Doc. A/100-1
09 March 2003

DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1)
PART 1: INTRODUCTION, ARCHITECTURE, AND COMMON FACILITIES

ATSC Standard

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

Blank Page

Table of Contents

Table of Figures

Table of Tables

DASE-1 Introduction, Architecture, and Common Facilities

ATSC Standard

1. SCOPE

1.1 Status

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained by the ATSC.

This specification is an ATSC Standard, having passed ATSC Member Ballot on September 16, 2002. This document is an editorial revision of the Approved Proposed Standard (PS/100-1) dated November 5, 2002.

The ATSC believes that this specification is stable, that it has been substantially demonstrated in independent implementations, and that it defines criteria that are necessary for effective implementation and interoperability of Advanced Television Systems. A list of cumulative changes made to this specification may be found at the end of this document.

A list of current ATSC Standards and other technical documents can be found at http://www.atsc.org/standards.html.

1.2 Purpose

This document introduces a set of related specifications, henceforth referred to as the ATSC DTV Application Software Environment Level 1 (DASE-1) Standard, which jointly define an architecture and a collection of facilities by means of which DASE-1 applications may be delivered to and processed by a DASE-1 application environment embodied by a compliant receiver.[1]

> *Note:* In the following text and related specifications, *DASE-1* is frequently abbreviated to *DASE*; that is, *DASE* is to be construed as *DASE Level 1*.

A DASE 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.

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

> *Note:* Java, Java TV, and Java-based marks are trademarks of Sun Microsystems, Inc.

[1] The user'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 note was editorially updated on 11 June 2008.)

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

Application environments are similarly classified into two categories depending upon whether they process declarative or procedural applications. These categories are referred to as declarative and procedural application environments, respectively. An example of a declarative application environment is a multimedia document browser, also known as a user agent. An example of a procedural application environment is a Java Virtual Machine and its associated Application Programming Interface (API) implementation.

Note: The DASE Standard does not specify the implementation of application environments in a compliant receiver. A receiver manufacturer may implement both environments as a single subsystem; alternatively, both environments may be implemented as distinct subsystems with well-defined, internal inter-environment interfaces.

1.3 Application

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

1.4 Organization

This document is organized as follows:

- Section 1 Describes purpose and application of the DASE Standard; describes organization of this document
- Section 2 Enumerates normative and informative references
- Section 3 Defines acronyms, terminology, and conventions
- Section 4 Introduces DASE Standard
- Section 5 Specifies DASE Architecture
- Section 6 Specifies DASE Common Facilities
- Annex A Specifies document type definitions
- Annex B Enumerates content types supported by DASE Standard
- Annex C Describes minimum color support
- Annex D Depicts examples of application metadata and trigger entities
- Changes Cumulative changes to specification
- Acknowledgments

Unless explicitly indicated otherwise, all annexes shall be interpreted as normative parts of this specification.

2. REFERENCES

2.1 Normative References

The following documents contain provisions which, through reference in this document, constitute provisions of the DASE Standard. At the time of publication, the editions indicated were valid. All referenced documents are subject to revision, and parties to agreements based on the DASE Standard are encouraged to investigate the possibility of applying the most recent edition of the referenced document.

In case of any conflicts, the ATSC standards take precedence over the other normative references.

Note: The DASE Standard uses a reference notation based on acronyms or convenient labels for identifying a reference (as opposed to using numbers).

[A/52]

Digital Audio Compression Standard (AC-3), A/52, ATSC

[A/53]

ATSC Digital Television Standard, A/53, ATSC

[DASE-ZIP]

DASE-1 Part 5: ZIP Archive Resource Format, A/100-5, ATSC

[DOM2-EVENTS]

Document Object Model (DOM) Level 2 Events, Recommendation, W3C

[JPEG]

Digital Compression and Coding of Continuous-Tone Still Images, ISO 10918-1, ISO

[LANG-TAGS]

Tags for the Identification of Languages, RFC3066, IETF

[MIME]

Multimedia Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC2045, IETF

[MNG]

Multiple Network Graphics, Version 1.0, PNG Development Group

[PCM]

ITU-T G.711 Pulse Code Modulation of Voice Frequencies, ITU

[PFR]

Information technology – Generic digital audio-visual systems – Part 6: Information representation, Annex A, Coding of Outline Fonts, ISO/IEC 16500-6:1999, Annex A, ISO

[PNG]

Portable Network Graphics, Version 1.2, PNG Development Group

[SRGB]

Multimedia Systems and Equipment – Colour Measurement and Management – Part 2-1: Colour Management – Default RGB Colour Space – sRGB, IEC 61966-2-1 (1999-10), IEC

[UNICODE]

Unicode Character Encoding Standard, Version 3.2, Unicode Consortium

Note: Unicode is a trademark of Unicode, Inc. Unicode Consortium is a registered trademark of Unicode, Inc.

[URI]

Uniform Resource Identifiers: Generic Syntax, RFC2396, IETF

[URI-LID]

Declarative Data Essence – The Local Identifier (lid:) URI Scheme, SMPTE 343M, SMPTE

[URI-TV]

Uniform Resource Identifiers for Television Broadcasts, RFC2838, IETF

[UTF-8]

UTF-8, A Transformation Format of ISO 10646, RFC2279, IETF

[UUID]

Information Technology – Open Systems Interconnection – Remote Procedure Call (RPC), ISO/IEC 11578:1996, ISO

[XML]

Extensible Markup Language (XML) 1.0, Recommendation, W3C

[XMLNAMES]

Namespaces in XML, Recommendation, W3C

2.2 Informative References

[HTTP]

Hypertext Transfer Protocol – HTTP/1.1, RFC2616, IETF

[PNG-GUIDE]

PNG: The Definitive Guide, Greg Roelofs, 1999, O'Reilly & Associates, Inc., ISBN 1-56592-542-4

[SAFE]

Safe Action and Safe Titling Areas for Television Systems, RP27.3, SMPTE

2.3 Reference Acquisition

ATSC Standards

Advanced Television Systems Committee (ATSC), 1750 K Street N.W., Suite 1200 Washington, DC 20006 USA; Phone: +1 202 828 3130; Fax: +1 202 828 3131; http://www.atsc.org/.

IEC Standards

International Electrotechnical Commission (IEC), 3, rue de Varembé, Case postale 131, CH-1211 Geneva 20, Switzerland; Phone: +41 22 919 02 11; Fax: +41 22 919 03 00; http://www.iec.ch/.

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/.

ISO Standards

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

ITU Standards

International Telecommunication Union (ITU), Place des Nations, CH-1211 Geneva 20, Switzerland; Phone: +41 22 730 51 11; Fax: +41 22 733 72 56; http://www.itu.ch/.

PNG Standards

Portable Network Graphics (PNG) Development Group, c/o Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111, USA; Phone: +1 617 542 5942; Fax: +1 617 542 2652; http://www.libpng.org/pub/png/.

SMPTE Standards

Society of Motion Picture and Television Engineers, 595 W. Hartsdale Avenue, White Plains, NY 10607-1824, USA; Phone: +1 914 761 1100; Fax: +1 914 761 3115; http://www.smpte.org/.

Unicode Standards

The Unicode Consortium, P.O. Box 391476, Mountain View, CA 94039-1476, USA; Phone: +1 650 693 3921; Fax: +1 650 693 3010; http://www.unicode.org/.

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/.

3. DEFINITIONS

This section defines conformance keywords, acronyms, abbreviations, and terminology which apply to the DASE Standard in its entirety and to each constituent part.

3.1 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, which may or may not be present at the option of the application or the application environment implementer.

3.2 Acronyms and Abbreviations

API	Application Programming Interface
DA	Declarative Application
DAE	Declarative Application Environment
DASE	DTV Application Software Environment
DOM	Document Object Model
DTV	Digital Television
JPEG	Joint Photographic Expert Group
MIME	Multipurpose Internet Mail Extensions
MNG	Multiple Network Graphics
PA	Procedural Application
PAE	Procedural Application Environment
PNG	Portable Network Graphics
URI	Universal Resource Identifier
UUID	Universal Unique Identifier
XDML	Extensible DTV Markup Language
XML	Extensible Markup Language

3.3 Terms

active object content: a type of content which takes the form of an executable program; a compiled Java Xlet is an example of active object content.

> *Note:* See *DASE-1 Part 3: Procedural Applications and Environment*, Section 5.1, for information on *active object* content.

application: see *DASE Application*.

application abort: see *application termination*.

application activation: the process of transitioning an application's lifecycle state from the initialized to the active state, a process which entails decoding the application initial entity.

application delivery system: a mechanism by which an application is announced and signaled, and has its resources delivered to an application environment.

application delivery file system: an optional file system provided by the application delivery system; an application delivery file system may be mounted (logically attached) to a directory of the local file system; in general, all nodes (directories and files) of an application delivery file system are constrained to support only read access.

application emitter: the entity which controls the emission of applications through mechanisms implemented by an application delivery system; for example, a terrestrial broadcaster.

application entity: a unit of information that expresses some portion of an application.

application entity collection: a collection of application entities which expresses an application as a whole.

application environment: the context (system) in which an application is processed.

application initial entity: the application entity which is initially decoded during application activation processing.

> *Note:* The initial entity of an application is decoded after the application root entity is decoded.

application initialization: the process of transitioning an application's lifecycle state from the uninitialized to the initialized state, a process which entails decoding the application root entity.

application resource: a bit-stream serialization (a physical embodiment) of an application entity; an application resource may be of bounded (determinate) or unbounded (indeterminate) length; an application resource may be manifest or implied.

application resource collection: the set of application resources which embody an application entity collection.

application resumption: the process of transitioning an application's lifecycle state from the suspended to the active state.

application root entity: a specific element of an application entity collection which is processed before all other elements in the collection during application initialization processing.

application termination: the process of transitioning an application's lifecycle state from to the uninitialized state.

bounded resource: an application resource of determinate length.

content: an unspecified unit of information; the essential nature or character of some material; for example, streaming video content and markup content.

content type: a specific type of content identified by a MIME media type; a (metadata) property of an application resource.

content processor: an identifiable component of an application environment which decodes or executes a specific content type.

DASE Application: a collection of information that expresses a specific set of externally observable behavior.

> *Note:* In the DASE Standard, this term is generally abbreviated to *application*.

> *Note:* The definition of application specified here differs from the definition employed by the ATSC Data Broadcast Standard, A/90. The definition used by A/90 focuses on the collection of resources which constitute the embodiment of an application's entity set. In contrast, the focus within the DASE Standard addresses the functional nature of the application's behavior and the form of the information represented by the application's resources.

DASE content: a DASE Application or the content that composes the application.

