
- region
- resolution

Note: As used in this subsection, the term *descriptor* is unrelated to the term as used in the MPEG-2 context.

8.2.10.2.2.1.1 *initial-container* Descriptor

An ACAP-X application may use and an ACAP-X environment shall support the *initial-container* viewport descriptor, as defined below.

Name:	initial-container
Value:	auto <shape>
Initial:	auto
Applies to:	the viewport
Inherited:	N/A
Percentages:	relative to viewport's extent
Media:	visual

Values of <shape> are limited to rect() and inset-rect(); the semantics of all values are specified below.

auto – The initial container block's computed origin and extent are the same as the viewport's origin and extent.

rect(top, right, bottom, left) – Each of the four arguments can be a <length> or a <percentage>. All length values are offsets relative to the origin of the viewport, which is its top left vertex. All <percentage> values are computed relative to the viewport's extent. The computed width and height of the initial container block are determined by subtracting the left from the right for the width, and similarly top from bottom for the height. However, if this computation results in a negative value, it is considered to be zero.

inset-rect(top, right, bottom, left) – Like rect(), except that the values are offsets relative to the respective edges of the viewport.

When specifying a <length> or a <percentage> as an argument to a shape function, the value may be negative.

The units of values specified as a <length> shall be restricted to pixels only. If no unit is specified, then pixels shall be assumed to be the units. Furthermore, values specified as pixels shall be interpreted in the viewport's logical (user) coordinate space, which may not be the same as the graphics plane's coordinate space.

Visible marks produced by formatting a markup content resource in an ACAP-X application may appear outside of the initial container block; however, all marks are clipped by the viewport region.

Note: Visible marks may be placed outside the initial container block by using absolute position style properties, in which case they are interpreted as relative to the viewport's origin, and not the initial container block's origin.

8.2.10.2.2.1.2 *region* Descriptor

An ACAP-X application may use and an ACAP-X environment shall support the *region viewport* descriptor, as defined below.

Name:	region
Value:	auto <shape>
Initial:	auto
Applies to:	the viewport
Inherited:	N/A
Percentages:	relative to graphics plane's extent
Media:	visual

Values of <shape> are limited to `rect()` and `inset-rect()`; the semantics of all values are specified below.

auto – The viewport's computed origin and extent are the same as the graphics plane's origin and extent.

rect(top, right, bottom, left) – Each of the four arguments can be a <length> or a <percentage>. All length values are offsets relative to the origin of the graphics plane, which is its top left vertex. All <percentage> values are computed relative to the graphics plane's extent. The computed width and height of the viewport are determined by subtracting the left from the right for the width, and similarly top from bottom for the height. However, if this computation results in a negative value, it is considered to be zero.

inset-rect(top, right, bottom, left) – Like `rect()`, except that the values are offsets relative to the respective edges of the graphics plane.

When specifying a <length> or a <percentage> as an argument to a shape function, the value may be negative.

All visible marks produced by formatting a markup content resource in an ACAP-X application shall be clipped to the region established by the *region* viewport descriptor. This descriptor effectively establishes the origin and extent of the *viewport* within the coordinate space of the *canvas* (i.e., the graphics plane).

8.2.10.2.2.1.3 *resolution* Descriptor

An ACAP-X application may use and an ACAP-X environment shall support the *resolution viewport* descriptor, as defined below.

Name:	Resolution
Value:	auto <number> <number>
Initial:	Auto
Applies to:	the viewport
Inherited:	N/A
Percentages:	N/A
Media:	Visual

The semantics of the values of this descriptor are specified below:

auto – The viewport’s logical (user) coordinate space has the same horizontal and vertical resolution as the coordinate space of the graphics plane (*canvas*); that is, an identity scale transform applies when mapping from the viewport coordinate space to the graphics plane coordinate space.

<number> <number> – The horizontal resolution (h_{res}) of the viewport’s logical (user) coordinate space is determined by the first <number>, while the vertical resolution (v_{res}) is determined by the second <number>; that is, the following scale transform applies when mapping the viewport coordinate space to the graphics plane coordinate space, where w and h represent the width and height of the viewport as determined by the region descriptor.

8.2.10.2.3 Media Types

An ACAP-X application may use and an ACAP-X environment shall support the *all* media type, as defined by CSS2 [48], and the *tv* media type, as defined by CSS-TV [35].

The following statement in CSS2 [48], Section 7.3, shall be considered to not apply to the *all* media type: “A user agent that claims to support a media type by name must implement all of the properties that apply to that media type.”

Note: Supporting the *all* media type effectively means that style declarations that appear in the context of an @media all rule are construed as applying to all media.

Note: The *tv* media type is also defined by CSS2 [48], however, this standard does not make use of that definition of the *tv* media type. Rather, this standard interprets the *tv* media type only with respect to its definition in CSS-TV [35].

8.2.10.2.4 Properties

An ACAP-X application may use and an ACAP-X environment shall support the following style properties as defined in this section:

- acap-dynamic-refresh
- crop
- nav-index
- nav-left
- nav-right
- nav-up
- nav-down

- opacity

In addition to defining additional properties, the following subsections extend the value space or value semantics of certain properties.

8.2.10.2.4.1 *acap-dynamic-refresh* Property

An ACAP-X application may use and an ACAP-X environment shall support the *acap-dynamic-refresh* property, as defined by A/100-2 [38], Section 5.2.1.8.3.1, with the name *acap-dynamic-refresh* being substituted for the name *atsc-dynamic-refresh*.

8.2.10.2.4.2 *crop* Property

An ACAP-X application may use and an ACAP-X environment shall support the *crop* property, as defined by CSS-BOX [33], Section 12.

Note: The *crop* property is intended to satisfy the functionality provided by the *clip-video* property of DVB-HTML as defined in MHP 1.1 [3].

For convenience sake, the definition of the *crop* property as specified in CSS-BOX [33] is as follows: “This property allows a replaced element to be just a rectangular area of an object, instead of the whole object.”

The *crop* property adds a step when determining the *intrinsic width* and *height* of an element. When the layout algorithms reference the “intrinsic width” (and/or height), they are referring to the computed intrinsic width and height. The computed intrinsic width and height of an element are the result of applying the *crop* to the actual intrinsic width and height of the element.

Name	crop
Value	auto <shape>
Initial	Auto
Applies to:	replaced elements
Inherited:	No
Percentages:	relative to intrinsic size
Media	visual
Computed value	specified value

Values of <shape> are limited to *rect()* and *inset-rect()*; the semantics of all values are specified below.

auto – The element’s computed intrinsic width and height are the same as its actual intrinsic width and height.

rect(top, right, bottom, left) – Each of the four arguments can be a <length> or a <percentage>. All percentage values are computed relative to the intrinsic dimensions of the element. Values are offsets relative to the top left of the element. The computed intrinsic width and height of the element are determined by subtracting the left from the right for the width, and similarly top from bottom for the height. However, if this computation results in a negative value, it is considered to be zero.

inset-rect(top, right, bottom, left) – Like *rect()*, except that the values are offsets relative to the respective edges of the element.

8.2.10.2.4.3 *font* Property

An ACAP-X application may use and an ACAP-X environment shall support the following system font names as values for the font property in accordance with CSS2 [48], Section 15.2.5:

- caption
- icon
- menu
- message-box
- small-caption
- status-bar

The mapping from these system font names to actual system font resources is implementation dependent.

8.2.10.2.4.4 *nav-index* Property

An ACAP-X application may use and an ACAP-X environment shall support the *nav-index* property, as defined by CSS-UI [36], Section 9.2.2.

Note: The *nav-index* property is intended to satisfy the functionality provided by the *atsc-nav-index* property of A/100-2 [38] and the *nav-index* property defined by DVB-HTML as defined in MHP 1.1 [3].

For convenience sake, the definition of the *nav-index* property as specified in CSS-UI [36] is as follows: “This property specifies the position of the current element in the sequential navigation order for the current document.”

The sequential navigation order defines the order in which elements will receive focus when navigated by the user via an input device. The sequential navigation order may include elements nested within other elements.

Name	nav-index
Value	auto <number> inherit
Initial	auto
Applies to:	all enabled elements
Inherited:	no
Percentages:	N/A
Media	interactive

The semantics of all values are specified below.

auto – The element's sequential navigation order is assigned automatically by the user agent.

<number> – The number (which is non-zero and positive) indicates the sequential navigation order for the element. ‘1’ means first. Elements with the same *nav-index* value are navigated in document order when that *nav-index* value is being navigated.

Elements that may receive focus should be navigated by user agents according to the following rules:

- 1) Those elements that support the *nav-index* property and assign a positive value to it are navigated first. Navigation proceeds from the element with the lowest *nav-index* value to the element with the highest value. Values need not be sequential nor must they begin with any

particular value. Elements that have identical *nav-index* values should be navigated in the order they appear in the character stream.

- 2) Those elements that do not support the *nav-index* property or support it and assign it a value of ‘auto’ are navigated next. These elements are navigated in the order they appear in the character stream.
- 3) Elements that are disabled do not participate in the sequential navigation order.

The actual key sequence that causes sequential navigation or element activation depends on the configuration of the user agent (e.g., the “tab” key is often used for sequential navigation, and the “enter” key is used to activate a selected element).

User agents may also define key sequences to navigate the sequential navigation order in reverse. When the end (or beginning) of the tabbing order is reached, user agents may circle back to the beginning (or end). “shift-tab” is often used for reverse sequential navigation.

8.2.10.2.4.5 *nav-{left,right,up,down}* Properties

An ACAP-X application may use and an ACAP-X environment shall support the *nav-left*, *nav-right*, *nav-up*, and *nav-down* properties, as defined by CSS-UI [36], Section 9.2.3.

Note: The *nav-** properties are intended to satisfy an essential subset of the functionality provided by the *atsc-nav-** properties defined by A/100-2 [38], Section 5.2.1.8.3.3, and the *nav-** properties defined by MHP 1.1 [3], Section 8.8.5.10.

For convenience sake, the definition of the *nav-left*, *nav-right*, *nav-up*, and *nav-down* properties as specified in CSS-UI [36] is as follows:

Name	nav-left, nav-right, nav-up, nav-down
Value	auto <uri> inherit
Initial	Auto
Applies to:	all enabled elements
Inherited:	No
Percentages:	N/A
Media	interactive

The semantics of all values are specified below.

auto – The user agent automatically determines which element to navigate the focus to in response to directional navigational input.

<uri> – The <uri> should indicate (through the use of a fragment identifier) the element to which the focus is to be navigated to in response to directional navigation input respective to the specific property.

User agents for devices with keyboards with arrow keys may respond to the four directional arrow keys (up arrow, right arrow, down arrow, left arrow) by navigating the focus according to four respective *nav-** directional navigation properties (*nav-up*, *nav-right*, *nav-down*, *nav-left*).

8.2.10.2.4.6 *opacity* Property

An ACAP-X application may use and an ACAP-X environment shall support the opacity property, as defined by CSS-COLOR [34], Section 3.2.

For convenience sake, the definition of the opacity property as specified in CSS-COLOR [34] is as follows:

Name	opacity
Value	< <i>alphavalue</i> > inherit
Initial	1.0
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media	interactive

The semantics of all values are specified below.

<alphavalue> – The uniform opacity setting to be applied across an entire object. Any values outside the range 0.0 (fully transparent) to 1.0 (fully opaque) will be clamped to this range. If the object is a container element, then the effect is as if the contents of the container element were blended against the current background using a mask where the value of each pixel of the mask is <*alphavalue*>.

8.2.10.2.5 Property Value Types

The following subsections describe additional property value types or extensions to existing property values types.

8.2.10.2.5.1 <color> Property Value Type

An ACAP-X application may use and an ACAP-X environment shall support the `rgba()` functional notation and the `transparent` keyword defined by CSS-COLOR [34], Sections 4.2.2 and 4.2.3, respectively, as a value in those contexts that specify use of the <color> property value as defined by CSS [48] Section 4.3.6.

Note: The `rgba()` functional notation is intended to satisfy the functionality provided by the `atsc-rgba()` functional notation defined by A/100-2 [38], Section 5.2.1.8.4.1.

Note: The `transparent` keyword can be considered a shorthand for `rgba(0,0,0,0)`.

For convenience sake, the definition of the `rgba()` functional notation as specified in CSS-COLOR [34] is as follows. “The format of an RGBA value in the functional notation is 'rgba(' followed by a comma-separated list of three numerical values (either three integer values or three percentage values), followed by an <*alphavalue*>, followed by ')'. The integer value 255 corresponds to 100 percent; e.g., `rgba(255,255,255,0.8) = rgba(100%,100%,100%,0.8)`. Whitespace characters are allowed around the numerical values.

8.2.11 Script Content

An ACAP-X application may use and an ACAP-X environment shall support the *Script Content* facilities defined by A/100-2 [38], Section 5.3, as restricted and extended below.

8.2.11.1 Restrictions

The following subsections describe restrictions upon the script content facility defined by A/100-2 [38], Section 5.3.

8.2.11.1.1 HTML Module Objects

8.2.11.1.1.1 HTMLDocument Object

The following methods of the HTMLDocument host object as defined by A/100-2 [38], Section 5.3.1.2.3.3, shall not be used by an ACAP-X application:

- write(DOMString)
- writeIn(DOMString)

If an attempt is made by an ACAP application to resolve a reference to the one of the above methods, then a run-time exception shall be raised.

During the construction of an HTMLDocument node instance and prior to the dispatch of the org.acap.document.domstable event, the HTMLDocument node and all descendant nodes shall be read-only. After the entire document is parsed and corresponding node construction has completed and immediately prior to dispatching the org.acap.document.domstable event, the HTMLDocument node and all descendant nodes shall be reverted to mutable, unless they are otherwise required to be immutable.

8.2.11.1.1.2 HTMLFormElement Object

The HTMLFormElementExt interface extension to HTMLFormElement as defined by A/100-2 [38], Section 5.3.1.2.3.4, shall not be used by an ACAP-X application.

If an attempt is made by an ACAP application to resolve a reference to the above interface, then a run-time exception shall be raised.

8.2.11.1.1.3 HTMLImageElement Object

The property HTMLImageElementExt::lowsrc extension to HTMLImageElement as defined by A/100-2 [38], Section 5.3.1.2.3.5, shall not be used by an ACAP-X application.

If an attempt is made by an ACAP application to resolve a reference to the above property, then a run-time exception shall be raised.

8.2.11.1.1.4 HTMLObjectElement Object

The property HTMLObjectElementExt::lowsrc extension to HTMLObjectElement as defined by A/100-2 [38], Section 5.3.1.2.3.7, shall not be used by an ACAP-X application.

If an attempt is made by an ACAP application to resolve a reference to the above property, then a run-time exception shall be raised.

The HTMLTriggerObjectElementExt interface extension to HTMLObjectElement as defined by A/100-2 [38], Section 5.3.1.2.3.7, shall not be used by an ACAP-X application.

If an attempt is made by an ACAP application to resolve a reference to the above interface, then a run-time exception shall be raised.

8.2.11.1.2 StyleSheets Module Objects

The following interfaces and host object as defined by A/100-2 [38], Sections 5.3.1.2.5 and 5.3.1.2.6, shall not be used by an ACAP-X application and need not be supported by an ACAP-X environment:

- Counter
- CSSCharsetRule
- CSSFontFaceRule

- CSSImportRule
- CSSMediaRule
- CSSPrimitiveValue
- CSSRule
- CSSRuleList
- CSSStyleRule
- CSSStyleSheet
- CSSUnknownRule
- CSSValue
- CSSValueList
- DocumentCSS
- DocumentStyle
- DOMImplementationCSS
- LinkStyle
- MediaList
- Rect
- RGBColor
- StyleSheet
- StyleSheetList
- ViewCSS

Note: The only stylesheet related interfaces and host objects required to be supported after removing the above are: `CSSStyleDeclaration` and `ElementCSSInlineStyle`.

In addition, the method `CSSStyleDeclaration::getPropertyCSSValue` and the property `CSSStyleDeclaration::parentRule` shall not be used by an ACAP-X application.

If an attempt is made by an ACAP application to resolve a reference to one of the above interfaces and host objects, methods, or properties, then a run-time exception shall be raised.

8.2.11.1.3 Event Types

This section describes restrictions to the *Event Set Module Objects* specified by A/100-2 [38], Section 5.3.1.2.8.

8.2.11.1.3.1 HTML Event Types

As described in A/100-2 [38], Section 5.3.1.2.1.4.1, the `HTMLEvents` feature string shall return *true* when the `DOMImplementation::hasFeature` method is invoked. As a consequence, the event types defined by DOM2-EVENTS [49], Section 1.6.5, “HTML Event Types,” shall be generated at the appropriate times by an ACAP-X environment and dispatched to registered event listeners.

An HTML event shall be instantiated and dispatched as an Event object as described in DOM2-EVENTS [49], Section 1.6.5.

Note: When creating an Event object that corresponds with an HTML event type, the *eventType* argument of the `DocumentEvent::createEvent` method is “HTMLEvent” as describe in DOM2-EVENTS [49], Section 1.6.5.

8.2.11.1.4 Environment Module Objects

This section describes restrictions on certain *Environment Module* objects.

8.2.11.1.4.1 Navigator Object

The following properties of the Navigator host object as defined by A/100-2 [38], Section 5.3.1.2.9.3, shall not be used by an ACAP-X application and need not be supported by an ACAP-X environment:

- ddeBackChannel
- ddeContentLevel
- ddeSourceId
- ddeEnabled
- ddeReleasable

If an attempt is made by an ACAP application to resolve a reference to one of the above properties, then a run-time exception shall be raised.

8.2.11.2 Extensions

8.2.11.2.1 Event Module Objects

This section describes extensions to the Events Module specified by A/100-2 [38], Section 5.3.1.2.7.

8.2.11.2.1.1 ApplicationEvent Object

An ACAP-X application may use and an ACAP-X environment shall support an ApplicationEvent object that implements the following interface:

```
interface ApplicationEvent : Event
{
    /* read-write properties */
    attribute DOMString detail;
    /* methods */
    void initApplicationEvent(in DOMString type, in DOMString detail);
};
```

8.2.11.2.1.1.1 ApplicationEvent::detail

The value of this mutable, string valued property is not defined, but may be used by application dependent logic for passing information between intermediate targets during the event capture phase.

8.2.11.2.1.1.2 ApplicationEvent::initApplicationEvent

The initApplicationEvent method is used to initialize the value of a ApplicationEvent created through the DocumentEvent interface. This method may only be called before the ApplicationEvent has been dispatched via the dispatchEvent method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence. This method has no effect if called after the event has been dispatched.

8.2.11.2.1.2 TimerEvent Object

An ACAP-X application may use and an ACAP-X environment shall support a TimerEvent object that implements the following interface:

```
interface TimerEvent : Event
{
    /* read-write properties */
    attribute DOMString detail;
    attribute unsigned long interval;
    /* methods */
    void initTimerEvent(in DOMString detail, in unsigned long interval);
};
```

8.2.11.2.1.2.1 TimerEvent::detail

An application supplied string used to provide detail and linkage from the registration of the timer to timer event handler. This property is considered opaque by the user agent.

8.2.11.2.1.2.2 TimerEvent::interval

This mutable, integer valued property indicates the timer restart interval in milliseconds. The default action of a timer event causes the timer to be restarted with this interval. This default action may be canceled by either: 1) initializing or setting the value of this interval to zero, or 2) invoking the method Event::preventDefault on the TimerEvent object.

8.2.11.2.1.2.3 TimerEvent::initTimerEvent

The initTimerEvent method is used to initialize the value of a TimerEvent created through the DocumentEvent interface. This method may only be called before the TimerEvent has been dispatched via the dispatchEvent method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence. This method has no effect if called after the event has been dispatched.

8.2.11.2.1.3 TriggerEvent Object

An ACAP-X application may use and an ACAP-X environment shall support a TriggerEvent object that implements the following interface.

```
interface TriggerEvent : Event
{
    /* constants */
    const unsigned long TIME_NONE = 0;
    const unsigned long TIME_NPT = 1;
    /* read-only properties */
    readonly attribute unsigned long timeType;
    readonly attribute DOMString time;
    /* read-write properties */
    attribute DOMString detail;
    /* methods */
    void initTriggerEvent(in DOMString type,
        in unsigned long timeType, in DOMString time, in DOMString detail);
};
```

A `TriggerEvent` object shall be constructed by means of the `DocumentEvent::createEvent` method, where the `eventType` argument to this method shall be “org.atsc.trigger”.

Example

The following ECMAScript fragment creates a `TriggerEvent` object corresponding to an asynchronous trigger event, initializes the object, then dispatches it to the `Window` object, which, in turn dispatches it to the document and its descendant element hierarchy. The `type` property of the event is “org.xyz.myTriggerEvent”; the `detail` property is “payload”.

```
var e = document.createEvent ( "org.atsc.trigger" );
e.initTriggerEvent ( "org.xyz.myTrigger", TriggerEvent.TIME_NONE, null, "payload" );
window.dispatchEvent ( e );
```

8.2.11.2.1.3.1 `TriggerEvent::timeType`

This immutable, enumerated valued property shall contain one the values `TIME_NONE` or `TIME_NPT`. This property determines how to interpret the value of `TriggerEvent::time`.

If the value of this property is `TIME_NONE`, then the value of `TriggerEvent::time` shall be either null or an empty string.

If the value of this property is `TIME_NPT`, then the value of `TriggerEvent::time` shall be the string representation of an integer normal play time value.

8.2.11.2.1.3.2 `TriggerEvent::time`

This immutable, string valued property shall contain a string representation of a trigger’s time, if the trigger is a synchronized trigger, or null, if the trigger is asynchronous. See Section 8.2.11.2.1.3.1, “`TriggerEvent::timeType`,” for information on the interpretation of a non-null value.

8.2.11.2.1.3.3 `TriggerEvent::detail`

This mutable, string valued property shall contain a string representation of a trigger’s payload. The syntax and semantics of this string are application defined, and considered opaque by the user agent.

8.2.11.2.1.3.4 `TriggerEvent::initTriggerEvent`

The `initTriggerEvent` method is used to initialize the value of a `TriggerEvent` created through the `DocumentEvent` interface. This method may only be called before the `TriggerEvent` has been dispatched via the `dispatchEvent` method, though it may be called multiple times during that phase if necessary. If called multiple times, the final invocation takes precedence. This method has no effect if called after the event has been dispatched.

The `type` parameter of this method shall be used to initialize the *type* property of the object. The `detail` parameter of this method shall be used to initialize the *detail* property of the object. The *bubbles* and *cancelable* properties of the object shall be initialized to `false`.

8.2.11.2.2 Event Types

This section describes extensions to the *Event Set Module Objects* specified by A/100-2 [38], Section 5.3.1.2.8.