DASE Extension: a well-defined set of functionality that extends the DASE Standard; an extension may be qualified as a *standard* extension, if defined by the ATSC, or a *non-standard* extension, if defined by a third party.

DASE level: a particular specification of the DASE Standard within a series of DASE Standards that specify functional supersets of prior DASE levels.

DASE receiver: a physical embodiment of the DASE System.

DASE Standard: the set of specifications, formally enumerated in the body of this specification (Part 1), that compose the ATSC Standard known as DASE (DTV Application Software Environment).

DASE System: a collection of logical components which supports the processing and presentation of DASE Applications.

DASE trigger: a bounded application resource which is asynchronously delivered to an active DASE Application; a DASE trigger is typically generated by an application emitter to cause some behavior in an active DASE Application.

data essence: content of an indeterminate type

declarative application: an application which primarily makes use of declarative information to express its behavior; an XDML document instance is an example of a declarative application.

declarative application environment: an environment that supports the processing of declarative applications; an XDML user agent (browser) is an example of a declarative application environment.

declarative information: information expressed in the form of assertions; e.g., *P* is, *Q* is, *R* is, or, more succinctly, {*P*, *Q*, *R*}.

end-user: the individual operating or interacting with a receiver.

entity set: alternative for *entity collection*.

environment: see *application environment*.

environment resource: a physical or logical component of an application environment; e.g., a region of the graphics frame buffer, an input device, a shared semaphore, a memory pool, etc.

facility: a non-empty collection of content types and their associated processors.

hybrid application: a hybrid declarative application or a hybrid procedural application.

hybrid declarative application: a declarative application that makes use of active object content; an XDML document with an embedded Java Xlet is an example of a hybrid declarative application.

hybrid procedural application: a procedural application that makes use of markup content; a Java Xlet that creates and causes the display of an XDML document instance is an example of a hybrid procedural application.

implied resource: an application resource whose content is not manifested directly to an application or application environment, but instead is visible only to the receiver platform.

legacy application: any application that is expressly marked as such in an application's metadata resource.

> *Note:* See Section 6.1.1.6.13.4 for more information on marking an application as a legacy application.

local file system: the file system provided by the local receiver platform.

manifest resource: an application resource whose content is manifested directly to an application or an application environment.

markup: text that is added to the primary information content of a document in order to convey information about that content.

markup language: a formalism that describes a class of documents which employ markup in order to delineate the document's structure, appearance, or other aspects; XDML is an example of a markup language.

markup content: a type of content which takes the form of a markup language; an XDML document is an example of markup content.

metadata: information about data essence; a type of content which describes content. Metadata may also be construed as data essence in certain contexts; that is, the relationship between metadata and data essence is mutually recursive.

MIME media type: a specification of a type of essence, the syntax of which is defined by [MIME].

native application: an intrinsic function implemented by a receiver platform; a close captioning display is an example of a native application.

native environment: see *receiver platform*.

persistent file system: see *local file system*.

procedural application: an application which primarily makes use of procedural information to express its behavior; a non-empty set of compiled Java Xlets is an example of a procedural application.

procedural application environment: an environment that supports the processing of procedural applications; a Java Virtual Machine and its public APIs constitute an example of a procedural application environment.

procedural information: information expressed in the form of procedures; e.g., do *F*, do *G*, do H, or, more succinctly, <*F*(), *G*(), *H*()>

receiver: see *DASE receiver*.

receiver platform: a physical embodiment of hardware, operating system, and native applications of the manufacturer's choice, which collectively constitute a receiver.

resource: an application resource or an environment resource.

resource identifier: an identifier which labels an application resource; e.g., a URI.

resource reference: the use of a resource identifier to refer to an application resource.

transport stream: an MPEG-2 Transport Stream, as defined by ISO/IEC 13818-1.

trigger: see *DASE trigger*.

Note: In the DASE Standard, the term *trigger* is used interchangeably with *DASE trigger*.

unbounded resource: an application resource of indeterminate length; e.g., a data stream.

4. INTRODUCTION

This section introduces the DASE Standard, DASE Content, and the DASE System.

4.1 DASE Standard

This section introduces the DASE Standard. The DASE Standard is organized as a set of related parts, each of which covers a distinct aspect or function of DASE:

- Part 1 Introduction, Architecture, and Common Facilities
- Part 2 Declarative Applications and Environment
- Part 3 Procedural Applications and Environment
- Part 4 Application Programming Interface
- Part 5 ZIP Archive Resource Format
- Part 6 Security
- Part 7 Application Delivery System – ARM Binding
- Part 8 Conformance

4.1.1 Introduction, Architecture, and Common Facilities

ATSC A/100-1, *DASE-1 Part 1: Introduction, Architecture, and Common Facilities*, introduces the DASE Standard, defines the DASE Architecture, and specifies Common Facilities which must be processed by both the DASE declarative application environment and the DASE procedural application environment.

> *Note:* The distinction between declarative and procedural application environments is for expository purposes only. No distinction need occur in a given implementation. In particular, an implementation of the DASE System may use the same implementation to process a common facility content type, no matter whether that content type is referenced from a declarative application or from a procedural application.

4.1.2 Declarative Applications and Environment

ATSC A/100-2, *DASE-1 Part 2: Declarative Applications and Environment*, specifies all facilities which are specifically processed by the DASE *declarative application environment*.

4.1.3 Procedural Applications and Environment

ATSC A/100-3, *DASE-1 Part 3: Procedural Applications and Environment*, specifies all facilities which are specifically processed by the DASE *procedural application environment*.

4.1.4 Application Programming Interface

ATSC A/100-4, *DASE-1 Part 4: Application Programming Interface*, specifies the syntax and semantics of the DASE specific APIs exposed to DASE Procedural Applications.

4.1.5 ZIP Archive Resource Format

ATSC A/100-5, *DASE-1 Part 5: ZIP Archive Resource Format*, specifies an archive content type supported by the common facilities.

4.1.6 Security

ATSC A/100-6, *DASE-1 Part 6: Security*, specifies all facilities which relate to the common security aspects of DASE Applications and DASE Systems.

4.1.7 Application Delivery System – ARM Binding

ATSC A/100-7, *DASE-1 Part 7: Application Delivery System – ARM Binding*, specifies all facilities which relate to the binding of DASE Applications and the DASE System to the ATSC Data Application Reference Model (A/94), which, in turn, specifies an application delivery system which employs the ATSC Data Broadcast Standard (A/90).

4.1.8 Conformance

ATSC A/100-8, *DASE-1 Part 8: Conformance*, specifies the overall conformance requirements for DASE Applications and DASE Systems.

4.2 DASE Content

DASE Content is generally organized as a collection of one or more DASE Applications, each of each of which takes the form of either a declarative application or a procedural application.

Either type of DASE Application may make use of facilities of both declarative and procedural application environments.

A DASE Application may be designed to work in a standalone fashion or in concert with a collection of cooperating applications.

4.2.1 Declarative Applications

A DASE *declarative application* (DA) is a DASE Application whose *initial entity* is the specific *markup content* type `application/xhtml+xml`. In addition to markup content, a DASE declarative application may contain stylesheet and script content as well as other content types.

4.2.2 Procedural Applications

A DASE *procedural application* (PA) is a DASE Application whose *initial entity* is the specific *active object content* type `application/javatv-xlet`. In addition to active object content, a DASE procedural application may contain archive and application defined content as well as other common content types.

4.3 DASE System

The DASE System interacts with receiver platform services in order to accept input from the broadcast transport and the end-user and generate graphics and audio output for presentation on the receiver platform's display and audio rendering systems. The receiver platform provides essential services to the DASE System such as operating system services, input/output services, and memory services. See Figure 1 DASE System Interconnect.

> *Note:* Synchronous audio and video decoder pipelines are expected to be implemented in the receiver platform services layer, and not within the DASE System layer. The DASE System does not have direct visibility of the raw broadcast transport stream or the program elements contained therein. Nevertheless, the receiver platform may provide services to the DASE System that permit a DASE Application to exert limited control over the selection and processing of the broadcast transport and its program elements.

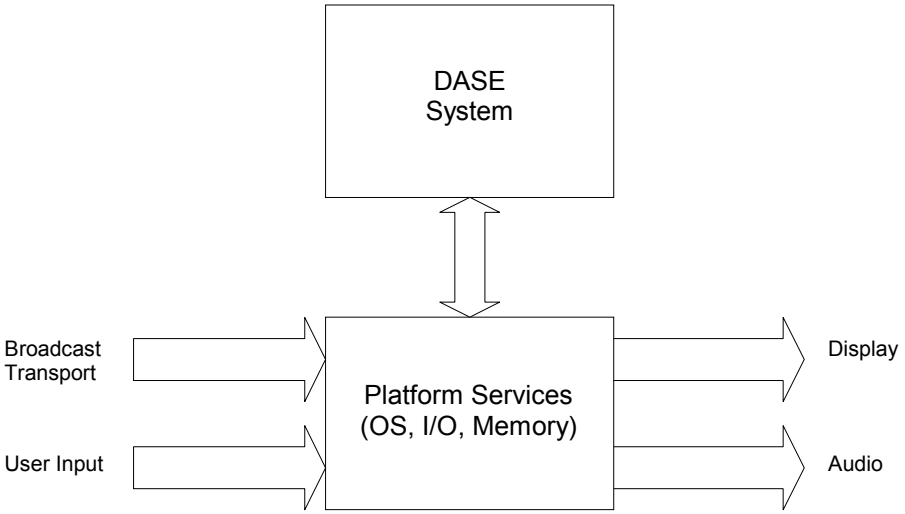

Figure 1 DASE System Interconnect

4.3.1 Declarative Application Environment

A DASE *declarative application environment* (DAE) is a logical subsystem of the DASE System which processes markup, stylesheet, and script content. A key component of the declarative application environment is the *declarative content decoding engine* (DCDE), which takes the form of an XDML parser and a stylesheet and script interpreter.

4.3.2 Procedural Application Environment

A DASE *procedural application environment* (PAE) is a logical subsystem of the DASE System which processes active object content. A key component of the procedural application environment is the *procedural content execution engine* (PCEE), which, for example, may take the form of a Java Virtual Machine.

5. ARCHITECTURE

This section defines the architecture for DASE Applications and the DASE System in which these applications are processed. This architecture is specified in terms of the following models:

- DASE Content Model (DASE Applications)
- DASE Environment Model (DASE System)

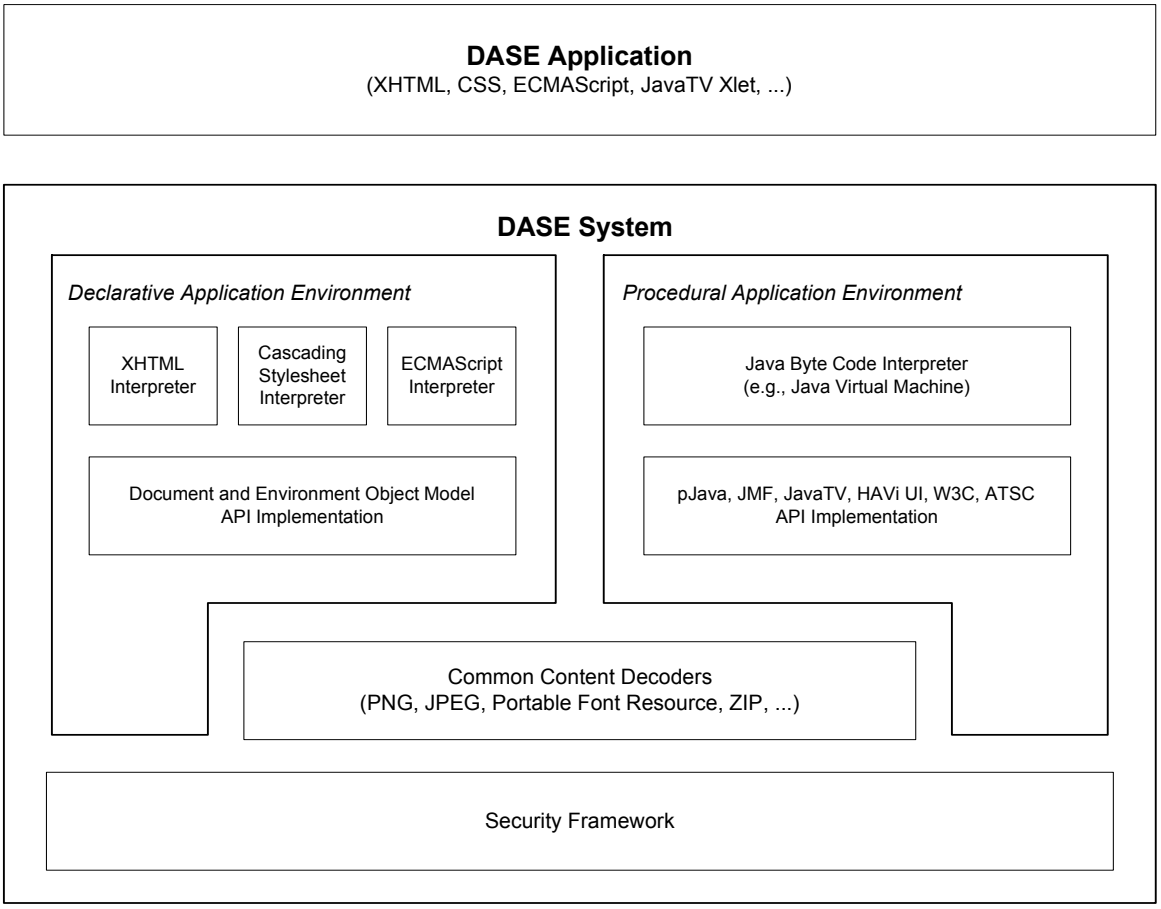

Figure 2 DASE Architecture

The overall DASE Architecture is shown in Figure 2 DASE Architecture.

The Declarative Application Environment processes declarative applications. The Procedural Application Environment processes procedural applications. Common content decoders serve both procedural and declarative application needs for the decoding and presentation of common content types such as PNG, JPEG and Portable Font Resource formats. The Java Byte Code Interpreter (a Java Virtual Machine) serves as the Procedural Content Execution Engine and is a part of the Procedural Application Environment. Java Application Programming Interfaces (APIs) provide procedural applications with access to the receiver's functions.

This architecture serves as a reference architecture intended to be supported by all DASE Systems. A DASE System may implement each component of the DASE architecture described above in any manner such that the component behaves and performs its functions as specified by the DASE Standard.

5.1 DASE Content Model

The following sections define a *content model* by means of which DASE Applications are organized, delivered (interchanged), and processed.

5.1.1 Application Structure

A *DASE Application*, hereafter abbreviated as *application*, is a collection of information that expresses a specific set of externally observable behavior.

An application is organized as an *application entity collection* each member of which is represented as an *application resource*. An *application resource* is the physical embodiment of an *application entity* which can be construed as a logical rather than physical unit of information.

> *Note:* In the present context, a physical embodiment means a bit (or octet) string which denotes some logical information whose meaning is defined by the application.
>
> *Note:* An *application entity* may also be characterized as a unit of *data essence*. All information comprising an application, whether *code* or *data* referenced by code, is considered to be *data essence*. In this context, *code* may consist of either declarative or procedural information.

An application's entities can be characterized as a rooted, possibly cyclical graph, where the individual entities are nodes and references to entities are arcs. The root node of this graph is referred to as the *application root entity*. The physical embodiment of this entity is referred to as the *application root resource*.

An application entity's *content type* is a specification of the semantics and the bit-serial syntax (i.e., the serialization) of the information represented by the entity. A *declarative content type* is primarily concerned with the representation of *declarative information*. A *procedural content type* is primarily concerned with the representation of *procedural information*.

The universe of applications may be partitioned into a set of declarative applications and a set of procedural applications. A *declarative application* is an application whose initial entity is of a declarative content type. A *procedural application* is an application whose initial entity is of a procedural content type. A *purely declarative application* is one whose every entity is of a declarative content type. A *purely procedural application* is one whose every entity is of a procedural content type. A *hybrid application* is one whose entity set contains entities of both declarative and procedural content types.

A *hybrid declarative application* is an application whose initial entity is of a declarative content type and whose entity set consists of entities of both declarative and procedural content types. A *hybrid procedural application* is an application whose initial entity is of a procedural content type and whose entity set consists of entities of both procedural and declarative content types.

5.1.2 Application Delivery

An application is delivered from an *application emitter* to the application environment by means of the interchange of (1) announcement metadata, (2) signaling metadata, and (3) data essence in the form of application resources, where this interchange occurs through an *application delivery system*.

5.1.2.1 Application Announcement

The future availability of an application for possible processing and presentation by an application environment should be announced by an application delivery system. The intention of application announcement is to notify an end-user of the future availability of the application so that the end-user can take an appropriate action to schedule use of the application.

The announcement of an application may be accompanied by the following metadata which describes the application:

- Application Identifier
- Application Name
- Application Description
- Application Level
- Application Availability Time and Duration
- Application Rating Information
- Application Language(s)

To the extent that this metadata is composed of textual information expected to be presented to an end-user, it shall make use of a character encoding system supported by the universal character set [UNICODE]. Furthermore, it should be capable of distinguishing multiple instances of such types of metadata according to distinct natural languages.

> *Note:* A character encoding system is supported by [UNICODE] if content encoded by that character encoding system has an equivalent representation in [UNICODE] and if that content can be transcoded both to and then from [UNICODE] without loss of information.

5.1.2.2 Application Signaling

The imminent arrival or current availability of an application for processing and presentation by an application environment shall be signaled by means of the application delivery system.

The signaling of an application shall be accompanied by an application metadata resource which describes the application as a whole as well as the resources which embody the application's entity set. This metadata resource shall specify the following:

- Application Identifier
- Application Level
- Application Initial Entity Resource Identifier

The signaling of an application may be accompanied by application metadata which describes the following information:

- Application Permission Entity Resource Identifier
- Application Parameters

It is not expected that this metadata will be presented to an end-user.

The specification of application metadata shall take the form defined in Section 6.1 below.

If the user has previously selected the application for instantiation or the application environment determines the application must be instantiated automatically without user intervention (e.g., based on application invocation directives), then this signaling information shall be provided to the application environment to effect application instantiation.

An application environment shall instantiate an application by (1) pre-allocating an initial set of environment resources (e.g., graphics frame buffer regions, input devices, etc.) and (2) initiating processing of the application root entity resource.

> *Note:* See Section 5.1.3, Application Lifecycle, for more information on application instantiation processing.

5.1.2.3 Application Resources

An application resource is a physical embodiment (bit-string serialization) of an application entity. Each application resource shall consist of the following information:

- Resource Identifier

- Resource Content Type
- Resource Content

Application resource metadata may include the following information:

- Resource Cache Directives
- Resource Last Modified Time

A resource shall be identified by the application delivery system with an identifier which takes the form of an absolute *universal resource identifier* as prescribed by [URI]. This identifier shall be unique within the scope of a single application's entity set.

> *Note:* The interpretation of *unique* in this context means that two application resources are not labeled with the same identifier within a single application

A resource content type shall take the form of a MIME media type. The syntax of a content type specification shall adhere to the following production, where the terms `type`, `subtype`, and `parameter` are defined in accordance with [MIME], Section 5.1, *Syntax of the Content-Type Header Field*:

```
content-type : type '/' subtype (';' parameter)*
```

A DASE System shall be capable of parsing any syntactically valid content type specification; any parameter whose support is not required by the DASE Standard may be ignored. Unless indicated to the contrary, a content type supported by the DASE Standard does not require support for a content type specification parameter.

Resource content may be of bounded or unbounded types. Bounded resource content shall take the form of an octet-string from 0 to 2^{32}-1 octets in length. Unbounded resource content shall take the form of an unbounded octet-string.

If present, resource cache directives shall take the form defined in Section 6.1.1.6.2.1 below.

If present, the resource last modified time shall take the form of a 64-bit, signed integer which specifies the number of milliseconds from 00:00:00 GMT Jan 1, 1970.

5.1.2.3.1 Resource Identifiers

A DASE Application may use and a DASE System shall support the syntax and semantics of the following URI scheme types as further described in the following subsections.

- `"archive:"`
- `"ecmascript:"`
- `"lid:"`
- `"tv:"`

Constraints on the use of resource identifiers to identify actual resources delivered by an application delivery system are specified by the application delivery system binding.

> *Note:* See *DASE-1 Part 7: Application Delivery System – ARM Binding* , Section 4.1.3.1, *Resource Identifiers*, for more information on the usage of resource identifiers.

5.1.2.3.1.1 Archive Identifier Scheme

A URI which employs the *archive* identifier scheme shall adhere to the following syntax:

```
archive_URI : "archive:" restricted_URI "!" abs_path [ "#" fragment ]
```

The syntactic token `restricted_URI` shall adhere to the following constraints after unescaping any escaped byte tokens:

(1) it shall adhere to the syntactic token *absoluteURI* or *relativeURI* prescribed by [URI];

(2) it shall adhere to a scheme whose use is permitted by this specification;
(3) it shall not take the form of an archive identifier;
(4) it shall not contain a *query* or *fragment* component;
(5) it shall not contain the substring `"!"`.