8.2.11.2.2.1 HTML Event Types

In addition to the events defined by DOM2-EVENTS [49], Section 1.6.5, “HTML Event Types”, the following event type shall be generated and dispatched by an ACAP-X environment as an HTML event type:

- `org.atsc.document.domstable`

8.2.11.2.2.1.1 `org.atsc.document.domstable` Event

The `org.atsc.document.domstable` event is used to indicate the completed construction of a document instance (i.e., its DOM instance).

Name:	<code>org.atsc.document.domstable</code>
Bubbles:	no
Cancelable:	no
Default Action:	none
Context:	none

This event shall be generated by an ACAP-X environment and dispatched to the Window object associated with the document being constructed.

This event signals the completed construction of the current document instance. After receipt of this event, any modification to the document instance is restricted to programmatic control by script content.

The dispatching of this event shall precede in time the dispatching of any other HTML event type; however, both application lifecycle events and timer events may be dispatched prior to the dispatching of the `org.atsc.document.domstable` event.

The target of the `org.atsc.document.domstable` event type shall be the frameset element node of a multiple-frame document; otherwise, the target shall be the body element node.

8.2.11.2.2.1.2 `org.atsc.document.load` Event

The `org.atsc.document.load` event shall be generated when the DOM implementation finishes loading all content within a document, all frames within a frame, or within an object element. The dispatching of the `org.atsc.document.load` event shall follow in time the initiation of playback of any media object that is referenced by an `img` or `object` element in the document to which this event is to be dispatched and the media objects are automatically started without explicit user or application defined programmatic action.

Name:	<code>org.atsc.document.load</code>
Bubbles:	no
Cancelable:	no
Default Action:	none
Context:	none

This event shall be generated by an ACAP-X environment and dispatched to the Window object associated with the document being constructed.

8.2.11.2.2.1.3 `org.atsc.document.unload` Event

The `org.atsc.dom.unload` event shall be generated when the DOM implementation removes a document from a window or frame.

The dispatching of the `org.atsc.document.unload` event shall follow in time the cessation of playback of any media object that is referenced by an `img` or `object` element in the document to which this event is to be dispatched and the media objects were automatically started without explicit user or application defined programmatic action.

Name: **org.atsc.document.unload**
Bubbles: no
Cancelable: no
Default Action: none
Context: none

This event shall be generated by an ACAP-X environment and dispatched to the Window object associated with the document being constructed.

8.2.11.2.2.2 Application Lifecycle Event Types

This section specifies the following application lifecycle event types:

- `org.atsc.application.started`
- `org.atsc.application.suspending`
- `org.atsc.application.resumed`
- `org.atsc.application.terminating`

An ACAP-X environment shall dispatch these events to the top-level Window object of the application to which they apply as described in the following subsections.

The target of all application lifecycle events shall be the top-level Window object of the ACAP-X application.

An ACAP-X application is not notified of a transition to the `LOADING` or `KILLED` states.

8.2.11.2.2.2.1 `org.atsc.application.started` Event

The `org.atsc.application.started` event is used to indicate that an ACAP-X application has just been activated. More specifically, the application has just transitioned from the `LOADING` state to the `ACTIVE` state as defined by the application lifecycle state model.

Name: **org.atsc.application.started**
Bubbles: No
Cancelable: No
Default Action: None
Context: None

The dispatching of the `org.atsc.application.started` event shall precede in time the initiation of playback of any media object that is referenced by an `img` or `object` element in the document to which this event is to be dispatched and the media objects are automatically started without explicit user or application defined programmatic action.

8.2.11.2.2.2.2 `org.atsc.application.suspending` Event

The `org.atsc.application.suspending` event is used to indicate that an ACAP-X application is about to be suspended. More specifically, the application is about to be transitioned to the `PAUSED` state as defined by the application lifecycle state model.

Name: **org.atsc.application.suspending**
Bubbles: no
Cancelable: no
Default Action: none
Context: none

An ACAP-X application shall strictly limit the type and amount of processing performed in an event handler registered for this event type. An implementation dependent time limit of no less than one second may be enforced on such processing by an ACAP-X environment, after which the application may be forcibly suspended.

An `org.atsc.application.suspending` event shall not be dispatched to an application if no prior `org.atsc.application.started` event was dispatched to the application.

8.2.11.2.2.2.3 `org.atsc.application.resumed` Event

The `org.atsc.application.resumed` event is used to indicate that a previously paused (suspended) ACAP-X application has just been resumed. More specifically, the application has just transitioned from the PAUSED state to the ACTIVE state as defined by the application lifecycle state model.

Name: **org.atsc.application.resumed**
Bubbles: No
Cancelable: No
Default Action: None
Context: None

An `org.atsc.application.resumed` event shall not be dispatched to an application if no prior `org.atsc.application.suspending` event was dispatched to the application.

8.2.11.2.2.2.4 `org.atsc.application.terminating` Event

The `org.atsc.application.terminating` event is used to indicate that an ACAP-X application has been signaled for graceful destruction. More specifically, the application has just transitioned to the DESTROYED state as defined by the application lifecycle state model.

Name: **org.atsc.application.terminating**
Bubbles: no
Cancelable: no
Default Action: none
Context: none

An ACAP-X application shall strictly limit the type and amount of processing performed in an event handler registered for this event type. An implementation dependent time limit of no less than one second may be enforced on such processing by an ACAP-X environment, after which the application may be forcibly terminated (killed).

An `org.atsc.application.terminating` event shall not be dispatched to an application if no prior `org.atsc.application.started` event was dispatched to the application.

The dispatching of the `org.atsc.application.terminating` event shall follow in time the cessation of playback of any media object that is referenced by an *img* or *object* element in the document to which this event is to be dispatched and the media objects were automatically started without explicit user or application defined programmatic action.

8.2.11.2.2.3 Timer Event Types

The `org.atssc.timer` event is used to indicate the firing of a timer event.

Name:	org.atssc.timer
Bubbles:	No
Cancelable:	Yes
Default Action:	restart timer with same timer id using <i>interval</i> property to determine interval
Context:	Detail

A timer event shall be generated by an ACAP-X environment when a timer is fired as a side effect of invoking the `Window::startTimer()` method as described in Section 8.2.11.2.3.1.1, “`TimerDispatcher::startTimer`.” It shall be dispatched to the `Window` object on which the `startTimer` method was invoked.

A timer event shall be instantiated and dispatched as a `TimerEvent` object as described in Section 8.2.11.2.1.2, “`TimerEvent Object`.”

The target of a `org.atssc.timer` event shall be the `HTMLDocument` node of the document associated with the `Window` object in which the timer is started.

8.2.11.2.2.4 Trigger Event Types

The `org.atssc.trigger` event is used to indicate the dispatch of a generic trigger.

Name:	org.atssc.trigger
Bubbles:	No
Cancelable:	No
Default Action:	None
Context:	Detail

A generic trigger event is never generated automatically by an ACAP-X environment. This event may be programmatically constructed and dispatched by application defined script content to a `Window` object as determined by the ACAP-X application.

In addition to the generic trigger type defined above, application defined trigger types shall be generated by an ACAP-X environment upon receipt of an asynchronous trigger or upon the firing of a synchronized trigger as describe in Section 8.1.4, “`Trigger Processing`.”

A trigger event shall be instantiated and dispatched as a `TriggerEvent` object as described in Section 8.2.11.2.1.3, “`TriggerEvent Object`.”

The target of both the generic `org.atssc.trigger` event type as well as application defined trigger types shall be the `HTMLDocument` node of the document associated with the top-level `Window` object of the ACAP-X application which the trigger is associated.

8.2.11.2.3 Environment Module Objects

This section describes extensions on certain Environment Module objects.

8.2.11.2.3.1 Window Object

An ACAP-X application may use and an ACAP-X environment shall support the `DOM-2 EventTarget` interface on each `Window` object. Furthermore, the event flow for dispatched events shall start with the `Window` object and proceed as described in Section 8.1.3, “`Event Processing`.”

An ACAP-X application may use and an ACAP-X environment shall support the following extension interface on each Window object:

```
interface TimerDispatcher
{
    /* methods */
    TimerId startTimer(in DOMString detail, in unsigned long interval);
    void cancelTimer(in TimerId id);
};
```

8.2.11.2.3.1.1 TimerDispatcher::startTimer

This number valued method shall cause the repeated dispatch of an `org.atsc.timer` event to occur every *interval* milliseconds, returning a unique, opaque numeric identifier which may be subsequently used by `Window::cancelTimer`.

The *detail* parameter is application defined and is considered to be opaque to the user agent.

The value of the *interval* parameter may be zero, in which case a single timer event is asynchronously dispatched, and is not repeated unless the `TimerEvent::interval` property is modified by an event handler to be non-zero and the default action is not canceled.

8.2.11.2.4 Inter-Environment Bridge

An ACAP-X application may use and an ACAP-X environment shall support an interface between the ACAP-X and ACAP-J environments as specified in MHP 1.1 [3], Section 8.10.2, as further amended by this section, with the term *ACAP-J* substituted for *DVB-J* and the term *ACAP-X* substituted for *DVB-HTML*.

Any use of this functionality by an ACAP-X application shall be subject to security considerations defined by Section 8.3.2, “Inter-Environment Bridge Access.”

8.2.11.2.4.1 Packages Object

The Packages object described MHP 1.1 [3], Section 8.10.2, shall have a set of properties consisting of an ECMAScript object for each root package name that has been loaded into the class loader associated with the ACAP-X application or is present in the underlying ACAP-J implementation.

8.2.11.2.4.2 Package Object

Each ECMAScript object representing a Java package shall have a set of properties consisting of an ECMAScript object for each top-level class and subpackage in the Java package.

8.2.11.2.4.3 Java Class Object

Each ECMAScript object representing a Java class shall have a property of the type Function for each public, static method name in the Java class. It shall also have an internal property, `[[Constructor]]`, which allows it to be used by the ECMAScript `new` operator. When invoked, the function attempts to find the best matching signature among the corresponding Java methods or constructors using the rules specified in MHP 1.1 [3] Section 8.10.2.7. If a compatible method is found, the arguments are converted according to MHP 1.1 [3], Section 8.10.2.9, and the return values as specified in MHP 1.1 [3], Section 8.10.2.11.

8.2.11.2.4.4 Java Method Object

Each ECMAScript object representing a Java method shall implement a `[[Call]]` function that invokes the Java method. The arguments are matched against the signature of the Java method as specified in MHP 1.1 [3], Section 8.10.2.7. If the method is compatible, the arguments are converted according to MHP 1.1 [3], Section 8.10.2.9, and the return values as specified in MHP 1.1 [3], Section 8.10.2.11.

8.2.11.2.4.5 Behavior of Java Objects in ECMAScript

The ECMAScript objects representing Java entities shall behave as host objects as defined in the ECMAScript specification. Table 8.4 lists the requirements on the internal properties of ECMAScript objects representing Java entities.

Table 8.4 ECMAScript Internal Properties for Java Entities

Property	Requirements
<code>[[Prototype]]</code>	Shall be null.
<code>[[Class]]</code>	Implementation dependent.
<code>[[Value]]</code>	Implementation dependent.
<code>[[Get]]</code>	Per ECMAScript specification for native objects.
<code>[[Put]]</code>	Per ECMAScript specification for native objects.
<code>[[CanPut]]</code>	Per ECMAScript specification for native objects.
<code>[[HasProperty]]</code>	Per ECMAScript specification for native objects.
<code>[[Delete]]</code>	Per ECMAScript specification for native objects.
<code>[[DefaultVaue]]</code>	Shall be implemented as specified below.
<code>[[Construct]]</code>	When the corresponding Java object is an instantiable class., an instance of the class shall be constructed if the argument list matches a constructor. If no match is found, it throws a <code>TypeError</code> exception. For objects corresponding to a Java object that is not an instantiable class, e.g. an abstract class, it is undefined.
<code>[[Call]]</code>	For an object corresponding to a Java method (or overloaded methods), <code>[[Call]]</code> shall be implemented to invoke a method as described above. Otherwise it is undefined.
<code>[[HasInstance]]</code>	Shall be undefined on all objects corresponding to Java entities
<code>[[Scope]]</code>	Implementation dependent.
<code>[[Match]]</code>	Shall be undefined on all objects corresponding to Java entities

All properties specified herein on objects representing Java entities shall have the `ReadOnly` ECMAScript attribute, with the following exception. During the execution of an ECMAScript subclass constructor described in Section 8.2.11.2.4.8, “Subclassing” the properties on the current object shall not have the `ReadOnly` attribute.

Note: The lack of `[[Prototype]]` or `[[HasInstance]]` means that ECMAScript `instanceof()` is not useful on objects representing Java entities. Instead `java.lang.Class.isInstance()` should be used for querying the inheritance of Java objects.

Note: Because the behavior of type conversion operators is unchanged from that specified in ECMAScript [50], Section 9, invoking `ToBoolean` on a `java.lang.Boolean` always returns `true`, since that is the value returned for all objects. And because the behavior of `[[DefaultValue]]` is unchanged, `ToNumber` on a `java.lang.Object` returns the result of `toString()`, since that is the value returned for all objects.

8.2.11.2.4.6 Explicit Method Selection

The mechanism for explicit method selection specified in MHP 1.1 [3], Section 8.10.2.5 is not supported by the ACAP-X application environment. Instead, a set of three methods are provided to allow particular methods to be explicitly retrieved so that it may be invoked.

The ECMAScript object corresponding to a Java object shall have a function that returns an object representing the corresponding Java method. It behaves as a Java method with the signature:

```
Method getMethod(String methodName, String methodSignature)
```

The ECMAScript object corresponding to a Java class shall have a function for retrieving static methods that returns an object representing the corresponding Java method:

```
Method getStaticMethod(String methodName, String methodSignature)
```

In addition, it shall provide function for retrieving constructors that returns an object representing the corresponding Java constructor:

```
Constructor getConstructor(String methodSignature).
```

The signature string argument is a comma separated list of formal Java type specifiers. The signature must exactly match the argument list portion; i.e. that portion between the parentheses, of the string returned by `java.lang.reflect.Method.toString()` or `java.lang.reflect.Constructor.toString()`.

8.2.11.2.4.7 Method Signature Matching

An implementation is required to match signatures of both static and non-static methods when the invocation is attempted on an ECMAScript object representing an instance of a Java class. The second item in the bulleted list in MHP 1.1 [3], Section 8.2.10.7 is amended to read:

“be static when the invocation is attempted on the class, and may be static or non-static when invoked on an instance.”

8.2.11.2.4.8 Subclassing

`Subtype` is a function-valued property on the `Global` object which allows one to subclass an existing Java class. It returns an ECMAScript object that corresponds to the Java class representing the subclass. If the function is invoked to create a subclass of a final class, a `TypeError` exception is thrown.

Invoking `new` on the returned object constructs an instance of the subclass. During the creation of the instance, the constructor that was passed to `Subtype` is invoked on an object representing the Java object with the parent constructors invoked. The ECMAScript constructor function may then add function-valued properties for the methods implemented in ECMAScript. As specified in Section 8.2.11.2.4.5, “Behavior of Java Objects in ECMAScript,” properties corresponding to Java methods are writable only during this constructors invocation.

Method invocation from Java is performed as follows:

- If the method appears on a superclass and is final, that method is invoked.
- Otherwise, if the ECMAScript object has a function-valued property with the method name, that function is invoked.

- Otherwise, if the Java method is implemented on a superclass, that method is invoked.
- Otherwise, a Java runtime exception is thrown.

In the example of subclassing in MHP 1.1 [3], Section 8.10.2.10, the line

```
this.ActionPerformed = arg1;
```

is amended to read

```
this.SetAction(arg1);
```

In MHP 1.1 [3], Section 8.10.2.10, the references to subclassing shall be limited to the implementation of Java interfaces. There is no requirement to be able to subclass Java classes (e.g abstract ones). Invoking the Subtype constructor on a Java class (as opposed to an interface) may cause the DVException with the error code SUBCLASS_NOT_ALLOWED_ERR to be raised.

8.2.11.2.4.9 Exceptions

Exceptions thrown by Java called from ECMAScript are not converted to ECMAScript exceptions. They appear in ECMAScript as a Java object.

ECMAScript functions that throw exceptions when called from Java should throw subclasses of `java.lang.exception` or `java.lang.error` if they wish specific behavior on from the Java caller. If such a function throws another type of object, it appears in Java as a Java runtime exception with a message of "ECMAScript".

Where MHP 1.1 [3], Section 8.10.2 specifies that an object of type DVException is thrown, an ECMAScript TypeError shall be thrown instead.

8.2.11.2.4.10 Security

Per MHP 1.1 [3], Section 8.10.2.3, the Java runtime shall enforce the MHP security model when ECMAScript calls into Java. Hence, an implementation is required to maintain all contextual information needed by the Java Security model through multiple layers of calls between languages in both directions.

8.2.11.2.4.11 Unicode Escapes

As described in ECMA SCRIPT [50], Section 6, ECMAScript identifiers, strings and characters may contain Unicode escape sequences that start with a `\` followed by a single `u`, whereas Java allows a `\` followed by one or more instances of the letter `u`. An ACAP-X environment is required to normalize these representations for purposes of comparisons so that regardless of the original or intermediate representation, equivalent Unicode sequences are considered equivalent when used in strings, characters and identifiers.

8.3 ACAP-X Security Specifics

8.3.1 Cookie Access

Access to cookie state information items shall be controlled by the `acap:cookie` permission request as described by Section 12.4.2.3.1, "Cookie Permission."

8.3.2 Inter-Environment Bridge Access

Access to the inter-environment bridge shall be controlled by the `acap:bridge` permission request as described by Section 12.4.2.3.3, “Inter-Environment Bridge Permission.”

8.3.3 Runtime Code Extension Access

Access to the runtime code extension mechanisms shall be controlled by the `acap:rce` permission request as described by Section 12.4.2.3.2, “Runtime Code Extension Permission.”

8.4 ACAP-X Transport Specifics

The following metadata items shall be signaled in all ACAP-X application transport scenarios:

- application identifier
- root directory
- root resource

The *application identifier* metadata item shall specify a UUID, an organization identifier, and an organization specific application identifier. See Section 8.2.1.2.5, “*identifier* Element,” for further information.

The *root directory* metadata item shall specify a directory either as an absolute URI or an absolute directory within some implied file system namespace, as determined by the transport scenario. This directory shall serve as the default *base* directory for relative references to resources that do not otherwise specify or imply a base directory.

The *root resource* metadata item shall specify a relative or absolute reference to the resource that represents the ACAP-X application’s root entity. If the reference is a relative reference, then it shall be resolved relative to the *root directory* metadata item. If it is an absolute directory with an implied file system namespace, and that implied file system is a broadcast object carousel, then it shall be interpreted as a file pathname starting with the root directory of the object carousel.

The *root resource* metadata item shall reference either an application metadata resource or a markup content resource as defined by this standard. If a markup content resource is referenced as the root resource, then an application metadata resource shall be implied as described in Section 8.1.1, “Application Behavior.”

8.4.1 ACAP-X Transport Binding

When delivered within an ATSC or Digital Cable Television compliant transport stream, an ACAP-X application shall be signaled and as described by Section 10, “Transport and Signaling.”

This standard does not preclude the use of other transport bindings for other delivery mechanisms.

8.4.1.1 Bounded Resource Encapsulation

A bounded resource (i.e., a resource of definite length) that is included in an ACAP-X application shall be encapsulated as a DSM-CC U-U BIOP::File object that makes use of any form of an interoperable object reference (IOR) as permitted by Section 10.4.1.2, “Interoperable Object References.”

8.4.1.2 Unbounded Resource Encapsulation

An unbounded resource (i.e., a resource of indefinite length) that is referenced by an ACAP-X application is limited to the streaming audio and stream video content types permitted by Sections

8.2.6, “Streaming Audio Content,” and 8.2.5, “Streaming Video Content.” These resources are limited to elementary streams carried directly by the MPEG-2 transport stream, and are not transported by the DSM-CC UU Object Carousel.

8.4.1.3 Trigger Encapsulation

An asynchronous or synchronized trigger that is targeted to an ACAP-X application shall be encapsulated as a combination of: 1) DSM-CC U-U BIOP::StreamEventMessage object as defined by MHP [2], Table B.30, that makes use of any form of an interoperable object reference (IOR) as permitted by Section 10.4.1.2, “Interoperable Object References,” and 2) DSM-CC Stream Event Descriptor as defined by Section 10.4.7.4.1, “Stream Event Descriptor.”

The mapping defined in Table 8.5 from the generic trigger event information items described by Section 8.1.4, “Trigger Processing,” shall apply:

Table 8.5 ACAP-X Trigger Event Transport Binding

Trigger Event Item	Mapping
event type	eventName_data field of BIOP::StreamEventMessage object
event time	eventNPT field of stream event descriptor
event payload	privateDataByte[] field of stream event descriptor

The BIOP::StreamEventMessage objects that provide mappings between event ids and event types shall appear in the root application directory of the object carousel.

If multiple ACAP-X applications are delivered in a single object carousel instance and each application is targeted for distinct trigger events, then the root application directories of these applications should be distinct. If they are not distinct, then each application shall receive copies of all triggers for which there is a matching event id value.

9. MONITOR APPLICATION SUPPORT (INFORMATIVE)

For ACAP receivers connected to cable networks, the monitor application, unbound applications and their supporting infrastructure shall be included as defined in OCAP 1.0 [4]. A non-exclusive informative list of the relevant sections of that document includes the following:

- Section 10.2.2.1 Unbound Applications
- Section 10.2.2.3 Application Manager Responsibilities
- Section 10.2.2.4 Application Priority
- Section 10.2.2.5 Host Device Resident Applications
- Section 11.2.2 Extensions to DVB-MHP (Normative)
- Section 13.2.2 Extensions to DVB-MHP (Normative)
- Section 18.2.1 Normative
- Section 19 Baseline Functionality
- Section 20 “Monitor Application”
- Annex G “OCAP 1.0 Application API”
- Annex P “OCAP 1.0 Service API”
- Annex R “Hardware POD API”
- Annex S “Media API”
- Annex Q “OCAP 1.0 System API”
- Annex K “OCAP User Input Event API”

- Annex A “XAIT Document Type Definition”
- Annex H “MPEG Component API”

Note: Unbound applications are explicitly defined only in the context of cable networks as defined by SCTE 90-1. The definition of unbound applications may be extended in the future for the terrestrial environment.

10. TRANSPORT AND SIGNALING

10.1 Introduction

This section of the specification specifies the transport and signaling of applications and application files. The specification is based on the MHP definitions of GEM functional equivalents adopted in this section. After the normative specification, several informative sections (labeled as such) describe the transport design. The scope is just those aspects of the transport protocol that relate to applications.

10.1.1 Notation

The notation used is distinctive to aid the reader in recognizing elements that are the same as they are in referenced standards. These references are typographically distinguished by the use of a different font (e.g., *restricted*), may contain the underscore character (e.g., *sequence_end_code*) and may consist of character strings that are not English words (e.g., *dynrng*). When syntactic elements from ETSI Standards are used the form used therein is retained (e.g., *thisIsString*, or *ThisIsString*). When elements from MPEG, SCTE, or ATSC standards are used the notation form “*sequence_end_code*” is used.

10.2 Carousel

As specified in Section 17.1, “Compliance with GEM,” this standard shall use the MHP definition of the “Carousel” functional equivalent as specified in GEM [1] clause 15.6. This definition is extended with the definitions in this section.

10.2.1 NSAP Address

GEM [1] clause 15.6.1.1.1 enables GEM terminal specifications to define a replacement for the definition of the NSAP address. This standard does so. The *specifierType* and *specifierData* definitions described in Table 10.1 shall replace the *specifierType* and *specifierData* definitions described in GEM [1] clause 15.6.1.1.1.

Table 10.1 Specifier and Service Location

<i>specifierType</i>	// 8 bit uimsbf. The value is 0x1, indicating IEEE OUI
<i>specifierData</i>	// 24 bit uimsbf. The value is 0x000979, indicating ATSC
<i>acap_service_location()</i> {	
<i>source_id</i>	// 16 bit uimsbf. When operating in a cable environment the value 0x0000 has the meaning defined in section 5.3.2 of SCTE 65 [44]; other values resolve to a virtual channel.
<i>reserved</i>	// 64 bit uimsbf, 0xffffffffffff
}	

10.2.2 Content Type and Timestamp Inheritance

GEM [1] clause 15.6.1.1.2 enables GEM terminal specifications to define additional mechanisms for determining the MIME type of a file. This standard does so by defining the content type inheritance mechanism described in this section. This inheritance rule applies also to inheritance of the timestamp value.

The timestamp descriptor is defined in Section 10.2.4, “Time Stamp Descriptor.” The content type descriptor is defined in MHP clause B.2.3.4, as included in this standard through the MHP definition of the “Carousel” functional equivalent as specified in GEM [1] clause 15.6. As with the MHP carousel, if a file message includes a content type descriptor, the type of the file shall be determined from this descriptor.

If a file message does not contain a content type descriptor, then the content type descriptor of the object containing the file shall be used to determine the type of the file. Similarly, if a file message does not include a timestamp descriptor, then the timestamp descriptor of the object containing the file is used to determine the type of the file.

Directory Objects may contain a content type descriptor and/or a timestamp descriptor. If one is present it shall be associated with the directory; otherwise the value associated with the object containing the directory shall be associated with the directory.

Service Gateway objects may contain a content type descriptor and/or a timestamp descriptor. If a content type descriptor is not present, the type “application/dvbj” shall be associated with the service gateway. If timestamp descriptor is not present, then a value of 0 shall be associated with the service gateway.

If an object contains more than one content type descriptor or more than one timestamp descriptor, then it is implementation dependant which one is used. If a file or directory object does not include a content type descriptor or does not contain a timestamp descriptor, and this object is contained within two or more objects, then it is implementation dependant which one is used for the inheritance of the missing descriptor value.

10.2.3 Application Transport Over HTTP

Note: As specified in Section 16, “Detailed Platform Profile Definitions,” the extensions in this section only apply to the profile including ACAP-X.

The Object Carousel definitions of MHP clause B.2.3 (as included in this standard through the MHP definition of the “Carousel” functional equivalent as specified in GEM [1] clause 15.6) are extended with the messages defined in this section.

10.2.3.1 HTTP Profile

HTTPProfileBody is defined below. The Interoperable Object Reference of a File message may include zero or one HTTPProfileBody instance or (see below) zero or one HTTPSPProfileBody instance. If the HTTPProfileBody is present in other object carousel messages, the implementation shall ignore it.

The profile describes the location of the file contents on the interaction channel. The profile contains the components (host, port, and path) which allow the implementation to construct the schema:

- http://host:port/path_segments

The HTTPProfileBody is given in Table 10.2.

Table 10.2 Semantics of the HTTPProfileBody

HTTPProfileBody {	
ProfileIdTag	// 32 bit uimsbf. The value is 0x44564200
ProfileDataLength	// 32 bit uimsbf
ProfileDataByteOrder	// 8 bit uimsbf
VersionMajor	// 8 bit uimsbf
VersionMinor	// 8 bit uimsbf
HostDataLength	// 8 bit uimsbf
for (k=0;k<N1;k++) {	
HostData	// 8 bit uimsbf
}	
Port	// 16 bit uimsbf
ObjectKeyLength	// 16 bit uimsbf
for (k=0;k<N2;k++) {	
ObjectKeyData	// 8 bit uimsbf
}	
}	

The semantics of the profile fields are:

ProfileDataByteOrder – The field shall be 0x00 to indicate big endian order.

VersionMajor – The field indicates the major portion of the protocol version. The implementation shall ignore it, as the implementation must support http1.1. The field is present to anticipate future versions of the protocol.

VersionMinor – The field indicates the minor portion of the protocol version. The implementation shall ignore it, as the implementation must support http1.1. The field is present to anticipate future versions of the protocol.

HostData – The character sequence specifies the host to which the client http messages will be sent. The schema may be either the fully qualified domain name, or the decimal shorthand (e.g “129.145.166.188”). The character encoding is UTF-8.

Port – The field is an unsigned integer that specifies the port at which the service side listens. The value 0xFFFF is reserved to mean to adopt the default port. (The default port for http, specified in RFC 2616, is port 80.)

ObjectKeyData – The field is the character sequence that represents the path that identifies the service side implementation. The character encoding is that described in “Uniform Resource Identifiers (URI): Generic Syntax” (RFC 2396, [29]). The implementation shall support the fragment identifier construct of section 4.1 of RFC 2396 and support the query construct of Section 5.0 and Section 5.2 of RFC 2396.

An HTTPProfileBody IOR may be used in conjunction with a BIOPProfileBody, in which case the priority for determining which IOR entry to use shall be as follows:

- 1) If there is no timestamp descriptor in the File message, then use the HTTPProfileBody to perform an unconditional GET using HTTP, and, if the request is successful, then use the resource returned by the HTTP response. If the request is not successful, then use the resource referenced by the BIOPProfileBody.

- 2) If there is a timestamp descriptor in the File message, then use the HTTPProfileBody to perform a conditional GET using HTTP with the value of the timestamp being used to construct an If-Modified-Since request header, and, if the request is successful and returns a newer resource, then use the resource returned by the HTTP response. If the request is not successful or no resource was returned, then use the resource referenced by the BIOProfileBody. If the timestamp descriptor is present, but the value equates to “unknown”, the logic of the first case is applicable. The semantics are as if the descriptor is not present.
- 3) If no HTTPProfileBody is present, then use the BIOProfileBody to obtain the resource.

A File Message's IOR shall contain no more than one instance of an HTTPProfileBody and no more than one instance of a BIOProfileBody. If Interoperable Object Reference of the File Message contains a LiteOptionsProfileBody, then it shall not contain either a HTTPSPProfileBody or HTTPProfileBody or a BIOProfileBody.

10.2.3.2 HTTPS Profile

The HTTPSPProfileBody is defined below. The profile is comparable to the profile for http, but signals that the protocol is to be https instead. The schema for the HTTPSPProfileBody is identical to that for HTTPProfileBody except for the ProfileIdTag code. See Table 10.3.

Table 10.3 Semantics of the HTTPSPProfileBody

HTTPSPProfileBody {	
ProfileIdTag	// 32 bit uimsbf. The value is <tbid>.
ProfileDataLength	// 32 bit uimsbf
ProfileDataByteOrder	// 8 bit uimsbf
VersionMajor	// 8 bit uimsbf
VersionMinor	// 8 bit uimsbf
HostDataLength	// 8 bit uimsbf
for (k=0;k<N1;k++) {	
HostData	// 8 bit uimsbf
}	
Port	// 16 bit uimsbf
ObjectKeyLength	// 16 bit uimsbf
for (k=0;k<N2;k++) {	
ObjectKeyData	// 8 bit uimsbf
}	
}	

The semantics of the fields are identical to that of the HTTPProfileBody. (The default port for https is 443.) The HTTPSPProfileBody may be used in conjunction with a BIOProfileBody. The timestamp algorithm is identical to the algorithm for HTTPProfileBody.

10.2.4 Time Stamp Descriptor

The Time Stamp Descriptor describes the time when the object was last modified.

Note: The schema is the same as that found in A/95 [59], but this standard relaxes certain semantic restrictions of that specification. Whereas A/95 requires the descriptor to be present in all file messages, this is not necessary to signal a TimeStamp value for each file in a directory hierarchy, due to the Content Type and Time inheritance mechanism.

The schema of the descriptor is:

```
TimeStampDescriptor() {
    DescriptorTag          // 8 bit uimsbf The value shall be 0x8C.
    DescriptorLength      // 8 bit uimsbf
    TimeStamp             //64 bit uimsbf
}
```

The definition for each field is:

Descriptor Tag – The field identifies the descriptor. For the TimeStampDescriptor the value shall be 0x8C.

Descriptor Length – The field describes the byte count of the data that follows the Descriptor Length field it. The value of the field shall be set to 0x08.

Time Stamp – This 64 bit unsigned integer represents the UTC time when the object (or child objects) was last modified. The units are milliseconds since 00:00:00 of January 1, 1970 GMT. The value 0xFFFFFFFFFFFFFFFF shall indicate that the time is not available. For a Service Gateway Object or a Directory Object, the object is “modified” if a binding has been added or deleted, or if a binding name has been changed. For a File Object, the object is “modified” if the ContentTypeDescriptor, the ContentLength, or the ContentData have been changed.

10.2.5 Usage of Private Data for non-ACAP Extensions

In the MHP definition of the “Carousel” functional equivalent as specified in GEM [1] clause 15.6, private data is specified in:

- The Download Info Indication Message in MHP clause B.2.2 (last structure)
- The message schema in the last structure of the Digital Storage Media Command and Control specification [23]

If a non-ACAP extension uses this private data, this standard requires that the first structure of the private data be the Registration Descriptor of the Motion Picture Experts Group. The purpose of the constraint is to eliminate name collisions. The Registration Descriptor provides the mechanism through which organizations reserve unique codes. The organization codes scope the private data found after the structure.

The Registration Descriptor shall adopt the restrictions of Report T3-548 [57] and Report T3-549 [58]. In addition to these conventions, the Format Identifier of the Registration Descriptor is to be registered with the Society of Motion Picture and Television Engineers. The web site for the organization (<http://www.smpte-ra.org>) provides further information on the registration process.

10.2.6 Data Broadcast Descriptor

The value of the data_broadcast_id field of the data_broadcast_descriptor as described in MHP clause 10.7.2 (as included in this standard through the MHP definition of of the “Application Signalling” functional equivalent as specified in GEM [1] clause 15.6) shall be 0x010D.

10.3 Application Signaling

As specified in Section 17.1, “Compliance with GEM,” this standard adopts the MHP definition of the “Application Signalling” functional equivalent as specified in GEM [1] clause 15.6. This definition is extended with the definitions in this section.

10.3.1 Application Content Types

The `application_type` definition of MHP clause 10.4.6 (as included in this standard through the MHP definition of the “Application Signalling” functional equivalent as specified in GEM [1] clause 15.6) is extended with the values given in Table 10.4.

Table 10.4 `application_type` Extensions

<code>application_type</code>	description
0x0006	ACAP-J
0x0007	ACAP-X

Note: As required by GEM, the `application_type` value of 0x0001 for DVB-J is supported. The terminal behavior is the same as for ACAP-J.

10.3.2 Application Protocol ID

GEM [1] clause 15.6.1.2.1 enables GEM terminal specifications to define additional values for the `protocol_id` field. This standard adds the value listed in Table 10.5.

Table 10.5 `protocol_id` Extension

Value	Description
0x0006	ACAP Object Carousel

When the `protocol_id` is 0x0006, the selector bytes in the transport protocol descriptor shall be as follows:

```

ACAPTransportProtocolSelector {
    RemoteConnection          // 1 bit uimsbf.
    ReservedFutureUse        // 7 bit uimsbf. The value shall be all ones.
    If (RemoteConnection == "1") {
        SourceId              // 16 bit uimsbf
        Reserved              // 32 bit uimsbf. The value shall be all ones.
    }
    ComponentTag              // 8 bit uimsbf
}

```

The semantics of each field are:

RemoteConnection – If the value is zero, the current service provides the transport connection. The subsequent fields are not present in this case. If the value is one, a service other than the current service provides the transport connection. Applications with this flag set shall have their application control code set to `REMOTE`, as specified for MHP applications carried by a remote MHP object carousel, as defined in clause 10.8.1.1 of MHP (as included in this standard through the MHP definition of the “Application Signalling” functional equivalent as specified in GEM [1] clause 15.6).

SourceId – Refers to the `source_id` of the transport stream that provides the transport connection.

Note: See Section 10.2.1, “NSAP Address,” of this standard.

ComponentTag – Identifies the “principal” service component that delivers the application. The identified component is the elementary stream that carries the DSI of the object carousel.

Note: The definition of ACAPTransportProtocolSelector is based on the syntax of the selector bytes for OC transport defined in MHP clause 10.8.1.1. It is identical except that sourceId is used instead of the (original_network_id, transport_stream_id, service_id) tuple taken from DVB-SI [19].

10.3.3 Signaling of Profiles and Versions Required by Applications

For applications fully compliant with this version of this standard, the following values shall always be signaled:

application_profile	1
version.major	1
version.minor	0
version.micro	0

10.3.4 ACAP-X Extensions

The descriptor tag values in this section shall have the associated semantics when used in the AIT.

Note: In the ACAP-J profile, no semantics are defined for the descriptors defined in this section. As a consequence, terminals that only support the ACAP-J profile may ignore them if signaled.

10.3.4.1 ACAP-X Application Descriptor

The Application Representation Specific Descriptor sequence may contain a single ACAP-X Application Descriptor for each ACAP-X application. The descriptor contains a sequence of octets that the implementation forwards to the application at application launch. The implementation shall support the descriptor as described in Table 10.6.

Table 10.6 ACAP-X Application Descriptor

Construct	Structure	Field	Restriction	Source	Section
ACAP-X Application Descriptor		Descriptor Tag	If the descriptor is present, the value shall be 0x60. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Descriptor		Descriptor Sequence Length	The value represents the length of the entire descriptor data. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Descriptor	Parameter Sequence	Parameter Sequence	The character sequence is string that is appended to the application initial path as parameters. The encoding shall be UTF-8. It is valid for the string to be null.	ACAP	Transport

10.3.4.2 ACAP-X Application Location Descriptor

The ACAP-X Application Location Descriptor contains information through which the implementation resolves the location of the ACAP-X application. The Application Representation Specific Descriptor sequence may contain a single ACAP-X Application Location Descriptor for each ACAP-X application. The implementation shall support the descriptor schema defined in Table 10.7.

Table 10.7 ACAP-X Application Location Descriptor

Construct	Structure	Field	Restriction	Source	Section
ACAP-X Application Location Descriptor		Descriptor Tag	If the descriptor is present, the value shall be 0x61 The size of the field is one byte.	ACAP	Transport
ACAP-X Application Location Descriptor		Descriptor Length	The value represents the length of the entire descriptor data. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Location Descriptor		Physical Root Length	The value represents the length of the physical root string. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Location Descriptor	Physical Root Character Sequence	Physical Root Character	The variable length field is either empty or contains a UTF-8 encoded string that specifies the path to the root directory of the application. The semantics are transport protocol specific. See below for details	ACAP	Transport
ACAP-X Application Location Descriptor	Initial Path Character Sequence	Initial Path Character	The variable length field shall contain a UTF-8 encoded string that specifies the relative path to either: 1) the ACAP-X Application Metadata Resource (AMR) file, or 2) the ACAP-X initial entity (i.e an XDML Family Document. The path is relative to the application root directory specified in the Physical Root field.	ACAP	Transport

The semantics of the physical root depends on the transport protocol. Table 10.18 lists the feasible values for the ProtocolId field.

If the ProtocolId is the value for the ACAP Object Carousel, that is 0x0006, then the physical root field represents the relative path from the root directory of the object carousel. The implication is that if the physical root string is empty, the physical root for the application is the physical root of the object carousel.

10.3.4.3 ACAP-X Application Boundary Descriptor

The descriptor may be present in the application representation specific descriptor sequence. The descriptor provides a regular expression that defines the data elements that form the application. If the descriptor is not present, the application boundary defaults to the complete set of all content that resides in the transport signaled in the Transport Protocol Descriptor associated with the application. There can be multiple ACAP-X Application Boundary Descriptor instances for the same ACAP-X application. In this case, the equivalent global regular expression is the OR combination (union) of the individual regular expressions. The syntax of the descriptor is given in Table 10.8. The syntax of the regular expression field is defined in Section 9.3.1.4.1 of MHP 1.1 [3].

Table 10.8 ACAP-X Application Boundary Descriptor

Construct	Structure	Field	Restriction	Source	Section
ACAP-X Application Boundary Descriptor		Descriptor Tag	If the descriptor is present, the value shall be 0x62. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Boundary Descriptor		Descriptor Length	The value represents the length of the entire descriptor data. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Boundary Descriptor		Label Length	The value represents the length of the label string. The size of the field is one byte.	ACAP	Transport
ACAP-X Application Boundary Descriptor	Label Character Sequence	Label Character	The variable length field is either empty or contains a UTF-8 encoded string that specifies the label that is associated with the set of data for the regular expression. The label can be used for prefetching in a transport specific manner.	ACAP	Transport
ACAP-X Application Boundary Descriptor	Regular Expression Byte Sequence	Regular Expression Byte	The variable length field shall contain a UTF-8 encoded string that specifies a regular expression. See below for details.	ACAP	Transport

The evaluation of the regular expression determines whether a resource is considered to be in the ACAP-X application's reference scope. The regular expression is subject to the schema and semantics described in the ACAP-X application Section.

10.4 Object Carousel Protocol (Informative)

This section presents an overview of the object carousel protocol. The normative requirements described in this section are normatively incorporated into this standard in other sections including Section 10.2 and in GEM, MHP, and other referenced specifications.

10.4.1 Message Template

ACAP implementations support the message template described in section B.2.3 of MHP [2].

10.4.1.1 Interoperable Object Protocol

The message set of the object carousel builds on the schema of the Broadcast Interoperable Object Protocol. This protocol extends the Interoperable Object Protocol to account for the nature of broadcast.

10.4.1.2 Interoperable Object References

ACAP implementations support Broadcast Interoperable Object References as described in section B.2.3.7 of MHP [2]. The Profile Body is as described in Section B.2.3.7.1 of MHP. The Lite Options are as described in Section B.2.3.7.2 of MHP. The HTTP Profile and HTTPS Profile are as specified in Sections 10.2.3.1, "HTTP Profile," and 10.2.3.2, "HTTPS Profile."

10.4.1.2.1 Network Service Access Point Address

The object carousel is a collection of resources organized as a graph. The root of the graph is the Service Gateway object, which aggregates the content for a specific service domain. To obtain access to the object carousel, the implementation must discover the Network Service Access Point

Address. The Digital Storage Media Command and Control Specification defines the address as shown in Table 10.9.

Table 10.9 Network Service Access Point Address

AFI	Type	Carousel Id	Specifier	Private Data
1-byte	1-byte	4-byte	4-byte	10-byte

The Digital Storage Media Command and Control Specification and MHP place certain restrictions on the fields of the structure; see Table 10.10.

Table 10.10 Network Service Access Point Address Fields

Construct	Structure	Field	Restriction	Source	Section
Carousel Address		Authorization Format Identifier	The field is zero so as to signal that the format is private.	MPEG 1998 [23] DB 1.0 [17]	9.2.1
Carousel Address		Type	The field is zero so as to signal that the address is for another object carousel.	MPEG 1998 [23] DB 1.0 [17]	9.2.1
Carousel Address		Carousel Id	The field designates a specific object carousel.	MPEG 1998 [23] DB 1.0 [17]	9.2.1
Carousel Address		Specifier	The field is 0x01, which signals the presence of an Organization Unique Identifier (OUI) structure. The structure implies the schema of the private data.	MPEG 1998 [23] DB 1.0 [17]	9.2.1
Carousel Address		Private Data	The field is specific to this standard. See below for details.	MPEG 1998 [23] DB 1.0 [17]	9.2.1

The purpose of the Specifier field is to define the schema for the fields that follow. The Specifier itself is:

```
Specifier {
    SpecifierType    // 8 bit uimsbf
    SpecifierData    // 24 bit uimsbf
}
```

The Digital Storage Media Command and Control Specification assigns certain values, given in Table 10.11.

Table 10.11 Specifier Type Assignments

Construct	Field	Value	Definition	Source	Section
Carousel Address	Specifier Type	0x00	ISO 13818-6 Reserved.	DB 1.0 [17]	9.2.1
Carousel Address	Specifier Type	0x01	IEEE Organization Unique Identifier.	DB 1.0 [17]	9.2.1
Carousel Address	Specifier Type	0x02- 0x07	ISO 13818-6 Reserved.	DB 1.0 [17]	9.2.1
Carousel Address	Specifier Type	0x08- 0xFF	The field is, for this standard, the value assigned to the Advanced Television Systems Committee. See below.	DB 1.0 [17] ACAP	9.2.1

The Specifier Type field is required to be 0x01, which requires the data field to be an Organization Unique Identifier (OUI) assignment. This is consistent with MHP, which adopts the

same technique to indicate the structure that follows. The Organization Unique Identifier is a unique code that the Institute of Electrical and Electronics Engineers assigns to organizations.

The values of the Organization Unique Identifier that an ACAP implementation recognizes are the assignments for the Advanced Television Systems Committee and CableLabs. Both values are listed in Table 10.12.

Table 10.12 Organization Unique Identifier Assignments

Construct	Field	Value	Definition	Source	Section
Carousel Address	Specifier Data	0x000979	Advanced Television Systems Committee Organization	ATSC	
Carousel Address	Specifier Data	0x001000	Cable Labs Organization	OCAP 1.0 [4]	
Carousel Address	Specifier Data	0x00015A	Digital Video Broadcast Organization	DB 1.0 [17]	9.2.1

Given the Organization Unique Identifier, the implementation understands how to interpret the last fields of structure, that is, the private data. The schema is illustrated in Table 10.13.

Table 10.13 ACAP Carousel Location

Transport	First 16 Bits		Second 16 Bits		Third 16 Bits	
	Value	Semantics	Value	Semantics	Value	Semantics
Terrestrial	0x0000	Reserved	0xFFFF	Reserved for Future Use	0x0FFF	Reserved for Future Use
Terrestrial	0x0001-0xFFFF	Source Id	0xFFFF	Reserved for Future Use	0x0FFF	Reserved for Future Use
Cable	0x0000	Source Id for Out-of-Band	0xFFFF	Reserved for Future Use	0xFFFF	Reserved for Future Use
Cable	0x0001-0xFFFF	Source Id for In-Band	0xFFFF	Reserved for Future Use	0xFFFF	Reserved for Future Use

The implementation resolves the Source Id to the carousel address. For this standard, the Source Id resolves to a virtual channel. For values between [0x0001, 0x0FFF], the scope is a single transport stream (known through the Transport Stream Id). For values between [0x1000, 0xFFFF], the scope is region specific. There is, for the present, no single universal value space. The network is responsible for managing the value space to avoid collisions.