If `restricted_URI` takes the form of a *relativeURI*, then the absolutized form of this URI shall adhere to these restrictions.

The syntactic tokens `abs_path` and `fragment` shall adhere to the syntax prescribed by [URI], Annex A. In addition, `abs_path` shall, after removing its initial segment separator (`"/"`), adhere to the syntax of an archive entry's pathname.

The semantics of resolving an *archive* identifier are: (1) resolve `restricted_URI` to a resource of an archive content type, (2) resolve `abs_path` to an entry within the archive resource, and (3) if `fragment` is specified, resolve to the referenced fragment within the archive entry.

Example: The following specifies an example of an *archive* identifier used to reference a specific element within an XDML document within a ZIP archive. This identifier uses an embedded absolute identifier to reference the archive resource.

```
archive:lid://xyz.com/app/app.zip!/top/doc.xml#gohere
```

Example: The following specifies an example of an *archive* identifier used to reference a Java class file within a JAR archive. This identifier uses an embedded relative identifier to reference the archive resource.

```
archive:app.jar!/com/xyz/MyXlet.class
```

> *Note:* See Section 6.8.1 for more information on the ZIP archive content type. See *DASE-1 Part 3: Procedural Applications and Environment*, Section 5.4.1, for more information on the JAR archive content type.

5.1.2.3.1.2 Ecmascript Identifier Scheme

In certain circumstances, a legacy declarative application may use a URI which employs the *ecmascript* URI scheme.

> *Note:* See *DASE-2 Part 2: Declarative Applications and Environment*, Section 4.7, for the definition of the *ecmascript* URI scheme.

5.1.2.3.1.3 Local Identifier Scheme

A URI which employs the *local* identifier scheme shall adhere to the syntax prescribed by *The Local Identifier (lid:) URI Scheme* [URI-LID].

A URI which employs this scheme may reference the following types of application resources:

- a non-streaming, bounded application resource
- a streaming, unbounded application resource

5.1.2.3.1.4 Television Scheme

A URI which employs the *tv* identifier scheme shall adhere to *Uniform Resource Identifiers for Television Broadcasts* [URI-TV].

A URI which employs this scheme may specify a query component in accordance with [URI], Section 3, *URI Syntactic Components*. If no query component is specified, then the URI shall resolve to a virtual channel (service); if a query component is specified, then it shall resolve to a part of a program element (component) of a virtual channel, an entire program element of a virtual channel, or an aggregate of program elements (components) of a virtual channel.

The form of a query component shall adhere to the following syntax:

```
query           : query_component [ ";" query_component ]*
query_component : component [ "=" component_name ]
component       : "video" | "audio" | "dtvcc"
component_name  : pchar+
pchar           : { any printable character except ';' }
```

A query which specifies a `video` component resolves to either a video elementary stream or an aggregate of program elements which includes a video elementary stream.

A query which specifies an `audio` component resolves to either an audio elementary stream or an aggregate of program elements which includes an audio elementary stream.

A query which specifies a `dtvcc` component resolves to the digital television closed captioning private user data portion of a video elementary stream.

When specified, a component name is used to resolve to a part or whole of a particular program element which is labeled with the component name.

5.1.2.3.2 Resource References

A resource reference is employed by applications to refer to a resource, by which is usually meant the resource's content and associated content type.

5.1.2.3.2.1 Relative Resource Identifiers

A resource identifier employed by a resource reference may take a relative or an absolute form. If it takes a relative form, then it is always possible to *absolutize* this form by resolving the relative identifier with respect to either (1) the absolute resource identifier of the resource in whose context the reference occurs, or (2) resource internal information, such as a *base* element in markup content. A resource identifier which is used to absolutize a relative identifier is referred to as a *base identifier*. A base identifier shall take the form of an absolute identifier.

> *Note:* To *absolutize* a resource identifier means to transform a relative identifier into an absolute identifier.

Given a relative identifier, the base identifier to be used to absolutize the relative identifier shall be determined by the following ordered rules. The first rule that produces a base identifier which permits resolving the relative identifier shall be used.

(1) use an application environment, context dependent identifier;
(2) use the identifier of the resource which contains the relative identifier reference;
(3) use the identifier of the application's root resource.

> *Note:* See *DASE-1 Part 2: Declarative Applications and Environment* , Section 4.6, *Relative Identifier Resolution*, and *DASE-1 Part 3: Procedural Applications and Environment* , Section 4.4, *Relative Identifier Resolution*, for more information on application environment, context dependent rules for determining a base identifier.

5.1.2.3.3 Resource Availability

During the processing of an application, references are made to the resources constituting its resource collection. Some of these references occur synchronously during the initial decoding of the application's resources, while others occur asynchronously as a result of end-user initiated actions (e.g., activating a hypertext link). When a resource reference is processed by the application environment, the resource must be partially or fully available; that is, the data essence that constitutes the resource must be available to some degree.

> *Note:* In this section, the terms *available* and *availability* refer to the ability of the DASE System to directly access the content of an application resource.

The DASE Standard does not define the degree to which a resource must be available for it to be processed by the application environment. In general, if a resource is not available when

referenced, the application environment should take whatever measures are necessary to obtain the resource. During this process, some feedback may be provided to the end-user to indicate that additional steps are being taken to obtain the resource.

> *Note:* An implementation of an application environment may take a conservative approach to resource availability, requiring all resources to be fully available prior to permitting the application to be instantiated; conversely, an application environment may take a lazy approach, requiring only that resource metadata be available prior to application instantiation. In the latter case, the application environment may suspend the application or provide end-user feedback while referenced, but unavailable resources are being obtained.
>
> *Note:* Limited control over pre-caching of resources may be obtained by use of application metadata *cache* elements as described below in Section 6.1.1.6.2.

An application shall not rely upon the availability of an application resource which is not part of that application.

5.1.2.3.4 Resource Caching

An application entity may be cached by a DASE System by saving an entity's resource for future re-use. An application entity shall not rely upon a resource being cached by a DASE System.

An application environment shall observe the semantics of any cache directives that accompany a resource. The semantics of `no-cache` and `no-store` directives shall be supported by the environment.

> *Note:* Certain content types permitted by the DASE Standard are not cacheable, irrespective of the presence of cache directives that accompany a resource. This restriction is explicitly marked in the content type descriptions that follow below or in the other documents composing the DASE Standard.

5.1.3 Application Lifecycle

This section specifies the states and events which affect an application's lifecycle, a diagram of which is shown in Figure 3 Application State Diagram.

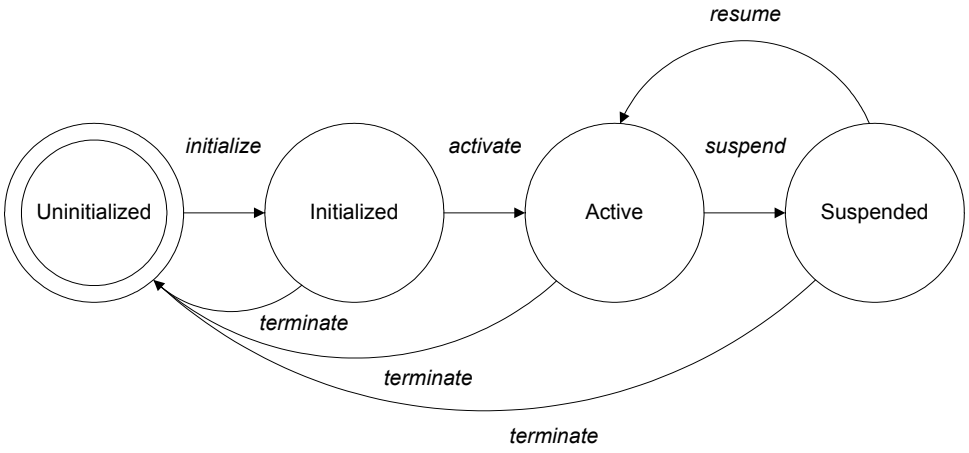

Figure 3 Application State Diagram

5.1.3.1 States

A DASE Application exhibits a well-defined lifecycle characterized by the following states:

- active
- initialized
- suspended
- uninitialized

An application's lifecycle state is affected by the application delivery system's state; furthermore, an application's lifecycle state affects the state of application environment dependent state in accordance to the type of application: declarative or procedural.

> Note: See DASE-1 Part 7: Application Delivery System – ARM Binding for information on the mapping of application delivery state to DASE Application lifecycle state. See DASE-1 Part 2: Declarative Applications and Environment and DASE-1 Part 3: Procedural Applications and Environment for information on the mapping of DASE Application lifecycle state to application environment dependent state.

A DASE System shall maintain the current state of a DASE Application with respect to this lifecycle state model. This state is not directly exposed to an application, but can be inferred through events dispatched to the application or actions performed on the application by the DASE System.

5.1.3.1.1 active state

An application shall be transitioned from the *initialized* state to the *active* state upon the successful decoding of its *initial entity*. This transition is referred to as *application activation*.

An application shall be transitioned from the *suspended* state to the *active* state upon the resumption (activation) of any application-defined thread. This transition is referred to as *application resumption*.

While in the *active* state, an application may consume any environment resource.

No more than one DASE Application shall be in the *active* state at a given time within a DASE System.

5.1.3.1.2 initialized state

An application shall be transitioned from the *uninitialized* state to the *initialized* state upon the receipt of an *initialize* event and the subsequent successful decoding of its *root entity*. This transition is referred to as *application initialization*.

> *Note:* An *initialize* event is generated in response to an application load request.

While in the *initialized* state, an application may consume any environment resource other than an application instantiated thread.

No more than one DASE Application shall be in the *initialized* state at a given time within a DASE System.

5.1.3.1.3 suspended state

An application shall be transitioned from the *active* state to the *suspended* state upon the receipt of a *suspend* event and the subsequent successful release of all exclusive environment resources and suspension of all application instantiated threads. This transition is referred to as *application suspension*.

> *Note:* A *suspend* event is generated by either the application delivery system or the DASE System in response to implementation dependent conditions. No mechanism is provided for a DASE Application to suspend itself.

While in the *suspended* state, an application environment shall cause or shall insure that an application has released all exclusive environment resources and suspended all application instantiated threads.

Multiple DASE Applications may be in the *suspended* state at a given time within a DASE System. If a DASE System does not have sufficient resources to maintain a DASE Application in a suspended state, it may cause the suspended application to transition to the *uninitialized* state.

5.1.3.1.4 uninitialized state

An application shall begin and end its lifecycle in the *uninitialized* state.