If the network is terrestrial, the value of zero for Source Id is reserved. If the implementation encounters the value zero for a broadcast network, the implementation can elect to abort the carousel access. The implementation raises an exception for ACAP-J applications. See the ACAP-J application section for details. There is no comparable exception for ACAP-X applications. If the network is cable, a value of zero is valid. This standard defines the value zero to mean that the carousel address is to be found in the out-of-band channel. See SCTE 40 [30] Sections 3.3, 4.2, and 4.3 for definitions of in-band versus out-of-band channels.

10.4.2 Service Gateway Message

The Service Gateway represents the root of the object carousel. ACAP implementations are required to support the Service Gateway Message of Section B.2.3.6 of MHP [2] as required by GEM [1], subject to the extensions defined in Section 10.2, “Carousel.” The resulting requirements are described below.

10.4.2.1 Message Schema

The Message SubHeader structure of the message contains the Object Info structure. The descriptor sequence inside this structure: 1) may include a single label descriptor, and 2) may include a single Time Stamp Descriptor. These descriptors conform to the inheritance rules presented in Section 10.4.5, “File Message.”

The Binding structure of the message also contains the ObjectInfo structure. The descriptor sequence inside this structure: 1) may include a single Content Type Descriptor, and 2) may include a single Time Stamp Descriptor. These descriptors conform to the inheritance rules presented in the File Message portion of this section. If the Content Type Descriptor is present, it matches a companion Content Type Descriptor of the object that the binding references.

Note: While the schema of the Content Type Descriptor is identical to that of MHP, this standard relaxes the semantic constraint that all leaf nodes of the object graph must include the descriptor. See Section 10.4.5, “File Message,” for the inheritance rules. The inheritance rules are backwards-compatible for a carousel where all leaf nodes include the descriptor.

The other constraints of the object carousel design are applicable. See Section B.2.3.5 and Section B.2.3.6 of MHP [2] for details.

10.4.2.2 Message Descriptors

The ObjectInfo structure may contain a single label descriptor and a single Time Stamp Descriptor as described under the File message.

10.4.2.2.1 Label Descriptor

The ObjectInfo structure of the ServiceGateway message may contain a single Label Descriptor as described under the File message.

10.4.2.2.2 Time Stamp Descriptor

The ObjectInfo structure of the ServiceGateway message may contain a single Time Stamp Descriptor as described under the File message.

10.4.3 Directory Message

ACAP implementations support the Directory Message of Section B.2.3.5 of MHP [2], subject to the clarifications and extensions of this section.

10.4.3.1 Message Schema

The Message SubHeader structure of the message contains the Object Info structure. The descriptor sequence inside this structure: 1) may include a single Content Type Descriptor, and 2) may include a single Time Stamp Descriptor. These descriptors conform to the inheritance rules given in Section 10.4.5, “File Message.” If the binding of the object that references the node contain a Content Type Descriptor, the Object Info of the Message SubHeader of the node contains a single Content Type Descriptor. The two descriptor are the same.

The Binding structure of the message also contains the Object Info structure. The descriptor sequence inside this structure: 1) may include a single Content Type Descriptor, and 2) may include a single Time Stamp Descriptor. These descriptors conform to the inheritance rules

presented in Section 10.4.5, “File Message.” If the Content Type Descriptor is present, it matches the companion Content Type Descriptor of the object that the binding references.

The other constraints of the object carousel design are applicable. See MHP [2], Section B.2.3.6 for details.

10.4.4 Message Descriptors

The ObjectInfo structure may contain a single Content Type Descriptor and a single Time Stamp Descriptor as described under the File message.

10.4.4.0.1 Label Descriptor

The ObjectInfo structure of the Directory message may contain a single Label Descriptor as described under the File message.

10.4.4.0.2 Time Stamp Descriptor

The ObjectInfo structure of the Directory message may contain a single Time Stamp Descriptor as described under the File message.

10.4.5 File Message

The discussion of the File Message considers both the message schema and the schema and semantics of its descriptors.

10.4.5.1 Message Schema

ACAP implementations support the File Message of Section B.2.3.3 of MHP [2], subject to the clarifications and extensions of this section. The Message SubHeader structure of the message contains the Object Info structure. The descriptor sequence inside this structure: 1) may include a single Content Type Descriptor, and 2) may include a single Time Stamp Descriptor. These descriptors conform to the inheritance rules presented below. If the binding of the object that references the node contain a Content Type Descriptor, the Object Info of the Message SubHeader of the node contains a single Content Type Descriptor. The two descriptor will be the same in a well-formed stream.

10.4.5.2 Message Descriptors

This section describes the descriptors that can be present within the File Message.

10.4.5.2.1 Content Type Descriptor

The Content Type Descriptor signals the format of the files that constitute the application. There should be at most one such descriptor in the descriptor sequence inside the Object Info of the Message SubHeader. If multiple formats could describe the content, the format found in the descriptor should be the most descriptive.

In MHP, if the descriptor is not present, or is present but the string is not known to the receiver, the receiver attempts to recognize the content through its file extension, as described in MHP [2], Section 11.3.1.6. In ACAP, this is overridden by the file type inheritance mechanism.

10.4.5.2.1.1 Descriptor Schema

MHP [2], Section B.2.3.4, defines the schema as:

```

ContentTypeDescriptor() {
    DescriptorTag           //8 bit uimsbf. The value is 0x72.
    DescriptorLength       //8 bit uimsbf. The length of the character sequence that follows.
    f or (i=0; i<DescriptorLength; i++) {
        ContentTypeDataByte //8 bit uimsbf. The character sequence that represents a MIME type.
    }
}

```

This standard adopts the same schema. The data within the descriptor forms a string the syntax of which is:

```

ContentTypeData = type "/" subtype *(";" parameter)

```

The type field, subtype field, and parameter field are consistent with Section 5.0 of “MultiPurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies” (RFC 2045). The character encoding of the string is UTF-8.

10.4.5.2.1.2 Descriptor Semantics

This standard relaxes the constraint that Object Info inside the Message SubHeader of Files Messages must include the Content Type Descriptor. (To be careful, the reference specifications do allow defaults. The Content Type Descriptor should be present if the File Object does not match the defaults.) This standard allows content attributes to be defined at the root of the object carousel, or at intermediate nodes, rather than just at the leaf nodes. The nodes of the object graph inherit these attributes. The concept is applicable to both the Content Type Descriptor and the Time Stamp Descriptor. This section considers the inheritance rules for the Content Type Descriptor.

When the file types are fully specified in each file object, the inheritance rules of this standard are backward-compatible to the reference specifications: the message conventions of the reference specifications are still valid under the inheritance rules described here. The implementation, however, does not expect the File Objects whose attributes differ from the defaults to contain the Content Type Descriptor. The implementation derives the attributes through inheritance rules.

Figure 10.1 illustrates the inheritance concept. For both object carousels shown, the leaf nodes are file objects. The premise is that for both object carousels, the files are, left to right, are a pair of ACAP-J applications and then a pair of ACAP-X applications.

For the object carousel on the left, the algorithm is that of MHP [2]. The algorithm requires that File Objects whose attributes differ from the defaults include descriptors to override the defaults. The default is that files contain ACAP-J applications. Thus for the object carousel on the left the File Objects that relate to ACAP-X applications include the Content Type Descriptor.

The object carousel on the right is the same object carousel, but the algorithm differs. The object carousel on the right adopts the inheritance rules described below. In the example, the root node declares that files are ACAP-J applications, and the intermediate node above the ACAP-X applications declares that subsequent files are ACAP-X applications. The observation is that the algorithm conserves bandwidth, since just the root of (homogeneous) sub-graphs contain the descriptor.

The inheritance rules are:

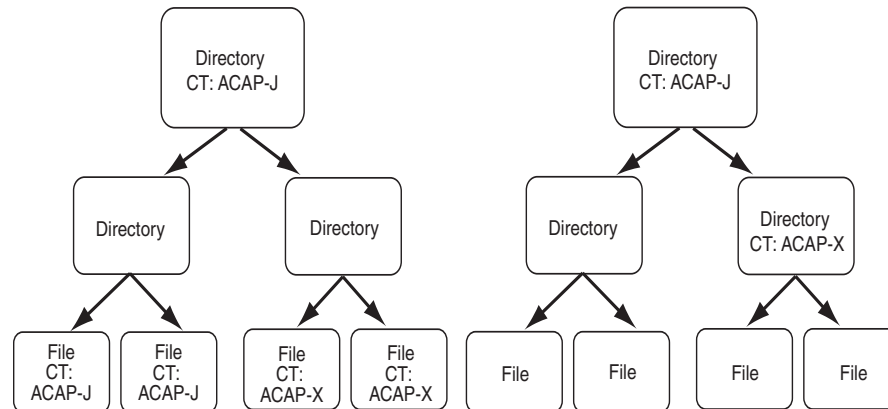


Figure 10.1 Content Type Inheritance.

- The default Content Type Descriptor is “ACAP-J”. Since the default content type is “ACAP-J”, the Service Gateway Object need not contain a Content Type Descriptor. The inheritance rules presume the content type is “ACAP-J” and require the subgraph to include Content Type Descriptors only if the object carousel contains files that differ from the default.
- The Service Gateway Object, any Directory Object along the path to a file, or the file itself may contain a Content Type Descriptor inside the Object Info structure of the Message Subheader. If the traverse to the leaf nodes encounters such a descriptor, the descriptor replaces the descriptor encountered at nodes above the subgraph. The attributes of the descriptor are applicable to all nodes of the subgraph unless the nodes of the subgraph include the descriptor, which then replaces the previous descriptor.
- The result of the inheritance evaluation should be unambiguous. If the carousel structure includes multiple traverse paths to a node, the result of the inheritance evaluation for the node is the same for all traverse sequences. (The implication of the last rule is that certain carousel structures require that file objects that relate to ACAP-J applications must include Content Type Descriptors.)

Figure 10.2 motivates the last rule. For both object carousels, the directory on the left contains both ACAP-J applications and ACAP-X applications. The implication is that, for the object carousel on the left, the traverse sequence through the directory on the left side of the figure evaluates the common file to be an ACAP-J application, since there is no descriptor to override the premise that the files are ACAP-J content. The traverse sequence through the directory on the right side of the figure evaluates the common file to be an ACAP-X application, since the traverse encounters a directory object that declares subsequent files to be ACAP-X content. The distinct traverse sequences reach opposite conclusions.

The object carousel on the right resolves the conflict. (There are other realizations that also resolve the conflict.) The object carousel declares that the common file is an ACAP-X application. The conclusion of the inheritance rule is consistent independent of traverse sequence.

10.4.5.2.2 Time Stamp Descriptor

The Time Stamp Descriptor describes the time at which the object was last modified. The schema is the same as that found in A/95 [59], but this standard relaxes certain semantic restrictions of that specification. The descriptor is defined in Section 10.2.4, “Time Stamp Descriptor.”

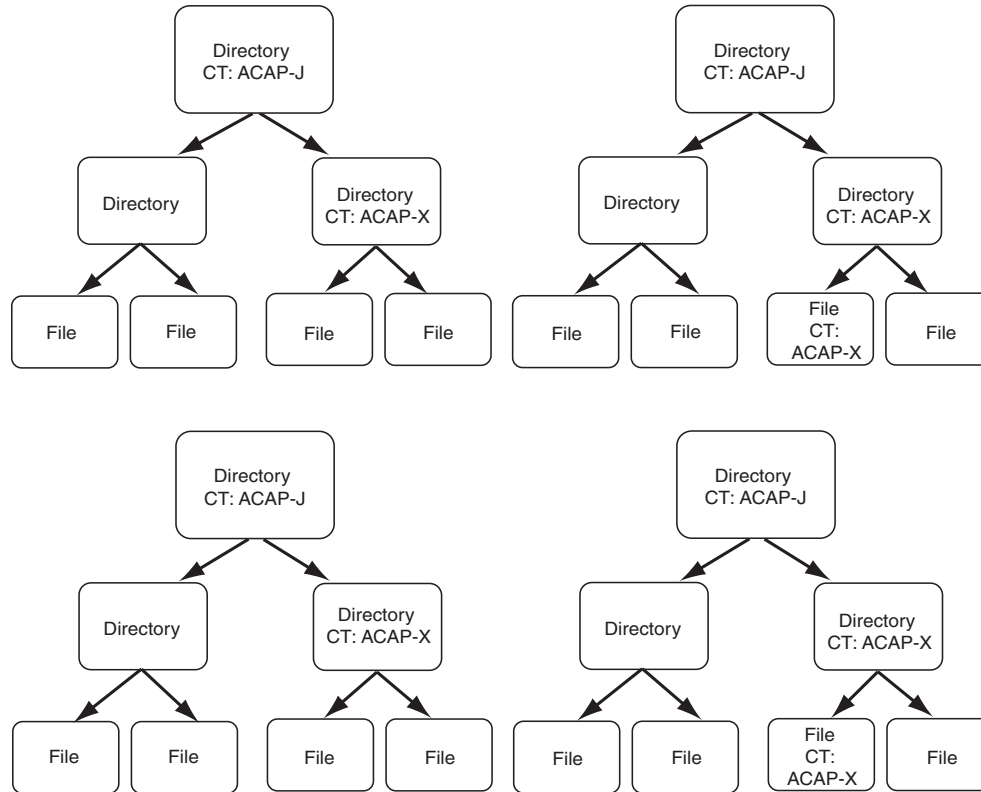


Figure 10.2 Content Type Inheritance conflict.

10.4.5.2.2.1 Descriptor Semantics

The inheritance algorithm for the Time Stamp Descriptor is as described below. The technique is comparable to the technique for the Content Type Descriptor. The inheritance algorithm is:

- The default Time Stamp Descriptor is the special value reserved to mean that the modification time is not available.
- The descriptor may be present in the Message SubHeader structure of a Service Gateway Message, Directory Message, or File Message, subject to the rules below. The Message SubHeader is at most a single descriptor.
- The descriptor may be present in the Binding structure of a Service Gateway Message or Directory Message, in which case the Message SubHeader of the object that the binding references will also include the same descriptor. The Binding should be at most a single descriptor.
- The Service Gateway Object and Directory Object should include the descriptor. In addition to the conditions of the Transport Stream File System specification (the addition of a binding, the deletion of a binding, or the change to a binding name), a modification to an object in the subgraphs below the Service Gateway Object or Directory Object should result in a change to parent descriptors. If the parent descriptor is present, the time value will represent the most recent change to the child objects.
- The File Objects should contain the descriptor. If the descriptor is present, the time value will represent the most recent time at which the ContentTypeDescriptor, ContentLength, or ContentBytes changed.

- The objects of the graph need not include the descriptor. If the descriptor is not present, inheritance rules determine the time stamp. If a child node does not include the descriptor, the node inherits the time stamp of its parent (or the default if the traverse to this point encounters no descriptor). If a child node does include a descriptor, the time stamp becomes the default for the children (if present) of this node as well as the node itself.
- The network is responsible for ensuring that the time stamp is unambiguous. If there are multiple traverse sequences to the same node, all traverse sequences will result in the same time stamp value.

10.4.6 Stream Message

The specification supports two dialects of stream messages. The Stream Object message describes streams that do not also contain stream events. The Stream Event Object message (see below) describes streams that also contain stream events. This section describes the Stream Object.

The implementation supports the Stream Message of Section B.2.3.8 of MHP [2]. The realizable values for the `Info_T:Audio`, `Info_T:Video`, and `Info_T:Data` fields are zero and one. The value zero means such a stream is not present. The value one means such a stream (or multiple streams) is present. If all three fields are zero, the nature of the stream was not known at the time the message was built.

The implementation adopts the algorithm to isolate the elementary stream that contains the object carousel described in Section B.3.1 of MHP [2]. An ACAP implementation will raise an exception for ACAP-J applications. There is no comparable exception for ACAP-X applications.

10.4.7 Stream Event Message

The stream event object is known in other designs as a trigger. The concept is that the platform, upon receipt, forwards the object inside the message to applications that register interest in these events. The application then performs some action.

The object carousel design provides multiple dialects of stream objects. The Stream Message (see above) applies if the stream does not also involve stream events. The Stream Event Message of this section applies if the stream does involve stream events.

10.4.7.1 Stream Event Concepts

The Stream Event feature of MHP [2] supports both stream events that are not time-aware and stream events that are time-aware. For stream events that are not time-aware, the platform forwards the Stream Event Object to the application upon receipt. The application then performs (to be precise initiates) some action. The application initiates the action at once. For stream events that are time-aware, the platform forwards the Stream Event Object at a specific time. These time aware events are also known as synchronous events. The implication of the time aware stream events is that the platform understands some concept of media time. For a more complete discussion of the concepts, see the companion appendix of this standard.

10.4.7.2 Message Schema

The implementation supports the Stream Event Message syntax as described in Section B.2.3.9 of MHP [2]. The realizable values for the `Info_T:Audio`, `Info_T:Video`, and `Info_T:Data` are as described for the Stream Message. The interpretation of the values are as for the Stream Message. The contents of the `EventList_T: Event Names List` can be just the null termination; in other words the event name

can be the empty string. (This is consistent with MHP, which is silent on the question, and thus does not preclude that the event name could be the empty string.)

10.4.7.3 Message Semantics

The scope of this standard is the device that receives the transport stream, that is the edge node that terminates the network. An ACAP implementation supports the semantics of Section B.2.4.1 of MHP [2]. The source nodes and intermediate nodes of the transport chain should, in addition, support the semantics of subsections of B.2.4.1 that relate to these nodes.

10.4.7.4 Message Descriptors

In the case of time aware events, the message requires two descriptors. The first descriptor associates the stream event with a time line. The second descriptor contains time values that allow the device to calculate the time line. This section considers the descriptors.

10.4.7.4.1 Stream Event Descriptor

An ACAP implementation supports the Stream Event Descriptor as described in Section B.2.4.2 of MHP [2].

10.4.7.4.2 NPT Reference Descriptor

Note: The NPT mechanism and scheduled stream events that depend on it are known to be vulnerable to disruption in many digital TV distribution networks. Existing deployed network equipment that re-generates the STC is unlikely to be aware of NPT and hence will not make the necessary corresponding modification to STC values inside NPT reference descriptors. This may cause scheduled stream events to fire at the wrong time or to never fire at all. Applications should only use scheduled stream events where they are confident that the network where they are to be used does not have this problem.

10.5 Data Carousel Protocol (Informative)

Note (informative): As a consequence of the fact that MHP does not use the data carousel as specified in EN 301 192 [17], ACAP does not either.

10.5.1 The Message Template

The data carousel design adopts certain conventions that are applicable to all messages. This section describes the conventions.

10.5.1.1 Message Header

An ACAP implementation supports the Generic Message Header of Sections 8.1 through 8.3 and Sections B.2.2.1 of MHP [2]. The restrictions of Section B.2.6 and B.2.7 of MHP are adopted.

10.5.1.2 Section Format

An ACAP implementation supports the Section Format construct of Section B.2.1 of MHP [2], subject to the clarifications and extensions of this section. The Section Format schema anticipates two error detection techniques. MHP requires support for just one technique, which is the CRC32 algorithm. (The algorithm is described in Annex A of the Motion Pictures Experts Group Systems

specification.) This standard requires support for both techniques. The second technique is the checksum algorithm defined in ISO 13818-6 (1998) Corrigenda 2-2001 (E) and described below:

Algorithm: The scope of the checksum is the entire section. The calculation treats the section as a sequence of 32-bit integers and performs one's complement addition on the entire integer sequence. The calculation begins at the most significant byte, then calculates the one's complement of the result. For the purpose of computing the checksum, the value of the checksum field itself is considered to be zero. If the message length is not a multiple of four bytes, the message is considered to be appended with bytes of 0x0 for the purpose of checksum calculation. If the result of the computation is zero, then the result is set to 0xFFFFFFFF (the alternative value for a one's complement representation of zero). Should a checksum not be desired, the value of checksum is set to '0x00000000' to indicate the checksum has not been calculated. This option is available for networks where the error protection occurs at a different strata of the protocol stack.

MHP [2] (Section B.2.1.1) requires the transport packets to contain at most two sections. This standard does not change this constraint.

10.5.2 Download Info Indication Message

The message specifies information to locate the modules of the carousel. The discussion considers three topics. The first describes the message structure, the second describes the Module Info structure found inside the message, and the third the descriptors that can appear inside the Module Info structure.

10.5.2.1 Message Schema

The Download Info Indication message is described in Section B.2.2.2 of MHP [2]. Section 10.2.5, "Usage of Private Data for non-ACAP Extensions," adds the following additional constraint to this standard. The last structure of the message is private data. This standard requires that, if the private data is used, the first structure of the private data be the Registration Descriptor of the Motion Picture Experts Group. The purpose of the constraint is to eliminate name collisions. The Registration Descriptor provides the mechanism through which organizations reserve unique codes. The organization codes scope the private data found after the structure.

The Registration Descriptor obeys the restrictions of Report T3-548 [57] and Report T3-549 [58]. In addition to these conventions, the Format Identifier of the Registration Descriptor is to be registered with the Society of Motion Picture and Television Engineers. The web site for the organization (<http://www.smpte-ra.org>) provides further information on the registration process.

10.5.2.2 Method Structures

The ModuleInfo structure is described in Section B.2.2.4 of MHP [2].

10.5.2.3 Message Descriptors

The Module Info Structure can contain a descriptor loop. This section describes the descriptors that are to be supported.

10.5.2.3.1 Compressed Module Descriptor

The Compressed Module Descriptor is described in Section 8.2.11 of EN 301 192 [17], and Sections B.2.2.4, and B.2.9 of MHP [2].

The compression technique is the "zib" technique of RFC 1950. If the Compressed Module Descriptor is present, then the data inside the module adopts the structure described in RFC 1951.

While this structure anticipates multiple compression techniques, the MHP specification requires support for just the “deflate” technique. If the code indicates another compression technique, the implementation can elect to abort the carousel assess. The implementation raises an exception for ACAP-J applications. There is no comparable exception for ACAP-X applications.

The Original Size field represents the size before compression, except where the Original Size is not known (or would require decompression on the source node to discover) in which case the value of the Original Size field will be zero.

10.5.2.3.2 Label Descriptor

The Label Descriptor is described in Section B.2.2.4.1 of MHP [2]. The descriptor loop may contain multiple Label Descriptors. These descriptors designate modules that the implementation should prefetch. The information is a hint; the implementation need not prefetch the modules. (See the description of the prefetch descriptor for details.) The character encoding of the character sequence that represents the label is UTF-8. A label with zero characters is valid.