If an application is not in an *active*, *suspended* or *initialized* state, then it shall be in the *uninitialized* state.

An application shall be transitioned from the *initialized* state to the *uninitialized* state upon (1) a failure to decode the application's initial entity, (2) an occurrence of an uncaught exception during the processing of the application's initial entity or (3) upon the receipt of a *terminate* event.

> *Note:* A *terminate* event is generated by either (1) the application delivery system or (2) the DASE System in response to either application initiated or implementation dependent conditions.

An application shall be transitioned from the *active* state to the *uninitialized* state upon (1) an occurrence of an uncaught exception during the processing of an application entity or (2) the receipt of a *terminate* event.

An application shall be transitioned from the *suspended* state to the *uninitialized* state upon the receipt of a *terminate* event.

Any transition to the uninitialized state is referred to as *application termination*.

While in the *uninitialized* state, an application shall not consume any environment resource. A DASE System may consume cache resources to retain an application's resources while it is in an *uninitialized* state.

> *Note:* The determination of whether or not to cache an uninitialized application's resources is not defined by this specification.

Multiple DASE Applications may be in the *uninitialized* state at a given time within a DASE System.

5.1.3.2 Events

A DASE Application's lifecycle is affected by the following internally or externally generated events.

- activate
- initialize
- suspend
- resume
- terminate

5.1.3.2.1 activate event

An *activate* event shall be generated by the DASE System in response to successful application initialization.

> *Note:* No mechanism is provided for an application to activate itself.

5.1.3.2.2 initialize event

An *initialize* event shall be generated by (1) the application delivery system or (2) the DASE System in response to an application load request.

Note: An *application load request* is generated internally within a DASE Application Environment as a side effect of certain operations that cause an application to be replaced by a new application.

5.1.3.2.3 resume event

A *resume* event may be generated by (1) the application delivery system or (2) the DASE System in response to an implementation dependent request.

Note: No mechanism is provided for an application to resume itself.

5.1.3.2.4 suspend event

A *suspend* event may be generated by (1) the application delivery system or (2) by the DASE System in response to an implementation dependent request.

Note: No mechanism is provided for an application to suspend itself.

5.1.3.2.5 terminate event

A *terminate* event may be generated by (1) the application delivery system or (2) by the DASE System in response to an application initiated request, an abort condition, or an implementation dependent condition.

Note: The process of generating a terminate event is also referred to as *aborting* an application. Termination due to an abort condition is referred to as *abnormal termination*; otherwise, termination is referred to as *normal termination*.

5.2 DASE Environment Model

The following sections define an *environment model*, an implementation of which DASE Applications are delivered to and thereby processed. A DASE System shall implement the environment model defined below either directly or indirectly through mechanisms provided by the underlying receiver platform.

5.2.1 User Input Capabilities

A DASE System shall support interaction with content according to the facilities defined by this standard. A DASE System shall support this interaction by means of a navigation device that permits the end-user to enter the following gestures:

- move left, right, up, down
- digits zero through nine
- activate (select or enter)
- color 0, color 1, color 2, color 3

The determination of which of these gestures are dispatched as events and when they are dispatched to a DASE Application shall be implementation dependent.

Note: It is recommended that color gestures 0 through 3 be associated with the colors *red*, *green*, *blue*, and *yellow*, respectively.

In addition to these capabilities, a DASE System shall support either a virtual keyboard or a physical keyboard device minimally capable of entering (1) all printable ASCII characters (0x21 through 0x7E, inclusive), SPACE (0x20), and HORIZONTAL TABULATION (0x09), and (2) a platform dependent representation of the *newline* function ('\n').

A DASE System may support a pointer (cursor) navigation device, e.g., a mouse or a trackball.

5.2.2 Audio Capabilities

A DASE System shall support the real-time decoding and presentation of streaming audio content to an audio device in accordance with [A/52] as constrained by [A/53].

Note: See Sections 6.4 and 6.6 for non-streaming and streaming audio content types, respectively.

Note: The mechanism and parameters for accomplishing audio presentation are not defined by this specification.

A DASE System need not support more than one audio decoding pipeline; furthermore, a DASE Application shall not rely upon the presence of support for multiple audio decoding pipelines.

5.2.3 Video Capabilities

A DASE System shall support the real-time decoding and presentation of streaming video content on a video plane in accordance with [A/53].

Note: See Sections 6.3 and 6.5 for non-streaming and streaming video content types, respectively.

A DASE System need not support more than one video decoding pipeline; furthermore, a DASE Application shall not rely upon the presence of support for multiple video decoding pipelines.

5.2.4 Graphics Capabilities

A DASE System should support the decoding and presentation of non-video, visual content on a graphics plane in accordance with one of the following resolutions:

- 1920 x 1080
- 1280 x 720
- 960 x 540

A DASE System shall support the following graphics plane resolution:

- 640 x 480

Note: See Section 5.2.5.4.1 for further requirements regarding display format support.

This graphics plane should support one or more of the following color models.

- 8-bit pseudo-color
- 16-bit direct color (RGBA4444 or RGBA5551)
- 24-bit direct color (RGBA6666 or RGBA8880)
- 32-bit direct color (RGBA8888)

Note: In the notation RGBA*rgba*, *r* specifies the number of bits in the red (R) sample, and so on with green (G), blue (B), and alpha (A).

Note: A receiver platform may make use of any specific color space. The use of RGBA serves as a reference color model for the DASE System architecture. It is expected that the receiver platform will be able to translate from this reference color space to the actual color space employed in the implementation.

A DASE System shall provide sufficient native font support for the rendition of those glyphs necessary to depict the subset of [UNICODE] characters in the range U+0000 to U+00FF (ISO Latin 1).

Note: Font resources needed to present other [UNICODE] characters not in this subset may be provided by content authors by including the necessary font content in the application's resource collection. See Section 6.7 for further information.

5.2.5 Display Model

A DASE System shall support the conceptual display model as described in the following subsections.

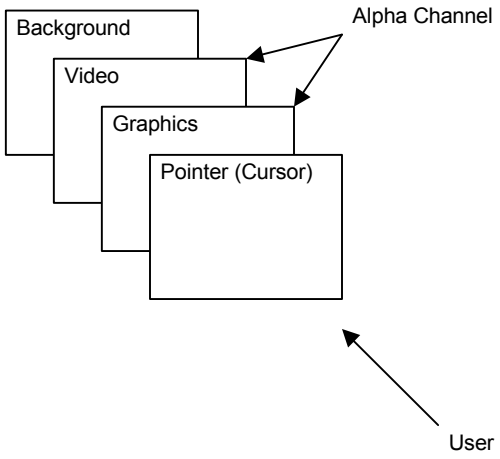

Figure 4 Drawing Model

5.2.5.1 Drawing

The display model permits drawing into four conceptual planes: a background plane, a video plane, a graphics plane and an optional pointer (cursor) plane. These planes are combined to produce the image that the end-user sees. The planes are ordered as shown in Figure 4 Drawing Model.

The video and graphics planes are blended with an alpha channel value as further described below.

5.2.5.2 Background Plane

The background plane shall support the setting of a background color which the screen will display when the other planes are disabled or otherwise not visible.

5.2.5.3 Video Plane

The video plane shall support the video output produced by [A/53]. The video plane shall be combined with the graphics plane using the alpha channel as described in Section 5.2.5.6.

5.2.5.3.1 Scaling and Translation

The video plane shall be capable of scaling to quarters (two by two) and ninths (three by three) of the current video format resolution. In addition, a DASE System should be capable of continuous scaling through the resolution of the video format.

The scaled video shall be capable of translation to any point such that the resulting video remains fully visible. Translation of video to off-screen coordinates is not required.

5.2.5.4 Graphics Plane

The graphics plane shall support the pixel oriented graphics output of applications. The graphics plane shall be minimally capable of displaying the colors specified by ANNEX C.

5.2.5.4.1 Display Format

The graphics plane shall minimally support a display format of 640x480 or greater, with square pixels of depth of 8bpp (bits per pixel) or greater.

Note: See [SAFE] for guidelines on overrun and underrun.

5.2.5.4.2 Coordinate System

The graphics plane coordinate system origin shall be in the upper left, increasing rightwards in the horizontal axis and downwards in the vertical axis.

Note: This type of coordinate system is sometimes referred to as a *fourth quadrant* system.

5.2.5.4.3 Color Model

The graphics plane shall support a true color model in which all pixel values are expressed and returned as (R,G,B) triplets expressed in the sRGB color space defined in [SRGB]. In addition, it may optionally support other color models.

Note: This standard does not require the graphics plane to have the property that if an application were to write a particular value to a pixel it would read the same value back.

Note: The implementation of a receiver platform may use any color model; it is expected that it be able to translate color values between this reference model and the actual implementation.

5.2.5.4.4 Colorimetry

The colorimetry for the graphics plane need not match the colorimetry for the video plane.

5.2.5.4.5 Gamma and Chroma Correction

No gamma or chroma correction is required for the graphics plane.

5.2.5.4.6 Clipping and Redraw

A DASE System is responsible for clipping and redraw functions relative to the embodiment of the environment.

5.2.5.4.7 Interaction with Native Environment

In general, a DASE System is unaware of any native applications that also may choose to use the graphics plane. These include but are not limited to: closed captioning, conditional access (CA) system messages, receiver menus, and native program guides.

Native applications may take precedence over a DASE System. Closed captioning and emergency messaging shall take precedence over a DASE System.

Some native applications, such as closed captioning, present a special case where the native application may be active for long periods concurrently with a DASE System.

5.2.5.5 Pointer (Cursor) Plane

If a pointing device capability is supported, then a pointer (cursor) plane shall be supported. A minimum pointer (cursor) size of 32x32 pixels is recommended.

5.2.5.6 Alpha Channel

Rendering of a semi-transparent graphics plane on top of the video plane shall be supported through alpha blending, with minimally binary alpha values. The graphics and video planes shall be combined pixel-by-pixel using an alpha value where different pixels may have different alpha values. The graphics and video planes may have different formats and, therefore, pixel-by-pixel blending may require adjusting the resolution (or format) of the planes by means of an implementation dependent mechanism.

This standard does not specify the maximum number of bits used to represent alpha values. The minimum number of bits shall be one (a binary alpha value). A maximum alpha value shall be used to completely obscure the video and render only graphics (minimum transparency). Symmetrically, the minimum (zero) alpha value shall be used to display only a video pixel with no graphics (maximum transparency). Intermediate alpha values shall obscure the video pixels according to a linear model. With a normalized alpha in the interval `[0,1]`, this model implies that for an arbitrary pixel:

```
d = a * g  + (1 - a) * v
```

where v, g, and d are color vectors for video, graphics, and display, and a is the normalized alpha value.