The specification further requires that the label be unique within the object carousel. The implication is that modules to which label refer must reside in the same Download Info Indication message. The label of a Label Descriptor matches the companion label found in a Prefetch Descriptor.

10.5.2.3.3 Caching Priority Descriptor

The Caching Priority Descriptor is described in Section B.2.2.4.2 of MHP [2]. The value is a hint. The implementation can elect to ignore the value.

10.5.3 Download Server Initiate Message

The Download Server Initiate Message is described in Section B.2.2.3 of MHP [2]. The Download Server Initiate message bootstraps the traverse of the object carousel. The message provides the object reference to the Service Gateway object of the carousel. The Service Gateway object represents the root of the carousel structure.

10.5.3.1 Message Schema

The ServerId is the Network Service Access Point (NSAP) Address for the Service Gateway for the object carousel. The address is as specified in Section 10.2.1, “NSAP Address.”

10.5.3.2 Method Structures

The Service Gateway Info Structure is described in Section B.2.2.5 of MHP [2].

10.5.3.3 Group Link Descriptor

The Compressed Module Descriptor is described in Section 8.2.9 of EN 301 192 [17].

10.5.3.3.1 Subgroup Association Descriptor

The Subgroup Association Descriptor is described in Section 8.2.1 of EN 301 192 [17].

10.5.3.4 Download Data Block Message

The Download Data Block message is defined in The Digital Storage Media Command and Control Specification.

10.5.3.5 Download Cancel Message

There are no semantics defined for the Download Cancel message. An implementation may ignore this message if present.

10.5.3.5.1 Message Schema

The message schema is described in Section 7.3.5 of Digital Storage Media Command and Control specification. This definition is further restricted by Section 10.2.5, “Usage of Private Data for non-ACAP Extensions,” in the case where private data is used. The last structure of the message is this private data. This standard requires that the first structure of the private data be the Registration Descriptor of the Motion Picture Experts Group. The purpose of the constraint is to eliminate name collisions. The Registration Descriptor provides the mechanism through which organizations reserve unique codes. The organization codes scope the private data found after the structure.

The Registration Descriptor obeys the restrictions of Report T3-548 [57] and Report T3-549 [58]. In addition to these conventions, the Format Identifier of the Registration Descriptor is to be registered with the Society of Motion Picture and Television Engineers. The web site for the organization (<http://www.smp-te-ra.org>) provides further information on the registration process.

10.5.3.5.2 Message Semantics

The Download Cancel Message contains the block count of the last valid block of the download session. While an implementation is required to accept the message, the response is implementation dependent.

10.6 Transport Protocol (Informative)

10.6.1 Introduction

The object carousel design builds on the data carousel design. The data carousel design in turn builds on certain basic transport protocol tables. The discussion below considers the protocol tables that relate to applications. To be specific this portion of the specification considers the Program Map Table (PMT) and Application Information Table (AIT).

10.6.2 Program Map Table

The Program Map Table provides information about the nature of the applications. The PMT contains the Application Signaling Descriptor, whose presence identifies the elementary stream that contains the AIT. The Application Information Table, the subject of the next section, provides further details about the applications. The applications often require companion data streams. The Program Map Table, in this case, also includes a Data Broadcast Id Descriptor for each data stream.

The implementation supports the descriptors listed in Table 10.14.

Table 10.14 Program Map Table

Table	Construct	Field	Description	Source	Section
Program Map Table	Carousel Id Descriptor		The descriptor is present for object carousels.	MHP 1.0.3 [2]	10.2.2
Program Map Table	Deferred Association Tags Descriptor		The descriptor is present. See Section 9.3.3 of the Data Broadcasting Specification [17] for details.	DB 1.0 [17]	9.9.3
Program Map Table	Elementary Stream Sequence: Generic Descriptors	Elementary Stream: Stream Type	The sequence includes at least one reference to a Program Element that contains an Application Information Table. The Stream Type is 0x05 for such Program Elements.	MHP 1.0.3 [2]	10.1.1 10.2.1
Program Map Table	Elementary Stream Sequence: Generic Descriptors	Elementary Stream: Application Signaling Descriptor	The sequence includes at least one reference to a Program Element that contains an Application Information Table. The value of the Stream Type is 0x05 for such Program Elements.	MHP 1.0.3 [2]	10.1.1 10.2.1
Program Map Table	Elementary Stream Sequence: Generic Descriptors	Elementary Stream: Stream Type (Broadcast Data)	The sequence can contain references to Program Elements that transport data. The field is a transport specific value to signal such Program Elements.	MHP 1.0.3 [2]	10.2.2
Program Map Table	Elementary Stream Sequence: Generic Descriptors	Elementary Stream: Data Broadcast Id Descriptor	If the reference is to a Program Element that transports data, the descriptor can be present.	MHP 1.0.3 [2]	10.2.2

The discussion below considers these descriptors in further detail.

10.6.2.1 Deferred Association Tags Descriptor

As required by MHP, the implementation supports the Deferred Association Tags Descriptor as described in Section 9.3.3 of the Data Broadcast specification [17]. The Deferred Association Tag Descriptor resides in the outer descriptor sequence of the Program Map Table. The last field of the descriptor is the Original Network Id field. The semantics of the field is network-specific. If the network is terrestrial, the field is reserved for future specification. The value will be zero. If the network is cable, the field is again reserved for future specification. The value will be zero. If the network is satellite, the value of the field and its semantics are not addressed by this standard.

10.6.2.2 Carousel Identifier Descriptor

The Carousel Identifier Descriptor is described in MHP [2] Annex B.2.10.1. The descriptor resides in the descriptor loop of the elementary stream entry of the Program Map Table that designates the application's object carousel stream (stream type 0x0B).

Note: It is recommended that AUTOSTART applications use a carousel identifier descriptor with formatid of 0x01, designating *enhanced boot* information.

10.6.2.3 Application Signaling Descriptor

The Application Signaling Descriptor is described in Section 10.7.1 of MHP [2] and Section 10.3, "Application Signaling." The descriptor identifies the Program Element that contains the Application Information Table. (There can be multiple such Program Elements.) The descriptor

may also contain fields that encode the Application Type and the Version Number. If the Application Type field is present, it can be the assignment for ACAP-J applications, DVB-J applications or ACAP-X applications.

If both the Application Type field of the Application Signaling Descriptor and the Content Type Descriptor of the Object Carousel messages is present, the values are to be consistent. Table 10.15 lists valid combinations for this standard. ACAP-J, OCAP-J, and DVB-J are treated as equivalent by ACAP terminals.

Table 10.15 Application Content Types

Application Type	Content Type
ACAP-J	application/acap-j
DVB-J	application/dvbj
ACAP-X	application/acap-x

10.6.2.4 Data Broadcast Id Descriptor

The Data Broadcast Id Descriptor is described in Section 10.7.2.1 of MHP [2].

The presence of this descriptor is optional in application signaling. If the descriptor is present, it is associated with the elementary stream that contains the Object Carousel, not the stream that contains the Application Information Table. This restriction comes from MHP.

If the optional ApplicationType field is present, then it will be one of the values defined by Section 10.6.3, “Application Information Table,” for use with ACAP-J or ACAP-X application types. The behavior of an ACAP terminal device in the case that this field is some other value is implementation-dependent, and may include aborting acquisition of the associated elementary stream and notifying the application through a run-time exception as appropriate.

10.6.3 Application Information Table

The Application Information Table is specified in Section 10.8 of MHP [2] and Section 10.3, “Application Signaling.”

The Application Information Table is described in Section 10.1.1, 10.1.4, 10.4.6, 10.4.7, 10.5.1, 10.5.2, and 10.8.1 of MHP [2] and Section 10.3, “Application Signaling,” of this standard. Table 10.16 provides a brief description of each construct and the section of the reference specification that contains the normative language.

Table 10.16 Application Information Table

Construct	Structure	Field	Description	Source	Section
Application Information Table		Table Id	The value is 0x74 for application information sections.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Section Syntax Indicator	The value is one.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Reserved Future Use	The value is all ones.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Reserved	The value is all ones.	MHP 1.0.3 [2]	10.4.6

Table 10.16 Application Information Table (Continued)

Application Information Table		Section Length	The field represents the section length. The value is less than 1021 (0x3FD).	MHP 1.0.3 [2]	10.4.6
Application Information Table		Test Application Flag	The value can be one, which indicates a test application. The test application is not visible through application interfaces.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Application Type	The value is 0x0001 for GEM/DVB-J/OCAP-J, 0x0006 for ACAP-J and 0x0007 for ACAP-X applications. The device can ignore other applications. See the table below.	ACAP	10.3.1
Application Information Table		Reserved	The value is all ones.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Version Number	The value increments when the contents of the sub-table change.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Current Next Indicator	The value is one.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Section Number	The value of the first section of the sub-table is zero. The value increments for sections with the same Table Id and Application Type.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Last Section Number	The value matches the last section of the sub-table.	MHP 1.0.3 [2]	10.4.6
Application Information Table		Common Descriptor Length	The value is the length of the common descriptor sequence. These descriptors are generic and thus applicable to all applications of the sub-table.	MHP 1.0.3 [2] ACAP	10.4.6
Application Information Table	Generic Application Descriptor Sequence	Common Descriptor	See the description of other Common Descriptors for details.	MHP 1.0.3 [2] ACAP	10.4.6
Application Information Table		Application Specific Descriptor Length	The value represents the length of the application specific descriptor sequence.	MHP 1.0.3 [2]	10.5.1
Application Information Table	Application Specific Descriptor	Application Identifier	The field identifies the application. The schema contains a unique organization code, which then scopes an application code. See the table below for details.	MHP 1.0.3 [2]	10.5.1
Application Information Table	Application Specific Descriptor	Application Control Code	The field encodes the execution state of the application. The values and semantics are application representation specific. The semantics for ACAP-J applications are described in Section 10 of MHP. The semantics for ACAP-X applications are described in the ACAP-X Section of this standard.	ACAP MHP 1.0.3 [2]	10.5.2 10.6.1
Application Information Table		Application Specific Descriptor Length	The length represents the sequence of application representation specific descriptors.	MHP 1.0.3 [2]	10.4.6

Table 10.16 Application Information Table (Continued)

Application Information Table	Application Representation Specific Descriptor Sequence	Application Specific Descriptors	See discussion on Application Specific Descriptors below.	MHP 1.0.3 [2] ACAP	10.1.1
Application Information Table		CRC32	The source node calculates the value. The scope is the entire section. The target node then compares its calculation with this value so as to detect the presence of bits errors.	MHP 1.0.3 [2]	10.4.6

The Application Type field is described in Section 10.4.6 of MHP [2]. This standard adds two assignments. These and the other assignments are given in **Table 10.17**.

Table 10.17 Application Type Assignments

Application Type	Description
0x0000	Reserved For Future Use
0x0001	DVB-J and OCAP-J Application
0x0002	Reserved (DVB-HTML Application)
0x0006	ACAP-J Application
0x0007	ACAP-X Application
0x0008-0x7FFF	Subject to Registration

An ACAP implementation is required to recognize the assignments for the ACAP-J, OCAP-J, DVB-J, and ACAP-X application types. The response for other standard assignments above is implementation-dependent. If the assignment is not standard (i.e., in the above list) the implementation can adopt pragmatics such as examination of file extensions, but the result is implementation dependent.

The Application Identification Structure of the Application Information Table is given in clause 10.5.1 of MHP [2]. The Organization Id field of the structure identifies the organization responsible for the application. The value is required to be registered in TR 101 162. The Application Id field of the structure identifies the application instance. It is the obligation of the organization responsible for the application to manage the value space; the response to duplicate values is implementation dependent.

10.6.3.1 Generic Application Descriptor Sequence

This section considers the generic application descriptors.

10.6.3.1.1 Transport Protocol Descriptor

The Transport Protocol Descriptor is described in Section 10.8.1 of MHP [2] and Section 10.3.2, “Application Protocol ID.”

10.6.3.1.1.1 Descriptor Schema

The schema is described in Section 10.8.1 of MHP [2]. The assignments for certain fields within the descriptor are given in this standard.

Table 10.18 lists the registered codes. An ACAP implementation is required to support the “ACAP Object Carousel” protocol of this section. If the code is a different value, and the implementation does not recognize or does not support the protocol, the implementation can elect

to abort the carousel access. The implementation raises an exception for ACAP-J applications. There is no companion exception for ACAP-X applications.

Table 10.18 Protocol Id Assignments

Value	Description
0x0000	Reserved For Future Use
0x0001	Reserved (DVB Object Carousel)
0x0002	Reserved (DVB Protocol Encapsulation)
0x0003-0x0005	Reserved
0x0007	Reversed For Future Use
0x0100	ACAP Object Carousel
0x0101-0xFFFF	Subject to Registration in TR 101 162 [32]

10.6.3.1.1.2 Object Carousel Selector Structure

The selector field of the Transport Protocol Descriptor is given in Section 10.3.2, “Application Protocol ID.”

10.6.3.2 Download Info Indication Location Descriptor

The Download Info Indication Descriptor is described in Section 10.8.3.3 of MHP [2].

The Transport Protocol Label field identifies the Transport Protocol Descriptor. The Protocol Id field of the Transport Protocol Descriptor to which the label refers is the assignment for the “ACAP Object Carousel” protocol.

10.6.4 Application Specific Descriptor Sequence

The second descriptor sequence that can be present is the application specific descriptor sequence. These application descriptors share a common schema, but the values and their semantics are specific to the application instance.

10.6.4.1 Application Descriptor

The Application Specific Descriptor sequence of the Application Information Table contains at least one Application Descriptor instance. The schema of the descriptor is given in Section 10.7.3 of MHP [2]. The Transport Protocol Label field within the descriptor identifies the Transport Protocol Descriptor. The Protocol Id field of the Transport Protocol Descriptor to which the label refers is the assignment for the “ACAP Object Carousel” protocol.

10.6.4.2 Application Name Descriptor

The Application Specific Descriptor sequence of the Application Information Table contains one or more Application Descriptor Name instances. The Application Name Descriptor is described in Section 10.7.4 of MHP [2].

10.6.4.3 Application Icon Descriptor

The Application Specific Descriptor sequence of the Application Information Table can contain zero or one Application Icon Descriptor instance. The structure associates icons with the application. The implementation supports the Application Icon Descriptor as described in Section 10.7.4 of MHP [2].

10.6.4.4 Prefetch Descriptor

The Application Specific Descriptor sequence of the Application Information Table can include a single Prefetch Descriptor. The purpose of the descriptor is to alert the implementation about which application resources are time critical. The Prefetch Descriptor is described in Section 10.3.3 of MHP [2].

10.6.4.5 Download Info Indication Location Descriptor

The Application Specific Descriptor sequence of the Application Information Table can contain zero or one Download Info Indication Location Descriptor as described in Section 10.8.3.3 of MHP [2].

10.6.5 Application Representation Specific Descriptor Sequences

In addition to descriptors that are specific to application instances, there are descriptors which are specific to the application representation. This section of the specification considers these descriptors.

10.6.5.1 ACAP-J Application Descriptors

This section considers those descriptors that are specific to ACAP-J applications.

10.6.5.1.1 ACAP-J Application Descriptor

The Application Representation Specific Descriptor sequence includes a single ACAP-J Application Descriptor for each ACAP-J application. It is specified in MHP clause 10.9.1 [2], and for this standard applies to ACAP-J/DVB-J applications. The descriptor contains a sequence of octets that the implementation forwards to the application at application launch. The descriptor schema is detailed in Table 10.19.

Table 10.19 ACAP-J Application Descriptor

Construct	Structure	Field	Restriction	Source	Section
ACAP-J Application Descriptor		Descriptor Tag		MHP 1.0.3 [2]	10.9.1
ACAP-J Application Descriptor		Descriptor Sequence Length	The value represents the length of the entire descriptor data.	MHP 1.0.3 [2]	10.9.1
ACAP-J Application Descriptor	Descriptor Sequence	Parameter Sequence Length	The value represents the length of the parameter data.	MHP 1.0.3 [2]	10.9.1
ACAP-J Application Descriptor	Descriptor Sequence	Parameter Sequence	The sequence is initialization data that the receiver forwards at application launch.	MHP 1.0.3 [2]	10.9.1

Note: The schema of the descriptor is identical to the schema of the MHP-J Application Descriptor found in MHP [2].

10.6.5.1.2 ACAP-J Application Location Descriptor

The ACAP-J Application Location Descriptor contains information through which the implementation resolves the location of the ACAP-J application. The Application Representation Specific Descriptor sequence contains a single ACAP-J Application Location Descriptor for each

ACAP-J application. The schema is specified in MHP clause 10.9.2 [2] and is summarized in Table 10.20.

Table 10.20 ACAP-J Application Location Descriptor

Construct	Structure	Field	Restriction	Source	Section
ACAP-J Application Location Descriptor		Descriptor Tag		MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor		Descriptor Sequence Length	The value represents the length of the entire descriptor data.	MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor		Base Directory Length	The value represents the length of the base directory character sequence. The base directory constitutes the first directory in the class path. See Section 10.9.2 of MHP for details.	MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor	Base Directory	Base Directory Character Sequence	The character sequence that represents the base directory.	MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor		Classpath Extension Character Sequence Length	The value represents the length of the classpath extension character sequence. The sequence specifies alternative locations for the classpath. See Section 10.9.2 of MHP for details.	MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor	Classpath Extension Sequence	Classpath Extension	The string that specifies the alternative locations for the classpath.	MHP 1.0.3 [2]	10.9.2
ACAP-J Application Location Descriptor	Initial Class File	Initial Class Bytes	String specifying name of object in filesystem that is the class implementing the Xlet interface	MHP 1.0.3 [2]	10.9.2

10.6.5.2 ACAP-X Application Descriptors

This section considers those descriptors that are specific to ACAP-X applications.

10.6.5.2.1 ACAP-X Application Descriptor

The Application Representation Specific Descriptor sequence may contain a single ACAP-X Application Descriptor for each ACAP-X application. The descriptor contains a sequence of octets that the implementation forwards to the application at application launch. The descriptor is specified in Section 10.3.4.1, “ACAP-X Application Descriptor.”

10.6.5.2.2 ACAP-X Application Location Descriptor

The ACAP-X Application Location Descriptor contains information through which the implementation resolves the location of the ACAP-X application. The Application Representation Specific Descriptor sequence may contain a single ACAP-X Application Location Descriptor for each ACAP-X application. The descriptor is specified in Section 10.3.4.2, “ACAP-X Application Location Descriptor”.

The semantics of the physical root depends on the transport protocol. Table 10.18 lists the feasible values for the ProtocolId field.

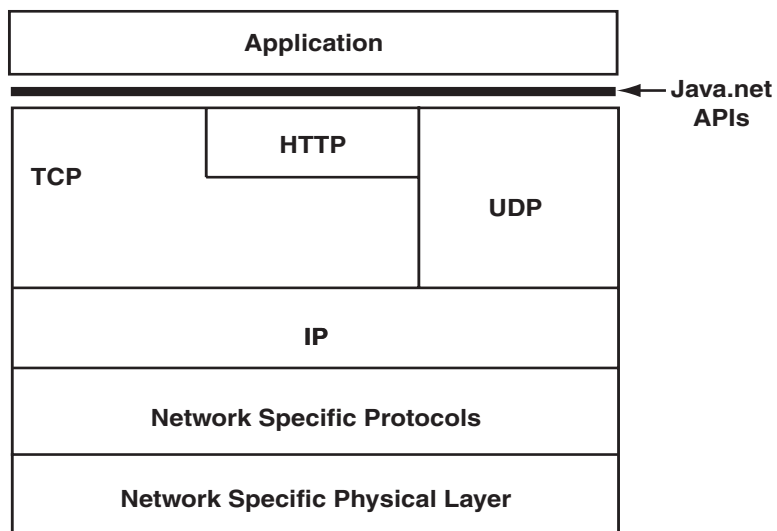


Figure 11.1 Interaction channel network protocols.

If the ProtocolId is the value for the ACAP Object Carousel, that is 0x0006, then the physical root field represents the relative path from the root directory of the object carousel. The implication is that if the physical root string is empty, the physical root for the application is the physical root of the object carousel.

10.6.5.2.3 ACAP-X Application Boundary Descriptor

The descriptor may be present in the application representation specific descriptor sequence. The descriptor provides a regular expression that defines the data elements that form the application. If the descriptor is not present, the application boundary defaults to the complete set of all content that resides in the transport signaled in the Transport Protocol Descriptor associated with the application. There can be multiple ACAP-X Application Boundary Descriptor instances for the same ACAP-X application. In this case, the equivalent global regular expression is the OR combination (union) of the individual regular expressions. The syntax of the descriptor is given in Section 10.3.4.3, “ACAP-X Application Boundary Descriptor.”

The evaluation of the regular expression determines whether a resource is considered to be in the ACAP-X application’s reference scope. The regular expression is subject to the schema and semantics described in the ACAP-X application section.

11. INTERACTION CHANNEL

11.1 Interaction Channel Protocols

This section describes the interaction channel protocols required in an ACAP device for use by an application.

Figure 11.1 illustrates the network protocols used for the interaction channel.

11.1.1 Network Specific Protocols

A wide range of network protocols defined by standards such as DOCSIS, DAVIC, POD Module, PSTN, Ethernet, and PPP or proprietary methods may be used to provide the interconnectivity

between an ACAP device and a server. All the necessary protocols associated with each network specific protocols shall be supported by the ACAP device.

11.1.2 Internet Protocol

An ACAP device shall support IP as described in Section 6.3.2 of GEM [1].

11.1.3 User Datagram Protocol (UDP)

An ACAP device shall support UDP as described in Section 6.3.9 of GEM [1].

11.1.4 Transmission Control Protocol (TCP)

An ACAP device shall support TCP as described in Section 6.3.3 of GEM [1].

11.1.5 Hyper-Text Transfer Protocol (HTTP)

In an ACAP device where the ACAP-X environment is present, HTTP 1.1 protocol shall be supported as defined in the RFC 2616 [39] with constraints and modifications defined in A/96 [40].

11.1.6 Domain Name Service (DNS)

An ACAP device shall support DNS as described in Section 6.3.10 of GEM [1].

12. SECURITY

Chapter 12 of GEM [1] shall apply with the clarifications and extensions detailed in the following sections.

12.1 Introduction

The ACAP security model is fully conformant to the GEM security model. It addresses the same areas of security; i.e. authentication of broadcast applications, security policies for applications, security over the interaction channel, and certificate management.

The ACAP security model specifies an alternate to the GEM security model in a way that takes into account the requirements of the ACAP terrestrial environment, of the ACAP cable environment, and the policy access to functionalities not specified in GEM [1] which are exposed to ACAP applications. As such,

- Section 14.2, “ACAP Trust Model,” specifies extensions to the GEM trust model for ACAP terminals as allowed by GEM [1] Section 12.1.3.
- Section 14.4.1, “ACAP Signing Framework,” specifies a modified signing framework as allowed by GEM [1] Section 12.1.3.
- Section 14.4.2, “ACAP Extensions to Security Policies for Applications,” specifies the syntax and semantics of the additional ACAP permissions in a new permission request file as allowed by GEM [1] Section 12.6.

12.2 ACAP Trust Model

12.2.1 General Rules

Applications that are eligible to be trusted shall be identified with an `application_id` from the signed applications range as defined by MHP [2], Table 12. Applications that are not eligible to be trusted shall be identified with an `application_id` from the unsigned applications range. An

application with an `application_id` from the unsigned applications range is treated as not eligible to be trusted even if the files might be transmitted with signatures.

ACAP terminals shall not grant any access rights to resources outside the sandbox to ACAP applications that have not requested the appropriate permissions through a GEM or an ACAP permission request file and that are not eligible to be trusted. Other criteria for deciding whether or not to grant access to resources outside the sandbox and that are intentionally not specified in the present document may subsequently apply such as the user own policies.

Note: See the definition of “Trusted Application” in Section 3, “Definitions and Abbreviations.”

12.2.2 Applications Received Over a Terrestrial Interface

Codesigning of applications received over a terrestrial interface is not required in order to establish that the application is eligible to be trusted, and therefore may be granted the right to access resources outside the sandbox. ACAP terminals that receive such an application are allowed to ignore any security files apart from the GEM or ACAP permission request files (see Section 12.4.2.1, “ACAP Permission Request File,” that could be present along the application before establishing that the application is eligible to be trusted. By default, an unsigned application received over a terrestrial interface will therefore be considered as eligible to be trusted.

12.2.3 Applications Received Over a Cable Interface

Codesigning of applications received over a cable interface is required in order to establish that the application is eligible to be trusted and therefore may be permitted to access resources beyond those granted to unsigned applications. *Note:* This concept is also known as “outside the sandbox”.

Either the GEM Signing Framework as defined in GEM [1] Section 12.6 or the ACAP Signing Framework as defined in Section 12.4.1, “ACAP Signing Framework,” shall be used for content signing.

12.3 Security Policy for Applications

For clarification, GEM [1] Section 12.6 shall apply.

Note: Attention is drawn to the second paragraph of GEM [1] Section 12.6 where the interpretation of the terms “unsigned applications” and “signed applications” is clarified in the context of a GEM terminal specification where code signing is not required to establish trust, which is the case in an ACAP terrestrial environment. See Section 12.2, “ACAP Trust Model.”

Additionally, according to GEM [1] Section 12.6, an ACAP terminal is required to be able to operate in a mode where it grants permission to provide access to all of the functionality required by the profiles and options that it supports when appropriately requested (e.g. via the GEM or ACAP permission request files). The mechanism for causing the terminal to operate in this mode is implementation-dependent. The granting of permissions for accessing functionality outside of the claimed ACAP profile and options is not required.

Note: Broadcasters should be aware that if they choose not to sign applications requiring access to privileged operations, then there is no guarantee that those

applications be signed at some point in the cable distribution chain and thus become eligible to be granted access to privileged operations when executed within a cable environment.

12.4 ACAP Extensions to GEM Security Model

12.4.1 ACAP Signing Framework

12.4.1.1 General Principles

The ACAP Signing Framework for ACAP applications exists in addition to the existing GEM Signing Framework. It shall be based on the Signing Framework specified in GEM [1] Section 12 with the following modifications:

- The name of the hashfile shall be `acap.hashfile`
- The content (MIME media) type label of the hashfile shall be `application/acap-digest`.
- The name of the signature files shall be `acap.signaturefile.<x>`, where `<x>` is a string that distinguishes between several possible signature files. Apart from that deviation, the rules with respect to the format of `<x>` shall conform to the rules expressed in GEM [1] Section 12.4.
- The content (MIME media) type label of a signature file shall be `application/acap-signature`.
- The name of a certificate files shall be `acap.certificates.<x>`, where `<x>` is identical to the extension of the signature filename that is authenticated by the ACAP certificate chain in this file. Apart from that deviation, the rules with respect to the format of `<x>` shall conform to the rules expressed in GEM [1] Section 12.4.
- The content (MIME media) type label of a certificate file shall be `application/acap-certificate`.
- The profile of X.509 certificates for authentication of applications shall be conformant to OCAP 1.0 [4] Section 14.2.1.6.
- The permission request file to be used shall be the one specified in Section 12.4.2.1, "ACAP Permission Request File."
- The description of the content of the signature file shall conform to GEM [1] Section 12.4 as modified by OCAP 1.0 [4] Section 14.2.1.24.
- ACAP terminals shall conform to OCAP 1.0 [4] Section 14.2.1.23.

and additions:

- The ACAP signing framework specifies the way ACAP-X applications are authenticated. See Section 12.4.1.2, "Authentication of ACAP-X Applications."

An ACAP terminal shall support both the GEM Signing Framework as specified in GEM [1] Section 12 and the ACAP Signing Framework as specified in this section.

ACAP applications should only include those security files defined in either GEM [1] or those security files defined in this standard and prefixed by «acap.». In the case that files from both models are included, the GEM security files shall not be used to authenticate the application but shall be listed in the appropriate ACAP hash file for the directory in which they occur.

12.4.1.2 Authentication of ACAP-X Applications

Authentication of ACAP-X applications shall be performed in the same way than for ACAP-J applications. If signed, an ACAP-X application shall follow either the GEM [1] signing framework or the ACAP signing framework as specified in the present document.

As such, non-authenticated ACAP-X applications will operate within a sandbox environment. Authenticated ACAP-X applications associated with a permission request file may be granted permissions outside the sandbox.

Note: The GEM [1] security model is independent of the type of the application. However, since GEM [1] only fully specifies the procedural environment, it is clarified here that ACAP-X applications must follow the same authentication process.

12.4.2 ACAP Extensions to Security Policies for Applications

12.4.2.1 ACAP Permission Request File

12.4.2.1.1 General Principles

An ACAP terminal shall support both the GEM [1] Permission Request File (PRF) Document Type Definition (DTD) and an extended PRF DTD called the ACAP Permission Request File DTD defined in Annex B Section 2, “ACAP Permission Request File Document Type,” as allowed by GEM [1] Section 12.6.

If both a GEM Permission Request File and an ACAP Permission Request File are in the same directory as the initial file of the ACAP application, then the GEM Permission Request File shall be ignored.

The returnchannel element and, in particular the phonenumbers element defined in GEM [1] may not have corresponding semantics in a cable environment. For clarification, an ACAP Cable-only implementation is required to interpret the presence of the phonenumbers element. It is however not required to process it.

The content (MIME media) type label of a permission request file shall be application/acap-permission.

The cookie, runtime code extension, and Java bridge permissions defined in Sections 12.4.2.3.1, “Cookie Permission,” 12.4.2.3.2, “Runtime Code Extension Permission,” and 12.4.2.3.3, “Inter-Environment Bridge Permission,” are only meaningful for terminals supporting the ACAP-X environment. An ACAP terminal not supporting an ACAP-X environment is required to interpret the presence of such permissions in an ACAP Permission Request File. It is however not required to process them.

12.4.2.1.2 DTD definition

The ACAP Permission Request File (PRF) DTD extends the GEM [1] Permission Request File DTD since additional permissions have been added to meet ACAP requirements. However, in order to be conformant with GEM [1], the ACAP PRF DTD includes all elements and attributes of the GEM [1] PRF DTD.

The following Formal Public Identifier (FPI) shall be used to identify the ACAP PRF DTD:

```
"-//ATSC//DTD ACAP Permission Request File 1.0//EN"
```

and the following URL for the SystemLiteral may be used to reference this file:

```
http://www.org.atsc/acap/dtd/acap-permission-1.dtd
```

The Name used in the document type declaration shall be “permissionrequestfile”.

The ACAP PRF DTD is provided in Annex B.

12.4.2.1.3 ACAP Permission Request File Name and Location

The format for the ACAP Permission Request File name shall be ‘acap.’<application name>.’perm’.

The prefix “acap” identifies this as a well known file defined by this standard. The portion “application name” carries the file name of the initial file of the application excluding any file name extension or suffix. The initial file depends on the application type as is shown in Table 12.1 for the types defined in this standard.

Table 12.1 Application Name for Different Application Types

Application Type		Path from which File Name Shall be Extracted
Value	Meaning	
0x0006	ACAP-J	The name initial_class_byte, see Section 10.6.5.1.2, “ACAP-J Application Location Descriptor”
0x0007	ACAP-X	The name initial_path_bytes, see Section 10.6.5.22, “ACAP-X Application Location Descriptor”

The ACAP permission request file shall be located in the same directory as the initial file.

12.4.2.2 Cable Specific Security Access Policy

In this section, the listed features are only accessible in a cable environment and therefore, can only be accessed by applications that have been signed using either the GEM [1] or the ACAP Signing Framework.

12.4.2.2.1 Monitor Application Features Access Policy

A Monitor Application permission can provide a set of permissions typically required by an OCAP monitor application as specified in Section 9, “Monitor Application Support (Informative).” Multiple instances of the ocap:monitorapplication element may appear, one for each type of permission that is requested. The following access policy is applied to Monitor Application Permissions:

12.4.2.2.1.1 Applications not Signed by the ACAP Signing Framework

An application not signed by the ACAP Signing Framework may not use any Monitor Application Features.

12.4.2.2.1.2 Applications Signed by the ACAP Signing Framework

By default, an application signed by the ACAP Signing Framework may not use any Monitor Application capabilities. However, the right to exercise specific Monitor Application capabilities can be requested with the Monitor Application Permission that can be put in the ACAP Permission Request File.

12.4.2.2.1.3 Privileged Monitor Application API access

This section shall conform to Section 14.2.2.2 of OCAP 1.0 [4].

12.4.2.3 ACAP Security Policy for Applications

12.4.2.3.1 Cookie Permission

12.4.2.3.1.1 Untrusted Applications

Untrusted applications have no access to cookie information items.

12.4.2.3.1.2 Trusted Applications

A trusted application has by default no access to cookie information items, unless otherwise requested by the Permission Request File and granted by the ACAP terminal.

12.4.2.3.1.3 Permission Request Syntax

```
<ELEMENT %acap.cookie.qname; EMPTY>
<!ATTLIST %acap.cookie.qname;
    %acap.target.qname;           CDATA    #REQUIRED
    %acap.actions.qname;         CDATA    #REQUIRED
    %acap.xmlns.attrib;
>
```

The target attribute shall be specified as a URI to indicate the cookie's domain and path. A special target value of "*" shall be used to specify any cookie.

The actions attribute shall consist of one or more of the following tokens: create, delete, read, and write. Multiple actions may be specified in a comma-separated list with optional intervening whitespace. If multiple actions are requested, then all requested actions shall be granted for any requested action to be granted; i.e., if some requested action is denied, then all requested actions shall be denied.

An `acap:cookie` element may appear as a child of the `permissionrequestfile` element of an ACAP application's permission request file.

12.4.2.3.2 Runtime Code Extension Permission

The following ECMAScript and DOM related operations shall be construed as privileged runtime code extension operations:

- `Global.eval()`
- `Function.[[constructor]]`
- `Window.setTimeout()`

Any function or property which permits the creation or mutation of an intrinsic event attribute

Any function or property which permits the creation or mutation of a script element

Notes:

- 1) The `Global.eval()` operation refers to the `eval()` method on the ECMAScript Global Object.
- 2) The `Function.[[constructor]]` operation refers to the internal `[[constructor]]` method on the ECMAScript Function Object.
- 3) A function or property that permits the creation or mutation of an intrinsic event attribute or a script element is considered to be a privileged operation only when it is attempting to create or

mutate an intrinsic event attribute or script element; i.e., if the function or property is used to mutate or create other attributes or elements, then it is not considered a privileged operation.

- 4) The legacy (DOM-0) methods `HTMLDocument::write` and `HTMLDocument::writeln` are not included in the above list since they are not supported by this standard.

12.4.2.3.2.1 Untrusted Applications

Untrusted applications have no access to runtime code extensions.

12.4.2.3.2.2 Trusted Applications

A trusted application has by default no access to runtime code extensions, unless otherwise requested by the Permission Request File and granted by the ACAP terminal.

12.4.2.3.2.3 Permission Request Syntax

```
<ELEMENT %acap.rce.qname; EMPTY>
<!ATTLIST %acap.rce.qname;
    %acap.rce.value.qname;          (true|false) #REQUIRED
    %acap.xmlns.attrib;
>
```

An `acap:rce` element may appear as a child of the `permissionrequestfile` element of an ACAP Application's permission request file.

12.4.2.3.3 Inter-Environment Bridge Permission

In order to make use of the Inter-Environment Bridge as defined in Section 8.2.11.2.4, “Inter-Environment Bridge,” an ACAP application shall request an appropriate permission as defined in this section.

12.4.2.3.3.1 Untrusted Applications

Untrusted applications have no access to the Inter-Environment bridge.

12.4.2.3.3.2 Trusted Applications

A trusted application has by default no access to the Inter-Environment bridge, unless otherwise requested by the Permission Request File and granted by the ACAP terminal.

12.4.2.3.3.3 Permission Request Syntax

```
<ELEMENT %acap.bridge.qname; EMPTY>
<!ATTLIST %acap.bridge.qname;
    %acap.bridge.value.qname;      (true|false) #REQUIRED
    %acap.xmlns.attrib;
>
```

An `acap:bridge` element may appear as a child of the `permissionrequestfile` element of an ACAP Application's Permission Request File.

12.5 Security over the Interaction Channel

Note: In contrast to the OCAP 1.0 X.509 certificate profile as used for broadcast application authentication, GEM [1] Section 12.10 requires the use of the PKIX profile as mandated by TLS 1.0.

12.6 Platform Minima

GEM [1] Section 12.12 is extended with the following:

- An ACAP platform hardware is required to support at least 4 root certificates in order to support the ACAP security model
- The key lengths that an ACAP terminal is required to support is specified in the OpenCable Security Specification [12].

12.7 ACAP Security Operational Model

The ACAP Security Operational model which defines the operational procedures in order to implement in an end-to-end way the ACAP Security Framework is outside the scope of this document.

Note: This Security Operational model should include the creation, delivery and management of root certificates, the creation of application codesigning certificates and the procedures for issuing and managing them.

13. GRAPHICS REFERENCE MODEL

Section 13 of GEM [1] shall apply.

14. SYSTEM INTEGRATION

14.1 Void

14.2 Resource Reference and Locators

14.2.1 ACAP URI Scheme

14.2.1.1 Scheme Definition

This section defines the ACAP URI scheme. The format of this shown in an informal notation is as follows.

Note: Some of the terms in these definitions have the same names as fields in standardized MPEG-2 tables and other data structures used in television. This does not imply any normative relationship between the term and any such field. All normative relationships between terms in the definition and fields in standardized MPEG-2 tables are explicitly defined below, for example in the tables found in Section 14.2.1.3.4, “Resolution of Locator Elements.”

```
ocap://<source_id>[.<stream_type>[,<ISO_639_language_code>]{&
<stream_type>[,<ISO_639_language_code>}}] [<event_id>]{/<path_segments>}
```

```
ocap://<source_id>[.<stream_type>[,<index>]{&<stream_type>[,<index>}}] [<event_id>]{/
<path_segments>}
```

```
ocap://<source_id>[.<PID>{&<PID>}}] [<event_id>]{/<path_segments>}
```

```

ocap://<source_id>[. $<component_name>{&<component_name>}] [;<event_id>] {/
<path_segments>}

ocap://n=<service_name>[.<stream_type>[, <ISO_639_language_code>]{&
<stream_type>[, <ISO_639_language_code>}] [;<event_id>] {/ <path_segments>}

ocap://n=<service_name>[.<stream_type>[, <index>]{&<stream_type>[, <index>]}}
[;<event_id>] {/ <path_segments>}

ocap://n=<service_name>[.+<PID>{&<PID>}] [;<event_id>] {/ <path_segments>}

ocap://n=<service_name>[. $<component_name>{&<component_name>}] [;<event_id>] {/
<path_segments>}

ocap://f=<frequency>.<program_number>[.<stream_type>[, <ISO_639_language_code>]{&
<stream_type>[, <ISO_639_language_code>}] [;<event_id>] {/ <path_segments>}

ocap://f=<frequency>.<program_number>[.<stream_type>[, <index>]{&<stream_type>[,
<index>]}} [;<event_id>] {/ <path_segments>}

ocap://f=<frequency>.<program_number>[.+<PID>{&<PID>}] [;<event_id>] {/ <path_segments>}

ocap://f=<frequency>.<program_number>[. $<component_name>{&<component_name>}] [;<
event_id>] {/ <path_segments>}

ocap:/ <path_segments>

```

A formal specification is expressed in BNF as used in IETF RFC 2396 [29]:

```

acap_uri = acap_scheme ":" acap_hier_part
acap_scheme = "ocap"
acap_hier_part = acap_net_path | acap_abs_path
                (see restriction 1 below)
acap_net_path = "/" acap_entity [ acap_abs_path ]
                (see restriction 2 below)
acap_entity = acap_service | acap_service_component
acap_service = source_id | service_name | acap_program
acap_service_component = acap_service [ "." program_elements ] [ ";" event_id ]
program_elements = language_elements | index_elements | PID_elements |
component_elements
language_elements = stream_type [ "," ISO_639_language_code ] * ( "&" stream_type [ ","
ISO_639_language_code ] )
                (see restriction 3 below)
index_elements = stream_type [ "," index ] * ( "&" stream_type [ "," index ] )
PID_elements = "+" PID * ( "&" PID )
component_elements = "$" component_name * ( "&" component_name )
acap_program = "f=" frequency "." program_number
service_name = "n=" 1* (unreserved_not_dot | escaped)
                (see restriction 4 below)
source_id= hex_string

```

```

component_name = 1* (unreserved | escaped)
                (see restriction 5 below)

frequency = hex_string

program_number = hex_string

stream_type = hex_string

ISO_639_language_code = alpha alpha alpha
                        (see restriction 6 below)

index = hex_string

PID = hex_string

event_id = hex_string

hex_string = "0x" 1*hex

hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

acap_abs_path = "/" path_segments
               (see restriction 7 below)

(path_segments is defined in IETF RFC 2396 [29].)

path_segments = segment *( "/" segment )

segment = *pchar *( ";" param )

param = *pchar

pchar = unreserved | escaped | ":" | "@" | "&" | "=" | "+" | "$" | ","

unreserved = alphanum | mark

unreserved_not_dot = alphanum | mark_not_dot

alphanum = alpha | digit

alpha = lowalpha | upalpha

lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m"
          | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

upalpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" |
          | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

escaped = "%" hex hex

mark = "-" | "_" | "." | "!" | "~" | "*" | "'" | "(" | ")"

mark_not_dot = "-" | "_" | "!" | "~" | "*" | "'" | "(" | ")"

```

This syntax is fully compliant with the generic syntax of URIs as specified in RFC 2396 [29] and uses the registry-based naming authority version of that recommendation. Furthermore, all generic definitions specified in RFC 2396 [29] must be valid for the *acap* URI as well (e.g., escaping of special characters within file names, etc.).

14.2.1.1.1 Additional Restrictions

The following additional restrictions apply to the ACAP URI scheme:

1. When the `acap_net_path` part is missing and only the `acap_abs_path` is present, the URL refers to a file in a default object carousel within the current service.
2. If the `acap_entity` is an `acap_service` (i.e. not a `acap_service_component`) then there shall only be one Object Carousel in the ACAP service.
3. If the `stream_type` is an audio stream type, then the `ISO_639_language_code` may be used to select a specific language track (namely audio). If the `ISO_639_language_code` is not present, then the default language track is selected.

The default language track is defined as follows:

- if exactly one audio stream for the default language (as defined by the "User Language " preference in `org.dvb.user.GeneralPreference`) is signalled then that stream is the default.
 - if no audio streams are signalled with the default language then the first audio stream listed in the PMT is the default.
 - if more than one audio stream is signalled with the default language then the first such stream listed in the PMT is the default.
4. The name may contain the characters other than "unreserved_not_dot" as defined above by encoding each such character using its ASCII representation. If the name needs to include other characters these must be represented using the escaped sequence defined in IETF RFC 2396 [29]. For example, the character sequence "B&B" can be expressed as "B%26B". The name in the URL shall be translated to UTF-8 before URL byte escaping is applied.
 5. The name may contain the characters other than "unreserved" as defined above by encoding each such character using its ASCII representation. If the `component_name` needs to include other characters these must be represented using the escaped sequence defined in IETF RFC 2396 [29]. For example, the character sequence "B&B" can be expressed as "B%26B". The name in the URL shall be translated to UTF-8 before URL byte escaping is applied.
 6. The encoding format of `ISO_639_Language_code` is UTF-8.
 7. The following restrictions apply to the `acap_abs_path` part of a name:
 - The total length of pathnames, separators and filename shall be less than or equal to 254 bytes long.
 - The following characters are not allowed in filenames and pathnames: character null (0xC080), byte zero.
 - The encoding of the filename is in UTF-8 (as defined in GEM [1] Section 7.1.5)
 - The directory separator character (i.e., Java's `path.separator` property) shall be a slash character (0x2F).
 - An absolute filename starts with a slash character (as indicated in the BNF above).

14.2.1.2 Extended ACAP URI Scheme for ACAP-X

The following extensions to the ACAP URI scheme shall be valid when used by ACAP-X applications.

```
acap_x_uri = acap_uri | acap_scheme ":" acap_x_hierpart
acap_x_hierpart = acap_x_net_path
acap_x_net_path = "/" acap_x_entity
```

```

acap_x_entity = acap_service_contextual | acap_service_component_contextual |
ait_specifier

acap_service_contextual = "current" | "original"
acap_service_component_contextual = "current.audio" | "current.video" | "current.av"
ait_specifier = ait_filter "." ait_abs_path
ait_filter = "current"
ait_abs_path = "/" ait_entity
ait_entity = ait_root_directory | ait_application
ait_root_directory = "app_root"
ait_application = org_id "." app_id [ "?" ait_params ]
org_id = lowercase_hex_string
app_id = lowercase_hex_string
lowercase_hex_string = "0" | lowercase_hex_not_zero 0*lowercase_hex
lowercase_hex = digit | "a" | "b" | "c" | "d" | "e" | "f"
lowercase_hex_not_zero = digit_not_zero | "a" | "b" | "c" | "d" | "e" | "f"
digit_not_zero = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
ait_params = "arg_" 1*digit "=" *uric ["&" ait_params ]

```

14.2.1.3 Referencing Specific Entities

14.2.1.3.1 Program Streams

Where `acap_entity` is an `acap_service`, the ACAP service that consists of entire program streams identified by the entity is referenced.

14.2.1.3.2 Program Elements

Where `acap_entity` is an `acap_service_component`, a single program element is referenced.

14.2.1.3.3 Files and Directories

When a path is present in a URL where the `acap_entity` part identifies an ACAP service, the path references an object in an object carousel within the service.