5.2.5.7 Registration of Video and Graphics Planes

The registration of video and graphics planes is not defined by this specification.

6. COMMON FACILITIES

This document defines a number of facilities common to both declarative and procedural applications and application environments. Each facility defines a category of content types by enumerating a set of one or more specific content types. The following categories are defined:

- application metadata content
- graphics content
- non-streaming video content
- non-streaming audio content
- streaming video content
- streaming audio content
- font content
- archive content
- trigger content

Note: See ANNEX B for which facilities and content types must be supported by a DASE System.

6.1 Application Metadata Content

This facility consists of an application metadata content type. Application metadata content is used to specify essential information about a DASE Application which is required or useful in the instantiation and processing of the application.

The root entity of a DASE Application shall take the form of this content type.

6.1.1 application/dase

Application metadata content shall adhere to the *Extensible Markup Language, Version 1.0* [XML] and shall be identified as content type `application/dase`.

Furthermore, application metadata content shall adhere to the following document type, labeled here according to its formal public identifier, and defined in ANNEX A Document Type Definitions according to procedures set forth in [XML] and [XMLNAMES]:

- `"-//ATSC//DTD DASE Application Metadata 1.0//EN"`

6.1.1.1 Well Formedness

An application entity which employs this content type shall be well formed as prescribed by [XML], Section 2.1.

If an entity of an application uses content type `application/dase`, is not well formed, and the entity is processed, then a DASE System shall abort the application.

6.1.1.2 Validity

An application entity which employs this content type shall be valid as prescribed by [XML], Section 2.8.

If an entity of an application uses content type `application/dase`, is not valid, and the entity is processed, then a DASE System shall abort the application.

6.1.1.3 XML Declaration

An application entity which employs this content type shall specify a valid XML declaration [XML:23].

Note: The notation [XML:23] refers to [XML] grammar production 23.

If an entity of an application uses content type `application/dase`, does not specify a valid XML declaration, and the entity is processed, then a DASE System shall abort the application.

An application entity which employs this content type shall specify an XML declaration with an encoding declaration [XML:80] according to one of the following character encoding systems, and, furthermore, the entity's representation shall employ the specified encoding system as its actual character encoding system.

- `"UTF-8"`
- `"ISO-8859-1"`

If an entity of an application uses content type `application/dase`, does not specify one of the preceding encoding declarations in the XML declaration according to the actual employed character encoding system, and the entity is processed, then a DASE System shall abort the application.

An application entity which employs this content type shall not specify an XML declaration with a standalone document declaration [XML:32] with the value `"yes"`.

If an entity of an application uses content type `application/dase`, does specify a standalone document declaration with the value `"yes"`, and the entity is processed, then a DASE System shall abort the application.

6.1.1.4 Document Type Declaration

An application entity which employs this content type shall specify a valid document type declaration [XML:28].

If an entity of an application uses content type `application/dase`, does not specify a valid document type declaration, and the entity is processed, then a DASE System shall abort the application.

An application entity which employs this content type shall specify a document type declaration with an external identifier [XML:75] containing one of the following public identifiers [XML:12]:

- `"-//ATSC//DTD DASE Application Metadata 1.0//EN"`

If an entity of an application uses content type `application/dase`, does not specify one of the preceding public identifiers in an external identifier in the document type declaration, and the entity is processed, then a DASE System shall abort the application.

An application entity which employs this content type shall not specify an internal declaration subset. An internal declaration subset is that part of [XML:28] consisting of the delimiters `'['`, `']'`, and all intervening delimiters and non-terminals.

If an entity of an application uses content type `application/dase`, does specify an internal declaration subset (whether empty or not), and the entity is processed, then a DASE System shall abort the application.

6.1.1.5 Attribute Semantics

This section defines the semantics for the common attributes specified for use with all elements permitted by this content type.

6.1.1.5.1 *id* attribute

This common attribute permits identifying a unique element in a document instance.

6.1.1.5.2 *xmlns* attribute

This common attribute permits the specification of an XML Namespace. In this content type, this attribute has a `#FIXED` value and shall not be specified in a document instance.

6.1.1.5.3 *xml:lang* attribute

This common attribute permits the association of a natural language with an element and its children. This attribute may be used to distinguish among multiple description sets that apply to different natural language settings as well as to indicate the natural language of any textual information contained within the element or its attribute values.

The values taken by this attribute shall adhere to the syntax prescribed by [LANG-TAGS].

6.1.1.6 Element Semantics

This section defines the semantics for all elements specified for use with this content type.

6.1.1.6.1 *application* element

The *application* element is the top-most, document element of a DASE Application metadata document instance. Its content consists of exactly one *identifier* element, followed by exactly one *entityset* element, followed by one or more *descset* elements, followed by zero or more *condset*, *cacheset*, or *paramset* elements. Its attributes consist only of common attributes specified above.

6.1.1.6.2 *cache* element

The *cache* element is used to specify one or more cache directives that apply to a set of target resources. Cache directives are used by a DASE System to enable the efficient loading and processing of a DASE Application.

The attributes of a *cache* element consist of the common attributes specified above and the following attributes:

- `directives`
- `target`

If multiple *cache* elements would specify the same target, then they shall be combined into a single element.

6.1.1.6.2.1 directives attribute

This required attribute specifies one or more cache directives which apply to a non-empty collection of application resources as specified by the *target* attribute. The value of this attribute shall be a comma-separated list of one or more of the directives specified below or a non-standard directive which starts with the prefix `"x-"`.

- `max-age "=" delta-seconds`
- `no-cache`

- `no-store`
- `preload [ "=" priority ]`

Note: See [HTTP], Section 13, for additional information regarding the `max-age`, `no-cache`, and `no-store` directives.

Note: Non-standard values of this attribute are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

6.1.1.6.2.1.1 *max-age* directive

The *max-age* cache directive may be used to specify the maximum time period over which an application resource may be cached. The value of the *max-age* directive, *delta-seconds*, specifies the number of seconds from the time of reception.

If the value of the *max-age* directive is zero, then the resource, if already cached, shall be expunged from the cache.

6.1.1.6.2.1.2 *no-cache* directive

The *no-cache* cache directive may be used to specify that an application resource shall not be cached.

6.1.1.6.2.1.3 *no-store* directive

The *no-store* cache directive may be used to specify that an application resource, if cached, shall not be stored in non-volatile memory.

6.1.1.6.2.1.4 *preload* directive

The *preload* cache directive may be used to request that a resource be preloaded into the cache prior to its initial reference.

The optional value of the *preload* directive, *priority*, shall be a non-negative integer, which indicates the relative priority of cache use in the case that the cache cannot satisfy all caching requirements. Lower values of priority shall take precedence over higher values. If no priority is specified on a *preload* directive, then its priority shall be interpreted as one greater than the highest specified priority on any *preload* directive.

If the *priority* value of the *preload* directive is zero, then the directive shall be interpreted as if no priority were specified.

A DASE System should make a best effort attempt to satisfy preload requests prior to performing application activation.

6.1.1.6.2.2 target attribute

This required attribute specifies a reference to either a specific application resource or a set of application resources. A specific application resource is specified as a URI. A set of application resources may be specified according to one of the following wildcard specifications:

- `[ <base uri> ] '*'`
- `[ <base uri> ] '-'`

If the optional `<base uri>` is specified, then it shall not specify either a fragment identifier or a query identifier.

The first wildcard specification, using `'*'`, indicates all resources whose identifiers have *<base uri>* as their prefix and do not contain any subsequent path separator (`'/'`) characters after this prefix. The second wildcard specification, using `'-'`, indicates all resources whose identifiers have *<base uri>* as their prefix.

If *<base uri>* is not specified, then the base prefix of the URI of the application root resource shall be used as if it had been specified as the *<base uri>*.

Example: If the value of the target attribute is `"*"` and the URI of the application root resource is `"lid://xyz.com/app/meta.xml"`, then the value of the implied *<base uri>* is `"lid://xyz.com/app/"` and the following resource identifiers would match this target:

```
lid://xyz.com/app/meta.xml
lid://xyz.com/app/perm.xml
```

whereas, the following resource identifiers would not match this target:

```
lid://xyz.com/app/page1/top.xml
lid://xyz.com/app/page1/frm1/top.xml
lid://xyz.com/app/page2/top.xml
lid://xyz.com/app/page2/frm2/top.xml
```

Example: If the value of the target attribute is `"-"` and the URI of the application root resource is `"lid://xyz.com/app/meta.xml"`, then the value of the implied *<base uri>* is `"lid://xyz.com/app/"` and the following resource identifiers would match this target:

```
lid://xyz.com/app/meta.xml
lid://xyz.com/app/perm.xml
lid://xyz.com/app/page1/top.xml
lid://xyz.com/app/page1/frm1/top.xml
lid://xyz.com/app/page2/top.xml
lid://xyz.com/app/page2/frm2/top.xml
```

6.1.1.6.3 *cacheset* element

The *cacheset* element serves as a container element for a non-empty set of *cache* elements. Its attributes consist only of common attributes specified above.

Within a *cacheset* element, the order of children *cache* elements shall be significant such that later *cache* elements take priority over earlier *cache* elements; in particular, if a target of a later *cache* element includes an application resource which is included in the target of an earlier *cache* element, then directives any the later *cache* element shall be added to (if a different directive) or shall override (if the same directive) those directives in an earlier *cache* element.

The target of a *cache* element child of one *cacheset* element shall not intersect with the target of a *cache* element child of another *cacheset* element. That is, the union of targets of one *cacheset* element shall be mutually exclusive with respect to the union of targets of another *cacheset* element.

6.1.1.6.4 *cond* element

The *cond* (condition) element specifies a condition which must or may be satisfied by a DASE System prior to activating the application.

> *Note:* See Section 6.1.1.6.4.2 for further information on how a condition is qualified as necessary or recommended.

The content of a *cond* element consists of zero or more *param* elements. The mandatory and permitted usage of child *param* elements is governed by the value of the *capability* attribute.

The attributes of a *cond* element consist of the common attributes specified above and the following attributes:

- `capability`

- `qualifier`

If multiple *cond* elements would specify the same capability and qualifier attribute values, then they shall be combined into a single element. Furthermore, once combined, no two *param* element children of a *cond* element shall specify the same value for a *param* element's *name* attribute. If multiple *param* element children of a single *cond* element do specify the same value for the *name* attribute, then a DASE System shall consider the condition expressed by the *cond* element to not be satisfied.

> *Note*: If an application metadata entity fails to combine multiple *cond* elements as described above, then a DASE System should interpret those element as being logically combined according to the above rules.