When a path is present in a URL where the `acap_entity` part identifies one component of an ACAP service and that component carries an object carousel stream, the path references an object in an object carousel whose “root” (i.e., DSI message) is sent within that component. In this case the component tag set shall only contain one element. The semantics when the path is present in URL where the `acap_entity` part identifies something else than the two cases described above are not specified in this standard.

14.2.1.3.4 Resolution of Locator Elements

In cable receivers, when the POD Module is present, locators shall be resolved using the SI present in the OOB signaling. In cable receivers when the POD Module is absent, the in-band SI shall be used for resolution. See ANSI/SCTE 65 [45] for out-of-band and A/65 [47] for in-band.

14.2.1.3.4.1 Contextual

Table 14.1 ACAP URI Contextual Constructs

Name	Cable	Terrestrial	Comment
current	The Virtual Channel to which the ACAP receiver is currently tuned to.		The ACAP receiver needs to keep a record for the Virtual Channel listing the application making use of this identifier. Otherwise, ambiguity may occur if the receiver has multiple tuners.
current.av	The default audio and video components of the 'current' service (see above) as specified in GEM [1], Section 11.6.2.		
current.audio	The default audio component of the 'current' service (see above) as specified in GEM [1], Section 11.6.2.		
current.video	The default video component of the 'current' service (see above) as specified in GEM [1], Section 11.6.2.		
original	The Virtual Channel to which the ACAP receiver was originally tuned to when launching the application.		

14.2.1.3.4.2 Universally Resolvable

Constructs listed in Table 14.2 rely on signaling which is mandatorily present in terrestrial and all profiles of cable.

Table 14.2 ACAP URI Universally Resolvable Constructs

Name	Cable	Terrestrial	Comment
source_id	The source_id field in the VCM_structure of the Short Form Virtual Channel Table (Profile 1 through 5) or the source_id field in Long Form Virtual Channel Table (Profile 5 and 6) as defined in ANSI/SCTE 65 [45]. If both are present, the Short Form version shall be used. If no POD Module is present, then the source_id field in the Cable Virtual Channel Table as defined in ATSC A65/B [47].	The source_id field in the Terrestrial Virtual Channel Table as defined in ATSC A/65B [47].	
stream_type	The first program element matching that stream type. The stream types are defined in the stream type assignments table of ISO 13818-1 [22] and in the Stream Type Codes table of SCTE 54 [31].	The first program element matching that stream type. The stream types are defined in the stream type assignments table of ISO 13818-1 [22] and in A/53D [7].	For cable, note that the specified stream is not guaranteed to be decoded if the OpenCable Core Functional Requirements [60] does not support decoding it.
org_id/app_id	The org_id and app_id identifier shall correspond to the organization_id and application_id field, respectively, in the Application Identifier of an Application Information Table (AIT) as defined in Section 10.6.3, "Application Information Table."		
app_root	The app_root name shall correspond to the root directory path of the application as found in the acap_x application location descriptor within the AIT, as defined in Section 10.6.5.2.2, "ACAP-X Application Location Descriptor."		
ISO 639 language code	The first audio program element where there is a match between the specified ISO 639-2 3-character language code and the contents of the ISO 639 descriptor.		

Where a source_id to be resolved is found in both terrestrial and cable, the source_id shall be resolved according to the original delivery mode of the application on whose behalf the source_id is being resolved. Hence where an ACAP application that was delivered over a terrestrial network interface uses a source_id found in both terrestrial and cable, the source_id shall be resolved according to the mechanism defined above for terrestrial networks and vice-versa for applications delivered over cable network interfaces.

14.2.1.3.4.3 Environment Specific

Constructs listed in Table 14.3 are those where the underlying signaling is not required to be present in all terrestrial and all cable profiles.

Table 14.3 ACAP URI Environment Specific Constructs

Name	Cable	Terrestrial	Comment
service_name	If the service information contains a Long-form Virtual Channel Table, Terrestrial Virtual Channel Table or Cable Virtual Channel Table, the short_name from that table is translated to a UTF-8 string and compared with the UTF-8 representation of service_name. Otherwise, if the service information contains a Source Name Sub-table in the Network Text Table, each source_name component with mode less than 0x40 is translated to a UTF-8 string according to its mode and byte string and compared with the UTF-8 representation of service_name. Components of source_name using format-effector modes are ignored in the comparison. Otherwise the service_name is not resolvable.		Use of this in cable assumes the MSO ensures these names are uniquely correlated with source_ids in their network. These names are not interchangeable between cable networks.
component_tag	A component_tag value in one of the Stream Identifier Descriptors located in the inner descriptor loop of the TS_program_map_section associated with the Virtual Channel identified.	Not defined.	Where component tag is used with an environment specific virtual channel identification (e.g. short_name) then it is also environment specific.
component_name	The component name string in the Component Name Descriptor located in the inner descriptor loop of the TS_program_map_section associated with the Virtual Channel (see below).	Not defined.	This identifier can only be used with cable systems supporting Profiles 4, 5 and 6 of ANSI/SCTE 65 [45].
event_id	The event_id identifier shall correspond to the event_ID in the Aggregate Event Information Table (AEIT) as defined in ANSI/SCTE 65 [45].	The event_id shall correspond To an event_ID field in an Event Information Table as defined in ATSC A/65B [47].	Event identifiers shall be scoped by a Virtual Channel identifier. In cable, this identifier may only be resolved in systems supporting Profiles 4,5,6.

The component_name in the PMT is represented as a Multiple String Structure with each set of string components associated with a specific language. The set of string components corresponding to language code eng are selected, and decompressed for comparison. Each PMT component_name string component with mode less than 0x40 is translated to a UTF-8 string according to its mode and byte string and compared with the UTF-8 representation of the component_name extracted from the locator. Components of the PMT component_name using format-effector modes are ignored in the comparison.

14.2.1.3.4.4 Physical Constructs

Constructs listed in Table 14.4 are specific to a particular environment or cable head-end.

Note: Applications should not include hard-coded values of these. Locators using them are intended to be dynamically constructed in the ACAP receiver based on locally accurate information; e.g., as would be returned by `org.ocap.si.PMTElementaryStreamInfo.getElementaryPID()`.

Table 14.4 ACAP URI Physical Layer Constructs

Name	Cable	Terrestrial	Comment
frequency	The frequency is a 32-bit hex value in hertz, which can be used in cable to tune to a service that is only defined within an inband PAT and PMT.		
program_number	A 16-bit value as specified in ISO 13818-1 [22].		
PID	In this case the program element is indicated by the PID.		
Index	In this case the program element is the indexed program element matching that stream type. The index specifies the ordinal number of the elementary streams that have same stream_type in the PMT. The first elementary stream of them is index = 0.		If multiple MPEG PES of the same stream type are present in the program, then the index can be used to select the first, second, third, and so forth.

14.3 Persistent Local Storage

As specified in Section 14.6 of GEM [1].

14.4 SCTE 90-1 Exceptions

The following exceptions to SCTE 90-1 shall apply:

1. The term ‘navigator’ in GEM Section 9 Application model and its subsections of DVB-GEM 1.0.0 shall be replaced with ‘monitor application,’ not ‘unbound application’ as in SCTE 90-1. This replacement results in the statement in SCTE 90-1 Section 10.2.1.1 effectively being:

“The term ‘navigator’ in the Section 9 Application model and its sub sections of DVB-GEM 1.0.0 [9] shall be replaced by ‘monitor application.’”

2. Section T.2.2.1.2 of 90-1 shall be replaced with:

“With POD device absent the content rating information is provided in the Rating Region Tables defined in PSIP and is available to bound applications from the system information delivered with the associated transport stream. For unbound applications the data may be available from one or the collection of transport streams available.”

3. Section T.2.2.1.4 of 90-1 shall be replaced with:

“There is no Network Text Table, the service name is provided in the Cable Virtual Channel Table or the Terrestrial Virtual Channel Table.”

4. Section T.2.2.1.6 of 90-1 shall be replaced with:

“There is no Shortform Virtual Channel Table, since the service name is provided in the Cable Virtual Channel Table or the Terrestrial Virtual Channel Table.”

5. The last sentence of Section T.2.2.1.8 of 90-1 shall be null and void. The resulting section (for ACAP applicability) then reads:

“The service name, service type and channel number are provided in the Cable Virtual Channel Table (CVCT) defined in PSIP. The extended channel name is provided in the Extended Channel Name descriptor.”

6. The last sentence of Section T.2.2.1.9 of 90-1 shall be replaced with:

“For other profiles, any event retrieval request will result in a call to the notifyFailure() method of the javax.tv.service.SIRequestor interface with a SIRequestFailureType of DATA_UNAVAILABLE.”

15. MINIMUM RECEIVER REQUIREMENTS

15.1 General

Annex G of GEM [1] shall apply.

15.2 User Input

For terrestrial receivers, Section G.5 of GEM [1] shall apply.

For cable receivers, additionally Section 25.2.1.2 “Input Events” of OCAP 1.0 [4] shall apply.

This standard recommends support for keyboard input, either provided at time of manufacture or the option of adding a keyboard input at a later time.

15.3 Graphics

GEM [1] Section G.1.2, “Minimum Color Lookup Table,” does not apply to this standard. No functional equivalent is provided in this standard. All ACAP receivers shall support a graphics resolution of at least 16 bits per pixel.

The minimal set of required device resolutions that ACAP terminals must support is as follows:

- HBackgroundDevice resolution of 640 x 480
- HVideoDevice resolution of 640 x 480
- HGraphicsDevice resolution of 640 x 480

These resolutions must be supported for display aspect ratios of 4:3 and 16:9.

If ACAP receivers either display or output HD video without down-conversion to SD then they shall support at least the graphics resolution of 960x540 in addition to those listed above.

16. DETAILED PLATFORM PROFILE DEFINITIONS

This section defines the capabilities of platforms as presented to applications. Products that claim to conform to a profile shall provide at least the minimum capabilities identified for the profile. In some cases this implies that specific hardware resources are present in the platform. See Table 16.1.

Table 16.1 Detailed Platform Profile Definitions

Area	Specification	ACAP-J only Profile	ACAP-J and ACAP-X Profile
GEM Compliance			
GEM	GEM [1], clause 15.0, "Interactive Broadcast Profile"	M	M
	Section 17.1, "Compliance with GEM"	M	M
Broadcast Streaming Formats			
Video	Section 6.3.1, "Video"	M	M
Audio	Section 6.3.2, "Audio"	M	M
Broadcast Channel Protocols			
Broadcast Channel Protocols	Section 10.1.1, "Notation" Section 10.2.1, "NSAP Address" Section 10.2.2, "Content Type and Timestamp Inheritance" Section 10.2.5, "Usage of Private Data for non-ACAP Extensions" Section 10.3.1, "Application Content Types" Section 10.3.2, "Application Protocol ID"	M	M
	Section 10.2.3, "Application Transport over HTTP" Section 10.2.4, "Time Stamp Descriptor" Section 10.3.4, "ACAP-X Extensions"	--	M
Interaction Channel Protocols			
Interaction Channel Protocols	Section 11.1.2, "Internet Protocol" Section 11.1.4, "Transmission Control Protocol (TCP)" Section 11.1.6, "Domain Name Service (DNS)"	M	M
	Section 11.1.5, "Hyper-Text Transfer Protocol (HTTP)"	--	M
ACAP-J environment			
ACAP-J environment	Section 7, "ACAP-J Applications and Environment," with the exception of: Section 7.2.1.2, "Inter-Environment DOM Integration" Section 7.2.1.3, "ACAP-X Permissions API"	M	M
	Section 7.2.1.2, "Inter-Environment DOM Integration" Section 7.2.1.3, "ACAP-X Permissions API"	--	M
ACAP-X environment			
ACAP-X environment	Section 8, "ACAP-X Applications and Environment"	--	M
Resource Reference and Locators			
Resource Reference and Locators	Section 14.2.1.1, "Scheme Definition"	M	M
	Section 14.2.1.2, "Extended ACAP URI Scheme for ACAP-X"	--	M
Minimum Receiver Requirements			
GEM Platform Minima	GEM [1], clauses 15.1 through 15.5	M	M
ACAP Platform Minima	Section 15, "Minimum Receiver Requirements"	M	M
Security	Section 12, "Security:"	M	M

17. CONFORMANCE

17.1 Compliance with GEM

ACAP terminals shall comply in full with GEM [1]. This standard adopts the MHP definition of the following functional equivalents, as specified in GEM [1] clause 15.6:

- Arch
- Carousel
- Application Signalling

- Text Wrapping

The following optional functional equivalents are not specified by this document, and need not be supported in an ACAP terminal:

- IP MPE
- Broadcast IP signalling
- Subtitles
- Conditional Access
- If an ACAP terminal supports any of the optional functional equivalents listed above, it must be minimally by the GEM requirements for that optional functionality.
- All other functional equivalents are defined in this document.
- For avoidance of doubt, in the event of a conflict between GEM [1] and this standard, the normative guarantees of GEM [1] shall take precedence except as detailed in Section 17.1.1, “GEM Errata.”

17.1.1 GEM Errata

No errata to GEM have been identified.

17.1.2 Modifications to MHP Definitions of Functional Equivalents

As described in GEM [1] clause 15.6.1, GEM terminal specifications may slightly modify the MHP definition of a functional equivalent in constrained ways.

17.1.2.1 Application Icons Descriptor

This standard builds on the Application Icons Descriptor, which is required by the “application signaling” functional equivalent and described in MHP [2] clause 10.7.4.2, as follows. The table entitled “Icon Locator Semantics” is extended for the application types introduced by Section 10.3.1, “Application Content Types.”

application_type	Description
0x0006	For ACAP-J this is a path relative to the base directory of the application as defined in MHP [2] clause 10.9.2, “DVB-J application location descriptor.”
0x0007	For ACAP-X this is a path relative to the base directory of the application as defined in Section 10.6.5.2.2, “ACAP-X Application Location Descriptor.”

A/101 Annex A: **Content Identification API**

A1. PACKAGE ORG.ATSC.SI

A1.1 Description

This package provides SI extensions for ACAP.

(See the following pages.)

org.atsc.si: ContentIdentification

- **Declaration:** public interface ContentIdentification
- **All Known Subinterfaces:** ISANIdentification, VISANIdentification
- **Description:** Superinterface for all content identification system specific retrieval interfaces.

Member Summary	
Methods	
int	getIdentificationSystem() Returns the value of the Metadata_application_format from the content labeling descriptor as defined in ISO/IEC 13818-1:2000/PDAM 4.
java.lang.String	getIdentifier() Returns a string representation of the underlying content identification value.
byte[]	getIdentifierBytes() Provides an array of n bytes that represent the underlying n byte value of the content identification.

Methods

getIdentificationSystem()

public int **getIdentificationSystem()**

Returns the value of the Metadata_application_format from the content labeling descriptor as defined in ISO/IEC 13818-1:2000/PDAM 4

Returns:

integer representation of the Metadata_application_format value.

getIdentifier()

public java.lang.String **getIdentifier()**

Returns a string representation of the underlying content identification value.

Returns:

string representation of the content identification.

getIdentifierBytes()

public byte[] **getIdentifierBytes()**

Provides an array of n bytes that represent the underlying n byte value of the content identification.

Returns:

array of n bytes.

org.atsc.si: ContentIdentifications

- **Declaration:** public interface ContentIdentifications
- **Description:** Provides an array of references to the various content identifiers associated with an instance of javax.tv.service.guide.ProgramEvent.

Member Summary
Methods
ContentIdentification[] getIdentifiers() Returns an array of objects that implement the ContentIdentification interface.

Methods

getIdentifiers()

public org.atsc.si.ContentIdentification[] **getIdentifiers()**

Returns an array of objects that implement the ContentIdentification interface. In the case where the underlying program event does not contain content identifiers, the getIdentifiers() method shall return an empty array.

See Also:

ContentIdentification

org.atsc.si: ISANIdentification

- **Declaration:** public interface ISANIdentification extends ContentIdentification
- **All Superinterfaces:** ContentIdentification
- **All Known Superinterfaces:** VISANIdentification
- **Description:** Interface for retrieving ISO 15706 compliant content identification value for the underlying program event. This interface shall only be implemented when the underlying program event is identified by a ISAN.

Member Summary	
Methods	
int	getISANEpisodeIdentifier() Provides the episode segment of an ISAN content identification.
java.lang.String	getISANIdentifier() Provides a string representation of the ISAN Identifier.
byte[]	getISANIdentifierBytes() Provides an array of 8 bytes that represent the underlying 8 byte value of the ISAN content identification.
long	getISANRootIdentifier() Provides the root portion of an ISAN content identification.

Inherited Member Summary
Methods inherited from interface ContentIdentification getIdentificationSystem(), getIdentifier(), getIdentifierBytes()

Methods

getISANEpisodeIdentifier()

public int **getISANEpisodeIdentifier()**

Provides the episode segment of an ISAN content identification.

Returns:

Integer value that is the episode segment of the ISAN content identification.

getISANIdentifier()

public java.lang.String **getISANIdentifier()**

Provides a string representation of the ISAN Identifier. The string representation shall be conformant with ISO 15706.

Returns:

String representation of the underlying ISAN content identification.

getISANIdentifierBytes()

public byte [] **getISANIdentifierBytes()**

Provides an array of 8 bytes that represent the underlying 8 byte value of the ISAN content identification.

Returns:

Array of 8 Bytes.

getISANRootIdentifier()

public long **getISANRootIdentifier()**

Provides the root portion of an ISAN content identification.

Returns:

Long value that is the root segment of the ISAN content identification.

org.atsc.si: VISANIdentification

- **Declaration:** public interface VISANIdentification extends ISANIdentification
- **All Superinterfaces:** ContentIdentification, ISANIdentification
- **Description:** Interface for retrieving ISO 20925-1 compliant content identification values. This interface shall only be implemented when the underlying program event is identified by a V-ISAN.

Member Summary	
Methods	
java.lang.String	getVISANIdentifier() Provides a string representation of the V-ISAN content identification.
byte[]	getVISANIdentifierBytes() Provides an array of 12 bytes that represent the underlying 12 byte value of the VISAN content identification.
int	getVISANVersionIdentifier() Provides the version segment of a V-ISAN content identification.

Inherited Member Summary
Methods inherited from interface ContentIdentification getIdentificationSystem(), getIdentifier(), getIdentifierBytes()
Methods inherited from interface ISANIdentification getISANEpisodeIdentifier(), getISANIdentifier(), getISANIdentifierBytes(), getISANRootIdentifier()

Methods

getVISANIdentifier()

public java.lang.String **getVISANIdentifier()**

Provides a string representation of the V-ISAN content identification. The string representation shall be conformant with ISO 20925-1.

Returns:

String representation of the underlying V-ISAN content identification.

getVISANIdentifierBytes()

public byte[] **getVISANIdentifierBytes()**

Provides an array of 12 bytes that represent the underlying 12 byte value of the V-ISAN content identification.

Returns:

Array of 12 bytes.

getVISANVersionIdentifier()

public int **getVISANVersionIdentifier()**

Provides the version segment of a V-ISAN content identification.

Returns:

Integer value that is the version segment of a V-ISAN content identification

A/101 Annex B:

Document Type Definitions (Normative)

B1. SCOPE

This annex specifies the following document types as used by ACAP XML based content types:

- ACAP Permission Request File Document Type
- ACAP-J Font Index File Document Type
- ACAP-X Application Metadata Document Type
- ACAP-X Markup Document Type

B2. ACAP PERMISSION REQUEST FILE DOCUMENT TYPE DEFINITION

This document type defines the ACAP permission request file schema.

B2.1 acap-permission-1.dtd

```
<!-- ..... -->
<!--          ACAP Permission Request File 1.0 DTD          -->
<!-- ..... -->
```

<!-- This is the DTD for the ACAP 1.0 permission request file.