6.1.1.6.4.1 capability attribute

This required attribute specifies the specific capability which must or may be satisfied by a DASE System in order to activate the application. The value of this attribute shall be either one of the values specified below or a non-standard value which starts with the prefix `"x-"`.

- `cache`
- `extension`
- `graphics`
- `locale`
- `memory`
- `system`

> *Note:* Non-standard values of this attribute are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

6.1.1.6.4.1.1 *cache* capability

The *cache* capability is used to specify requirements or requests with respect to the caching capabilities of a DASE System.

In the present context, *caching* refers to the storage of one or more bounded application resources in their original (undecoded) form in either volatile or non-volatile memory.

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

- `minSize`

6.1.1.6.4.1.1.1 minSize *parameter*

The *minSize* parameter, if specified, indicates the minimum size (in bytes) of cache storage which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer> [ <unit> ]
<unit>  := 'k' | 'K' | 'm' | 'M'
```

The *unit* `k`/`K` is interpreted as kilobytes; the *unit* `m`/`M` is interpreted as megabytes.

Example: The following specifies a required condition that a DASE System must satisfy in order to initiate the specifying DASE Application; in this case, one megabyte of cache must be provided to the application.

```
<cond capability="cache" qualifier="required">
<param name="minSize" value="1M"/>
</cond>
```

6.1.1.6.4.1.2 *extension* capability

The *extension* capability is used to specify requirements or requests with respect to the use of a DASE Extension.

The *extension* capability requires the following parameter, as specified by child *param* elements of the *cond* element specifying this capability:

- `type`

In addition to the above parameter, other parameters are permitted as determined by the referenced extension type.

6.1.1.6.4.1.2.1 type *parameter*

The *type* parameter, which must be present, indicates the extension type, and shall be either a standard value or a non-standard value which consists of either a reversed domain name or starts with the prefix `"x-"`.

The value of this parameter shall be treated as case-insensitive for the purpose of determining value equality.

> *Note:* No standard values are defined by this specification. It is expected that standard values for extension types will be defined in a future specification intended to be employed with DASE-1 or a future level of the DASE Standard.

6.1.1.6.4.1.3 *graphics* capability

The *graphics* capability is used to specify requirements or requests with respect to the graphics capabilities of a DASE System.

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

- `heightPx`
- `sampleBitsA`
- `sampleBitsB`
- `sampleBitsR`
- `sampleBitsG`
- `widthPx`

6.1.1.6.4.1.3.1 heightPx *parameter*

The *heightPx* parameter, if specified, indicates the height (in pixels) of the graphics plane which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.3.2 sampleBitsA *parameter*

The *sampleBitsA* parameter, if specified, indicates the number of bits of each pixel of the graphics plane devoted to the *alpha* (opacity) sample which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.3.3 sampleBitsB *parameter*

The *sampleBitsB* parameter, if specified, indicates the number of bits of each pixel of the graphics plane devoted to the *blue* sample which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.3.4 sampleBitsG *parameter*

The *sampleBitsG* parameter, if specified, indicates the number of bits of each pixel of the graphics plane devoted to the *green* sample which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.3.5 sampleBitsR *parameter*

The *sampleBitsR* parameter, if specified, indicates the number of bits of each pixel of the graphics plane devoted to the *red* sample which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.3.6 widthPx *parameter*

The *widthPx* parameter, if specified, indicates the width (in pixels) of the graphics plane which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer>
```

6.1.1.6.4.1.4 *locale* capability

The *locale* capability is used to specify requirements or requests with respect to the internationalization capabilities of a DASE System.

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

- `lang`

When the *locale* capability is used to specify requirements for support of a locale by a DASE System, then the application shall support the use of that locale.

> *Note:* The set of required locales specified in an application metadata resource also serves to indicate the minimum set of locales supported by the application. This information may be used by application or data service announcement mechanisms to indicate the language(s) supported by the application.

6.1.1.6.4.1.4.1 lang *parameter*

The *lang* parameter, if specified, indicates one or more language tags which correspond with locales which must or may be supported by the DASE System in order to process the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <tag> [ S* ',' S* <tag> ]
<tag>   := a language tag according to [LANG-TAGS]
S       := any whitespace character
```

Example: The following specifies a required condition that a DASE System must satisfy in order to initiate the specifying DASE Application; in this case, locales which correspond to English (en), Spanish (es), and French (fr) language tags must be supported. This application is able to support users of these three languages, provided the system supports these locales.

```
<cond capability="locale" qualifier="required">
<param name="lang" value="en,es,fr"/>
</cond>
```

6.1.1.6.4.1.5 *memory* capability

The *memory* capability is used to specify requirements or requests with respect to the memory capabilities of a DASE System.

In the present context, *memory* refers to the volatile (run-time) storage of application in system memory during the application's decoding and processing phases.

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

- `minSize`

6.1.1.6.4.1.5.1 minSize *parameter*

The *minSize* parameter, if specified, indicates the minimum size (in bytes) of system volatile storage which must or may be provided to the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <non-negative decimal integer> [ <unit> ]
<unit>  := 'k' | 'K' | 'm' | 'M'
```

The *unit* `k`/`K` is interpreted as kilobytes; the *unit* `m`/`M` is interpreted as megabytes.

> *Note*: Different implementations of a DASE System may be more or less efficient with respect to the volatile storage requirements during the decoding and processing a given DASE Application; in particular, the ability of one implementation to process an application in a given memory size does not imply that a different implementation will be able to process the same application with the same memory size.

Example: The following specifies a required condition that a DASE System must satisfy in order to initiate the specifying DASE Application; in this case, two megabytes of system memory must be provided to the application.

```
<cond capability="cache" qualifier="required">
<param name="minSize" value="2M"/>
</cond>
```

6.1.1.6.4.1.6 *system* capability

The *system* capability is used to specify requirements or requests with respect to the overall capabilities of a DASE System.

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

- `level`

6.1.1.6.4.1.6.1 level *parameter*

The *level* parameter, if specified, indicates the particular level of the DASE Standard which must or may be supported by the DASE System in order to process the DASE Application.

The value of this parameter shall adhere to the following syntax:

```
<value> := <positive integer>
```

This specification (DASE-1), admits only the value `'1'` (one) for the *level* parameter.

Example: The following specifies a required condition that a DASE System must satisfy in order to initiate the specifying DASE Application; in this case, the DASE System must implement DASE Level 1 (DASE-1) in order to process the application.

```
<cond capability="system" qualifier="required">
<param name="level" value="1"/>
</cond>
```

6.1.1.6.4.2 qualifier attribute

This required attribute specifies a qualifier regarding the condition which must or may be satisfied by a DASE System in order to activate the application. The value of this attribute shall be either one of the values specified below or a non-standard value which starts with the prefix `"x-"`.

- `requested`
- `required`

Note: Non-standard values of this attribute are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

6.1.1.6.4.2.1 *requested* qualifier

The *requested* qualifier is used to specify that the condition is in the form of a recommendation which may, but need not be satisfied by a DASE System in order to process the DASE Application.

If a DASE System can satisfy a requested capability, then it should satisfy it. If it cannot satisfy the requested capability, then the DASE Application shall not be aborted due to this condition alone.

6.1.1.6.4.2.2 *required* qualifier

The *required* qualifier is used to specify that the condition is in the form of a requirement which must be satisfied by a DASE System in order to activate the DASE Application.

If a DASE System can satisfy a required capability, then it shall satisfy it. If it cannot satisfy the required capability, then the DASE Application shall not be activated and shall be terminated.

6.1.1.6.5 *condset* element

The *condset* (condition set) element serves as a container element for a non-empty set of *cond* (condition) elements. Its attributes consist only of common attributes specified above.

The use of multiple *condset* elements shall be logically interpreted as equivalent to specifying all of the *cond* element children of these multiple *condset* elements in a single *condset* element.

6.1.1.6.6 *desc* element

The *desc* (description) element is used to specify a human readable description of the DASE Application. The content of this element consists of parsed character data (PCDATA). Its attributes consist only of common attributes specified above.

6.1.1.6.7 *descset* element

The *descset* (description set) element is used to specify various descriptive information about a DASE Application intended for presentation to the end-user. Its content consists of exactly one *name* element followed by exactly one *desc* element. Its attributes consist only of common attributes specified above.

Multiple *descset* elements may be specified in the case that different languages are intended to be supported by the DASE Application; in this case, a *descset* element shall be provided for each supported language. No two *descset* elements shall have the same value for the *xml:lang* attribute.

6.1.1.6.8 *entityset* element

The *entityset* element serves as a container element for a non-empty set of *entity* elements. Its attributes consist only of common attributes specified above.

One and only one *entityset* element shall appear as a child of an *application* element

6.1.1.6.9 *entity* element

The *entity* element is used to specify references to certain essential or important entities required to instantiate and decode a DASE Application.

The attributes of an *entity* element consist of the common attributes specified above and the following attributes:

- `entitytype`
- `uri`

6.1.1.6.9.1 entitytype attribute

This required attribute specifies the type of entity. The value of this attribute shall be either one of the values specified below or a non-standard value which starts with the prefix `"x-"`.

- `initial`
- `permissionRequest`

> *Note:* Non-standard values of this attribute are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

6.1.1.6.9.1.1 *initial* entity type

The *initial* entity type is used to specify the resource which represents the initial entity of a DASE Application. A DASE System shall initiate the decoding of a DASE Application by decoding the initial entity.

A DASE Application shall specify exactly one initial entity in the DASE Application's metadata resource.

6.1.1.6.9.1.2 *permissionRequest* entity type

The *permissionRequest* entity type is used to specify the resource which represents the permission request entity of a DASE Application.

A DASE Application shall specify exactly zero or one permission request entity in the DASE Application's metadata resource.

Note: See *DASE-1 Part 6: Security* for more information on the semantics of this entity.

6.1.1.6.9.2 uri attribute

This required attribute specifies a URI which represents a valid reference to a resource which embodies the entity.

6.1.1.6.10 *identifier* element

The *identifier* element is used to specify a unique identifier for a DASE Application.

A DASE Application shall specify an identifier which is, with high probability, unique in space and time.

An *identifier* element may contain zero or more *param* element children which specify alternative identifiers of a DASE Application.

Note: No alternative identifier for a DASE Application is defined by this specification.

The attributes of an *identifier* element consist of the common attributes specified above and the following attribute:

- `uuid`

6.1.1.6.10.1 uuid attribute

This required attribute specifies a UUID (universal unique identifier) which serves as a unique identifier of the DASE Application.

The value of a UUID shall adhere to the semantic constraints and string representation syntax specified in [UUID].

6.1.1.6.11 *name* element

The *name* element is used to specify a human readable name of the DASE Application. The content of this element consists of parsed character data (PCDATA). Its attributes consist only of common attributes specified above.