The following formal public identifier shall be used to identify it:

"-//ATSC//DTD ACAP Permission Request File 1.0//EN"

The following URL for the SystemLiteral may be used to reference this file :

<http://www.atsc.org/acap/dtd/acap-permission-1.dtd>

-->

<!-- Basic entities definition -->

<!ENTITY % URI.datatype "CDATA" >

<!-- -->

<!-- The section below declares the OCAP extensions to the GEM PRF -->

<!-- -->

<!-- Declare the OCAP namespace -->

<!ENTITY % ocap.xmlns "http://www.cablelabs.com/ocap" >

<!-- Declare the OCAP prefix associated with this namespace -->

<!ENTITY % ocap.prefix "ocap" >

<!-- Declare the xml namespace attribute for OCAP -->

<!ENTITY % ocap.xmlns.attrib

xmlns:ocap.prefix; %URI.datatype; #FIXED '%ocap.xmlns;' "

>

<!-- Declare the additional OCAP-defined elements and attributes -->

<!ENTITY % ocap.monitorapplication.qname "%ocap.prefix;monitorapplication" >

<!ENTITY % ocap.monitorapplication.name.qname "name" >

```

<!ENTITY % ocap.monitorapplication.value.qname "value" >

<!-- ..... -->
<!-- The section below declares the ACAP extensions to the GEM PRF -->
<!-- ..... -->
<!--      Declare the ACAP namespace -->
<!ENTITY % acap.xmlns "http://www.atsc.org/acap#permission" >
<!--      Declare the ACAP prefix associated with this namespace -->
<!ENTITY % acap.prefix "acap" >

<!-- Declare the xml namespace attribute for ATSC -->
<!ENTITY % acap.xmlns.attrib
  "xmlns:%acap.prefix; %URI.datatype; #FIXED '%acap.xmlns;' "
>

!--      Declare the additional ACAP-defined elements and attributes -->
<!ENTITY % acap.cookie.qname "%acap.prefix;:cookie" >
<!ENTITY % acap.cookie.target.qname "target" >
<!ENTITY % acap.cookie.actions.qname "actions" >

<!ENTITY % acap.rce.qname "%acap.prefix;:rce" >
<!ENTITY % acap.rce.value.qname "value" >

<!ENTITY % acap.bridge.qname "%acap.prefix;:bridge" >
<!ENTITY % acap.bridge.value.qname "value" >

<!-- All elements and attributes defined in GEM 1.0.x shall be supported and -->
<!-- inserted at this level with appropriate extensions necessary for -->
<!-- ACAP (prefixed by "acap:") and CableLabs (prefixed by "ocap:") -->

<!ELEMENT permissionrequestfile
  (file?, capermission?, applifecyclecontrol?, returnchannel?, tuning?,
  servicesel?, userpreferences?, network?, dripfeed?, persistentfilecredential*,
  %acap.cookie.qname;, %acap.rce.qname;, %acap.bridge.qname;,
  %ocap.monitorapplication.qname;*)>

<!ATTLIST permissionrequestfile
  orgid CDATA #REQUIRED
  appid CDATA #REQUIRED
>

<!ELEMENT file EMPTY>
<!ATTLIST file
  value (true|false) "true"
>

<!ELEMENT capermission (casystemid)+>
<!ELEMENT casystemid EMPTY>
<!ATTLIST casystemid
  entitlementquery (true|false) "false"
  id CDATA #REQUIRED
  mmi (true|false) "false"
  messagepassing (true|false) "false"
  buy (true|false) "false"

```

```
>

<!ELEMENT applifecyclecontrol EMPTY>
<!ATTLIST applifecyclecontrol
  value (true|false) "true"
>

<!ELEMENT returnchannel (defaultisp?,phonenumber*)>
<!ELEMENT defaultisp EMPTY>
<!ELEMENT phonenumber (#PCDATA)>

<!ELEMENT tuning EMPTY>
<!ATTLIST tuning
  value (true|false) "true"
>

<!ELEMENT servicesel EMPTY>
<!ATTLIST servicesel
  value (true|false) "true"
>

<!ELEMENT userpreferences EMPTY>
<!ATTLIST userpreferences
  write (true|false) "false"
  read (true|false) "true"
>

<!ELEMENT network (host)+>
<!ELEMENT host (#PCDATA)>
<!ATTLIST host
  action CDATA #REQUIRED
>

<!ELEMENT dripfeed EMPTY>
<!ATTLIST dripfeed
  value (true|false) "true"
>

<!ELEMENT persistentfilecredential (grantoridentifier, expirationdate, filename+,
  signature, certchainfileid)>
<!ELEMENT grantoridentifier EMPTY>
<!ATTLIST grantoridentifier
  id CDATA #REQUIRED
>
<!ELEMENT expirationdate EMPTY>
<!ATTLIST expirationdate
  date CDATA #REQUIRED
>
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
  write (true|false) "true"
  read (true|false) "true"
>
<!ELEMENT signature (#PCDATA)>
```

```

<!ELEMENT certchainfileid (#PCDATA)>

<!-- In addition, the following elements and attributes are defined in order
to support OCAP specific behaviour. -->

<!ELEMENT %ocap.monitorapplication.qname; EMPTY>

<!ENTITY % OCAPMonitorAppPermType.class
"(registrar | service | servicemanager | security | reboot | systemevent |
handler.appFilter | handler.resource | handler.closeCaptioning |
filterUserEvents | handler.eas | setVideoPort | podApplication |
signal.configured | properties | storage | registeredapi | vbifiltering |)"
>
>
<!ATTLIST %ocap.monitorapplication.qname;
%ocap.monitorapplication.name.qname; %OCAPMonitorAppPermType.class; #REQUIRED
%ocap.monitorapplication.value.qname; (true | false) #REQUIRED
%ocap.xmlns.attrib;
>

<!-- In addition, the following elements and attributes are defined in order
to support ACAP-X application specific behaviour. -->

<!-- cookie permission request -->
<!ELEMENT %acap.cookie.qname; EMPTY>
<!ATTLIST %acap.cookie.qname;
%acap.target.qname; CDATA #REQUIRED
%acap.actions.qname; CDATA #REQUIRED
%acap.xmlns.attrib;
>

<!-- runtime code extension permission request -->
<!ELEMENT %acap.rce.qname; EMPTY>
<!ATTLIST %acap.rce.qname;
%acap.rce.value.qname; (true|false) #REQUIRED
%acap.xmlns.attrib;
>

<!-- Java bridge permission request -->
<!ELEMENT %acap.bridge.qname; EMPTY>
<!ATTLIST %acap.bridge.qname;
%acap.bridge.value.qname; (true|false) #REQUIRED
%acap.xmlns.attrib;
>

```

B3. ACAP-J FONT INDEX FILE DOCUMENT TYPE DEFINITION

This document type defines the ACAP-J application font index file schema.

B3.1 acap-j-font-index-1.dtd

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!-- ACAP-J Font Index File 1.0 DTD -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- This is the DTD for the ACAP-J 1.0 font index file.

```

The following formal public identifier shall be used to identify it:

```
"-//DVB//DTD Font Directory 1.0//EN"
```

The following URL for the SystemLiteral may be used to reference this file :

```
http://www.atsc.org/acap/dtd/acap-j-font-index-1.dtd
```

```
-->
```

```
<!ELEMENT fontdirectory (font)+>
<!ELEMENT font (name,fontformat,filename,style*,size?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT fontformat (#PCDATA)>
<!ELEMENT filename (#PCDATA)>
<!ELEMENT style (#PCDATA)>
<!ELEMENT size EMPTY>
<!ATTLIST size
  min CDATA "0"
  max CDATA "maxint"
>
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               END END END                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
```

B4. ACAP-X APPLICATION METADATA DOCUMENT TYPE DEFINITION

This document type defines the ACAP-X application metadata schema.

B4.1 acap-x-metadata-1.dtd

```
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               ACAP-X Application Metadata 1.0 DTD                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!--
  This is ACAP-X Application Metadata 1.0, an XML Document Type specified
  for use with ACAP-X Applications.

  This module shall be identified by the following formal public identifier:

  "-//ATSC//DTD ACAP-X Application Metadata 1.0//EN"

  The following URL for the SystemLiteral may be used to reference
  this file :

  http://www.atsc.org/acap/dtd/acap-x-metadata-1.dtd
-->

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Parameters                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % XMLNS "http://www.atsc.org/acap#metadata" >
```

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--          Data Type Entity Declarations          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- media type, as per [RFC2045] -->
<!ENTITY % ContentType.datatype "CDATA" >

<!-- a language code, as per [LANG-TAGS] -->
<!ENTITY % LanguageCode.datatype "NMTOKEN" >

<!-- a Uniform Resource Identifier, see [URI] -->
<!ENTITY % URI.datatype "CDATA" >

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--          Attribute Entity Declarations          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % id.attrib
  "id          ID          #IMPLIED"
>

<!ENTITY % lang.attrib
  "xml:lang    %LanguageCode.datatype; #IMPLIED"
>

<!ENTITY % xmlns.attrib
  "xmlns      %URI.datatype;    #FIXED '%XMLNS;'"
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--          Qualified Element Name Entity Declarations          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % application.qname      "application" >
<!ENTITY % identifier.qname      "identifier" >
<!ENTITY % entityset.qname       "entityset" >
<!ENTITY % entity.qname          "entity" >
<!ENTITY % descset.qname         "descset" >
<!ENTITY % name.qname            "name" >
<!ENTITY % desc.qname            "desc" >
<!ENTITY % condset.qname         "condset" >
<!ENTITY % cond.qname            "cond" >
<!ENTITY % cacheset.qname        "cacheset" >
<!ENTITY % cache.qname           "cache" >
<!ENTITY % paramset.qname        "paramset" >
<!ENTITY % param.qname           "param" >

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--          Element Classes          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % optsets.class
  "%condset.qname; |
  %cacheset.qname; |
  %paramset.qname;" >

```

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Element Declarations                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % application.content          "(%identifier.qname;,
                                         %entityset.qname;,
                                         %descset.qname;+,
                                         (%optsets.class;)*)" >
<!ELEMENT %application.qname;          %application.content; >
<!ATTLIST %application.qname;
          %xmlns.attrib;
          %lang.attrib;
          %id.attrib;
>

<!ENTITY % identifier.content           "(%param.qname;)*">
<!ELEMENT %identifier.qname;           %identifier.content; >
<!ATTLIST %identifier.qname;
          %xmlns.attrib;
          %lang.attrib;
          %id.attrib;
          uuid          CDATA          #REQUIRED
>

<!ENTITY % entityset.content           "(%entity.qname;)+>
<!ELEMENT %entityset.qname;           %entityset.content; >
<!ATTLIST %entityset.qname;
          %xmlns.attrib;
          %lang.attrib;
          %id.attrib;
>

<!ENTITY % entity.content              "EMPTY">
<!ELEMENT %entity.qname;              %entity.content; >
<!ATTLIST %entity.qname;
          %xmlns.attrib;
          %lang.attrib;
          %id.attrib;
          entitytype   CDATA          #REQUIRED
          uri          %URI.datatype; #REQUIRED
>

<!ENTITY % descset.content             "(%name.qname;, %desc.qname;)" >
<!ELEMENT %descset.qname;             %descset.content; >
<!ATTLIST %descset.qname;
          %xmlns.attrib;
          %lang.attrib;
          %id.attrib;
>

<!ENTITY % name.content                "(#PCDATA)">
<!ELEMENT %name.qname;                %name.content; >
<!ATTLIST %name.qname;

```

```

    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
>

<!ENTITY % desc.content          "(#PCDATA)">
<!ELEMENT %desc.qname;          %desc.content; >
<!ATTLIST %desc.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
>

<!ENTITY % condset.content      "(%cond.qname;)+">
<!ELEMENT %condset.qname;      %condset.content; >
<!ATTLIST %condset.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
>

<!ENTITY % cond.content         "(%param.qname;)*">
<!ELEMENT %cond.qname;         %cond.content; >
<!ATTLIST %cond.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
    capability    CDATA          #REQUIRED
    qualifier     CDATA          #IMPLIED
>

<!ENTITY % cacheset.content     "(%cache.qname;)+">
<!ELEMENT %cacheset.qname;     %cacheset.content; >
<!ATTLIST %cacheset.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
>

<!ENTITY % cache.content        "EMPTY">
<!ELEMENT %cache.qname;        %cache.content; >
<!ATTLIST %cache.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
    target        %URI.datatype; #REQUIRED
    directives    CDATA          #REQUIRED
>

<!ENTITY % paramset.content     "(%param.qname;)+">
<!ELEMENT %paramset.qname;     %paramset.content; >
<!ATTLIST %paramset.qname;
    %xmlns.attrib;
    %lang.attrib;

```

```

    %id.attrib;
>

<!ENTITY % param.content          "EMPTY">
<!ELEMENT %param.qname;          %param.content; >
<!ATTLIST %param.qname;
    %xmlns.attrib;
    %lang.attrib;
    %id.attrib;
    name          CDATA          #REQUIRED
    value         CDATA          #IMPLIED
>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               END END END                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

B5. ACAP-X MARKUP DOCUMENT TYPE DEFINITION

This document type defines the ACAP-X application markup schema, also known as XDML.

B5.1 acap-x-xdml-1.dtd

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               XDML 1.0 DTD Driver                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!--
    This is XDML 1.0, an XHTML Host Language Document Type specified for
    use with ACAP-X applications.

    This module shall be identified by the following formal public identifier:

    "-//ATSC//DTD XHTML ACAP XDML 1.0//EN"
-->

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               XHTML Driver Parameters                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % XHTML.version "-//ATSC//DTD XHTML ACAP-X XDML 1.0//EN">
<!ENTITY % XHTML.profile "">

<!ENTITY % XHTML.dtd.sysid.base
    "http://www.w3.org/TR/xhtml-modularization/DTD/">

<!ENTITY % ACAP.xmlns "http://www.atsc.org/acap#markup">

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Framework                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Framework Parameter: Include BIDI support -->
<!ENTITY % XHTML.bidi "INCLUDE">

```

```

<!-- Framework Parameter: Ignore intrinsic event attributes -->
<!ENTITY % xhtml-events.module "IGNORE">

<!-- Framework Parameter: Define Content Model -->
<!ENTITY % xhtml-model.mod
    PUBLIC "-//ATSC//ENTITIES ACAP-X XXML Content Model 1.0//EN"
        "acap-x-xml-model-1.ent">

<!-- Framework Parameter: %Core.attrib; extensions -->
<!ENTITY % base.attrib "xml:base CDATA #IMPLIED">
<!ENTITY % style.attrib "style CDATA #IMPLIED">
<!ENTITY % Core.extra.attrib "%base.attrib; %style.attrib;">

<!-- Framework Parameter: missing qnames -->
<!ENTITY % frameset.qname "frameset">

<!-- Modular Framework Module -->
<!ENTITY % xhtml-framework.mod
    PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-framework-1.mod">
%xhtml-framework.mod;

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Element Modules                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Basic Text Module -->
<!ENTITY % xhtml-text.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-text-1.mod">
%xhtml-text.mod;

<!-- Hypertext Module -->
<!ENTITY % xhtml-hypertext.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-hypertext-1.mod">
%xhtml-hypertext.mod;

<!-- Lists Module -->
<!ENTITY % xhtml-list.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-list-1.mod">
%xhtml-list.mod;

<!-- Presentation Module -->
<!ENTITY % xhtml-pres.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-pres-1.mod">
%xhtml-pres.mod;

<!-- BIDI Override Element Module -->
<!ENTITY % xhtml-bdo.mod
    PUBLIC "-//W3C//ELEMENTS XHTML BIDI Override Element 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-bdo-1.mod">

```

```
%xhtml-bdo.mod;

<!-- Forms Module -->
<!ENTITY % xhtml-form.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-form-1.mod">
%xhtml-form.mod;

<!-- Tables Module -->
<!ENTITY % xhtml-table.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-table-1.mod">
%xhtml-table.mod;

<!-- Param Element Module -->
<!ENTITY % xhtml-param.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Param Element 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-param-1.mod">
%xhtml-param.mod;

<!-- Object Element Module -->
<!ENTITY % xhtml-object.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-object-1.mod">
%xhtml-object.mod;

<!-- Image Element Module -->
<!ENTITY % xhtml-image.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-image-1.mod">
%xhtml-image.mod;

<!-- Client-side Image Map Module -->
<!ENTITY % xhtml-csismap.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Client-side Image Maps 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-csismap-1.mod">
%xhtml-csismap.mod;

<!-- Link Element Module -->
<!ENTITY % xhtml-link.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Link Element 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-link-1.mod">
%xhtml-link.mod;

<!-- Base Element Module -->
<!ENTITY % xhtml-base.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-base-1.mod">
%xhtml-base.mod;

<!-- Document Metainformation Module -->
<!ENTITY % xhtml-meta.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-meta-1.mod">
```

```

%xhtml-meta.mod;

<!-- Scripting Module -->
<!ENTITY % xhtml-script.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-script-1.mod">
%xhtml-script.mod;

<!-- Stylesheets Module -->
<!ENTITY % xhtml-style.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Style Sheets 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-style-1.mod">
%xhtml-style.mod;

<!-- Target Attribute Module -->
<!ENTITY % xhtml-target.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-target-1.mod">
%xhtml-target.mod;

<!-- Frames Module -->
<!ENTITY % xhtml-frames.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Frames 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-frames-1.mod">
%xhtml-frames.mod;

<!-- Document Structure Module -->
<!ENTITY % xhtml-struct.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-struct-1.mod">
%xhtml-struct.mod;

<!-- Name Identifier Module -->
<!ENTITY % xhtml-nameident.mod
    PUBLIC "-//W3C//ELEMENTS XHTML Name Identifier 1.0//EN"
        "%XHTML.dtd.sysid.base;xhtml-nameident-1.mod">
%xhtml-nameident.mod;

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Non-Parameterized Extensions                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<![%legend.attlist;[
<!ATTLIST %legend.qname;
    align      ( top | bottom | left | right ) #IMPLIED
>
]]>

<![%param.attlist;[
<!ATTLIST %param.qname;
    %base.attrib;
>
]]>

```

```
<![%frameset.attlist;[
<!ATTLIST %frameset.qname;
    acap:onload %Script.datatype; #IMPLIED
    acap:onunload %Script.datatype; #IMPLIED
    acap:ondomstable %Script.datatype; #IMPLIED
    xmlns:acap %URI.datatype; #FIXED '%ACAP.xmlns;'
>
]]>

<![%body.attlist;[
<!ATTLIST %body.qname;
    acap:onload %Script.datatype; #IMPLIED
    acap:onunload %Script.datatype; #IMPLIED
    acap:ondomstable %Script.datatype; #IMPLIED
    xmlns:acap %URI.datatype; #FIXED '%ACAP.xmlns;'
>
]]>

<![%meta.attlist;[
<!ATTLIST %meta.qname;
    %base.attrib;
>
]]>

<![%script.attlist;[
<!ATTLIST %script.qname;
    %base.attrib;
>
]]>

<![%style.attlist;[
<!ATTLIST %style.qname;
    %base.attrib;
>
]]>

<![%title.attlist;[
<!ATTLIST %title.qname;
    %base.attrib;
>
]]>

<![%head.attlist;[
<!ATTLIST %head.qname;
    %base.attrib;
>
]]>

<![%html.attlist;[
<!ATTLIST %html.qname;
    %base.attrib;
>
]]>
```

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               END END END                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

B5.2 acap-x-xml-model-1.ent

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               XDMML 1.0 Document (Content) Model                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

```

<!--

```

This is the XDMML 1.0 Document (Content) Model Module for use with the XDMML 1.0 Document Type specified for ACAP Declarative Applications.

This module declares certain parameter entities that define groupings of elements employed in the definition of XDMML 1.0 content models.

This module shall be identified by the following formal public identifier:

```

"-//ATSC//ENTITIES ACAP XDMML Content Model 1.0//EN"

```

```

-->

```

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Non-Empty Content Group Classes                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

```

<!ENTITY % BlkStruct.class      "%p.qname;
                                |%div.qname;">

```

```

<!ENTITY % InlStruct.class      "%br.qname;
                                |%span.qname;">

```

```

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                               Optionally Empty Content Group Classes                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

```

<!ENTITY % Anchor.class        "|%a.qname;">

```

```

<!ENTITY % BlkPhras.class      "|%pre.qname;
                                |%blockquote.qname;
                                |%address.qname;">

```

```

<!ENTITY % BlkPres.class       "|%hr.qname;">

```

```

<!ENTITY % Heading.class       "|%h1.qname;
                                |%h2.qname;
                                |%h3.qname;
                                |%h4.qname;
                                |%h5.qname;
                                |%h6.qname;">

```

```

<!ENTITY % I18n.class          "|%bdo.qname;">

```

```

<!ENTITY % InlPhras.class      "|%abbr.qname;">

```

```

| %acronym.qname;
| %cite.qname;
| %code.qname;
| %dfn.qname;
| %em.qname;
| %kbd.qname;
| %q.qname;
| %samp.qname;
| %strong.qname;
| %var.qname; ">

<!ENTITY % InlPres.class      "| %b.qname;
| %big.qname;
| %i.qname;
| %small.qname;
| %sub.qname;
| %sup.qname;
| %tt.qname; ">

<!ENTITY % InlForm.class     "| %input.qname;
| %select.qname;
| %textarea.qname;
| %label.qname;
| %button.qname; ">

<!ENTITY % InlSpecial.class  "| %map.qname;
| %img.qname;
| %object.qname; ">

<!ENTITY % List.class        "| %ul.qname;
| %ol.qname;
| %dl.qname; ">

<!ENTITY % Table.class       "| %table.qname; ">

<!ENTITY % Misc.class        "| %script.qname;
| %noscript.qname; ">

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--           Optionally Empty Content Group Class Extras           -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % Block.extra       "%Table.class;
| %form.qname;
| %fieldset.qname; ">

<!ENTITY % BlkNoForm.extra   "%Table.class; ">

<!ENTITY % BlkNoTable.extra  "| %form.qname;
| %fieldset.qname; ">

<!ENTITY % Inline.extra     "">

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```

```

<!--          Aggregate Content Group Classes          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % Block.class      "%BlkStruct.class;
                             %BlkPhras.class;
                             %BlkPres.class;
                             %Block.extra;">

<!ENTITY % BlkNoForm.class  "%BlkStruct.class;
                             %BlkPhras.class;
                             %BlkPres.class;
                             %BlkNoForm.extra;">

<!ENTITY % BlkNoTable.class "%BlkStruct.class;
                             %BlkPhras.class;
                             %BlkPres.class;
                             %BlkNoTable.extra;">

<!ENTITY % Inline.class     "%InlStruct.class;
                             %I18n.class;
                             %InlPhras.class;
                             %InlPres.class;
                             %InlSpecial.class;
                             %InlForm.class;
                             %Inline.extra;
                             %Anchor.class;">

<!ENTITY % InlNoAnchor.class "%InlStruct.class;
                             %I18n.class;
                             %InlPhras.class;
                             %InlPres.class;
                             %InlSpecial.class;
                             %InlForm.class;
                             %Inline.extra;">

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--          Content Group Mixes          -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % Block.mix        "%Block.class;
                             %List.class;
                             %Heading.class;
                             %Misc.class;">

<!ENTITY % BlkNoForm.mix    "%BlkNoForm.class;
                             %List.class;
                             %Heading.class;
                             %Misc.class;">

<!ENTITY % BlkNoTable.mix   "%BlkNoTable.class;
                             %List.class;
                             %Heading.class;
                             %Misc.class;">

```

```

<!ENTITY % HeadOpts.mix      "(%script.qname;
                               |%style.qname;
                               |%meta.qname;
                               |%link.qname;
                               |%object.qname;)*">

<!ENTITY % Flow.mix          "%Block.class;
                               %List.class;
                               %Heading.class;
                               |%Inline.class;
                               %Misc.class;">

<!ENTITY % FlowNoTable.mix   "%BlkNoTable.class;
                               %List.class;
                               %Heading.class;
                               |%Inline.class;
                               %Misc.class;">

<!ENTITY % Inline.mix        "%Inline.class;
                               %Misc.class;">

<!ENTITY % InlNoAnchor.mix   "%InlNoAnchor.class;
                               %Misc.class;">

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--           Element Content Model Predeclarations                       -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

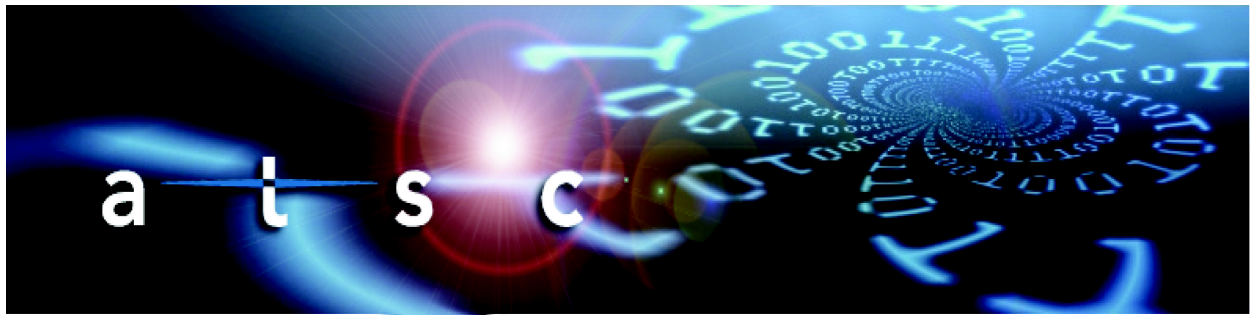
<!ENTITY % html.content      "( %head.qname;,
                               ( %body.qname; | %frameset.qname; ) )">

<!ENTITY % noscript.content  "( #PCDATA | %Flow.mix; )*>

<!ENTITY % blockquote.content "( #PCDATA | %Flow.mix; )*>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--           END END END                                               -->
<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

```



advancedtelevisionssystemscmmitteeinc

Advanced Television Systems Committee, Inc.
1750 K Street, N.W., Suite 1200
Washington, D.C. 20006