6.1.1.6.12 *param* element

The *param* element is used for the following purposes: (1) to provide additional information for conditions according to a condition's capability (Section 6.1.1.6.4), (2) to specify general application parameters (Section 6.1.1.6.13), and (3) to specify alternative identifiers for an application (Section 6.1.1.6.10).

The content of a *param* element is `EMPTY`.

The attributes of a *param* element consist of the common attributes specified above and the following attribute:

- `name`
- `value`

6.1.1.6.12.1 name attribute

This attribute specifies the name of a parameter. All parameter names except those that start with the prefix `"x-"` are reserved for use by the DASE Standard.

Note: Non-standard parameter names are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

6.1.1.6.12.2 value attribute

This attribute specifies the value of a parameter.

Note: For information on permitted values of this attribute, see the individual parameter definitions.

6.1.1.6.13 *paramset* element

The *paramset* element serves as a container element for a non-empty set of *param* elements which specify application parameters. Its attributes consist only of common attributes specified above.

The *param* (parameter) children elements of the *paramset* element shall specify a name which is either one of the values specified below or a non-standard value which starts with the prefix `"x-"`.

- `arg.`*n*, where *n* is a non-negative integer
- `classpath`
- `deferDisplay`
- `legacy`
- `noautoload`

Note: Non-standard parameter names are intended for use by DASE Application and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

6.1.1.6.13.1 arg.n application parameter

This set of parameters is used to specify general arguments to DASE Applications.

The syntax and semantics of the values of this set of parameters are not defined by the DASE Standard, but are application specific.

If specified, the value of *n* shall range from 0 to *N*-1 where *N* is the number of parameters in this set. The order of appearance of these parameters need not be ordered in *n*. No more than one parameter may use the same value *n*.

6.1.1.6.13.2 classpath application parameter

This parameter is employed by active object content. The syntax and semantics of this application parameter are specified in *DASE-1 Part 3: Procedural Applications and Environment*, Section 5.1.1.1.3.

6.1.1.6.13.3 deferDisplay application parameter

The *deferDisplay* parameter is used by a declarative application environment to initialize the value of the `DocumentViewExt::refreshOnChange` property as specified by *DASE-1 Part 2: Declarative Applications and Environment*, Section 5.3.1.2.5.1.3. The value of this parameter, if specified, shall be either `true` or `false`; if no value is specified, a value of `true` shall be used. If this parameter is not specified by an application, then a default value of `false` shall be used.

6.1.1.6.13.4 legacy application parameter

The *legacy* parameter is used by a declarative application environment in order to enable certain compatibility behavior required to process legacy applications. See *DASE-1 Part 2:*

Declarative Applications and Environment, for more information. The value of this parameter, if specified, shall be either `true` or `false`; if no value is specified, a value of `true` shall be used. If this parameter is not specified by an application, then a default value of `false` shall be used.

6.1.1.6.13.5 noautoload application parameter

The *noautoload* parameter is used by a DASE Application to disable application auto-loading in circumstances where transport signaling would cause automatic application load. The value of this parameter, if specified, shall be either `true` or `false`; if no value is specified, a value of `true` shall be used. If this parameter is not specified by an application, then a default value of `false` shall be used.

A DASE Application whose effective *noautoload* parameter is `true` shall not be loaded by a DASE System except by an explicit load request by another application or by the end-user.

6.2 Graphics Content

Graphics content comprises non-streaming content types that represent a single static (still) image frame. Graphics content shall adhere to one of the following content types as specified below:

Table 1 Graphics Content Types

`image/jpeg`	Joint Photographic Expert Group Graphics
`image/png`	Portable Network Graphics

If an entity of a DASE Application takes the form of a graphics content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

6.2.1 image/jpeg

An application entity identified as content type `image/jpeg` shall adhere to the interchange format defined by Annex B of ISO/IEC 10918-1 *Digital Compression and Coding of Continuous-Tone Still Images* [JPEG]. In addition, the following constraints shall apply:

- compression mode shall be sequential DCT-based with Huffman coding of DCT coefficients;
- color space shall be YCbCr normalized to 256 levels; the resulting RGB components shall not be gamma corrected; if one component is used, that component shall be the Y component;
- image orientation shall be top down;
- the position of pixels in sub-sampled components are defined with respect to the highest resolution component;
- an APP0 marker, defined as follows, shall follow immediately after the SOI marker;

The following table defines the APP0 marker.

Table 2 JPEG APP0 Marker Format

Field	Size (bytes)	Value
APP0	1	JPEG APP0 code
length	2	length of structure including this field
identifier	5	`'JFIF'` with null terminator and zero parity
version	2	0x0102
units	1	0 = aspect ratio, 1 = dots per inch, 2 = dots per cm
x density	2	horizontal pixel density
y density	2	vertical pixel density
x thumbnail	1	horizontal thumbnail pixel density

y thumbnail	1	vertical thumbnail pixel density
rgb	3*n*	thumbnail samples, at 24 bits per pixel

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the validity requirements specified by this document and by [JPEG].

If an entity of a DASE Application uses content type `image/jpeg`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

An application entity which employs this content type may use an application data segment whose marker code is not 0xFFE0 as well as a JPEG extension segment whose marker code is in the range 0xFFF0 to 0xFFFD. A DASE System shall ignore any application data segment or JPEG extension segment which it does not support.

Note: See [JPEG], Annex B, *Compressed Data Formats*, for more information on application data segments and JPEG extension segments.

A DASE System which supports this facility shall implement the sequential DCT-based decoding processes with Huffman coding of DCT coefficients as defined by [JPEG], Sections F.2.1 and F.2.2.

Note: A DASE System is expected to transform the color space of the image to the color space of the display prior to or during image presentation.

6.2.2 image/png

An application entity identified as content type `image/png` shall adhere to *Portable Network Graphics, Version 1.2* [PNG].

Note: See [PNG-GUIDE] for additional information on the PNG image format.

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the validity requirements specified by [PNG]; in particular, the entity shall contain a valid `IHDR` chunk, one or more `IDAT` chunks, and an `IEND` chunk.

If an entity of a DASE Application uses content type `image/png`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

6.2.2.1 Critical Chunks

A DASE System which supports this facility shall implement the semantics of all critical chunk types, which are:

Table 3 PNG Critical Chunks

`IDAT`	Image Data
`IEND`	Image Trailer
`IHDR`	Image Header
`PLTE`	Palette

Note: See [PNG], Section 4, for the definition of *critical chunks*.

6.2.2.1.1 Palette Chunk

Images of color type 2 or 6 (true color or true color with alpha) should incorporate a palette `'PLTE'` chunk which specifies a suggested set of colors to which the true color image may be quantized in the case that a DASE System does not support direct true color rendering.

6.2.2.2 Ancillary Chunks

A DASE System which supports this facility shall implement the semantics of the following ancillary chunk types:

Table 4 PNG Required Ancillary Chunks

`tRNS`	Transparency

A DASE System which supports this facility should implement the semantics of the following ancillary chunk types:

Table 5 PNG Recommended Critical Chunks

`cHRM`	Chromaticities
`gAMA`	Image Gamma
`pHYs`	Physical Pixel Dimensions

An application entity which employs this content type may incorporate other ancillary chunks of both public and private types.

A DASE System shall ignore any chunk of an unsupported chunk type.

Note: See [PNG], Section 4, for the definition of *ancillary chunks*.

6.3 Non-Streaming Video Content

Non-streaming video content comprises content types that represent one or more image frames to be progressively presented in a synchronous fashion. Non-streaming video content shall adhere to one of the following content types as specified below:

Table 6 Non-Streaming Video Content Types

`video/mng`	Multiple Network Graphics

If an entity of a DASE Application takes the form of a non-streaming video content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

Note: All non-streaming video content types are intended to be delivered as bounded resources.

6.3.1 video/mng

An application entity identified as content type `video/mng` shall adhere to *Multiple Network Graphics, Version 1.0* [MNG], Profile 11 (MNG-LC).

Note: See [PNG-GUIDE] for additional information on the MNG image format.

Note: Although this content type is identified as video, it is not the intent of the DASE standard that this content type be used to represent video information; rather, use of this content type is intended to satisfy the needs of simple image animation represented by non-streaming application entities.

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the validity requirements specified by [MNG]; in particular, the entity shall contain a valid `MHDR` chunk, one or more frame definitions, and an `MEND` chunk.

If an entity of a DASE Application uses content type `image/mng`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

6.3.1.1 Critical Chunks

A DASE System which supports this facility shall implement the semantics of the following critical chunk types, a subset of those defined by [MNG]:

Table 7 MNG Critical Chunks

BACK	Background
DEFI	Define Image
FRAM	Frame Definition
IDAT	PNG Image Data
IEND	PNG Image Trailer
IHDR	PNG Image Header
MEND	Image Trailer
MHDR	Image Header
PLTE	Palette

Note: See [MNG], Section 4, for the definition of *critical chunks*.

6.3.1.2 Ancillary Chunks

A DASE System which supports this facility shall implement the semantics of the following ancillary chunk types:

Table 8 MNG Required Ancillary Chunks

tRNS	Transparency

A DASE System which supports this facility should implement the semantics of the following ancillary chunk types:

Table 9 MNG Recommended Ancillary Chunks

gAMA	PNG Image Gamma
nEED	Need Resources
pHYg	Physical Pixel Size Global

An application entity which employs this content type may incorporate other ancillary chunks of both public and private types.

A DASE System shall ignore any chunk of an unsupported chunk type.

Note: See [MNG], Section 4, for the definition of *ancillary chunks*.

6.4 Non-Streaming Audio Content

Non-streaming audio content comprises content types that represent one or more audio frames to be progressively presented in a synchronous fashion. Non-streaming audio content shall adhere to one of the following content types as specified below:

Table 10 Non-Streaming Audio Content Types

audio/basic	Basic Audio

If a non-streaming audio content type other than one of the above specified types is encountered and is not supported by a DASE System, then no side effect should result. If an entity of a DASE Application takes the form of a non-streaming audio content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

Note: All non-streaming audio content types are intended to be delivered as bounded resources.

6.4.1 audio/basic

An application entity identified as content type `audio/basic` shall consist of single channel audio encoded using 8-bit ISDN mu-law [PCM] at a sample rate of 8000 Hz.

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding format specified by [PCM].

If an entity of a DASE Application uses content type `audio/basic`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

Presentation of this content type shall cause rendition of the isochronous audio content represented by the application entity.

6.5 Streaming Video Content

Streaming video content comprises content types that represent one or more image frames to be progressively presented in a synchronous fashion and, optionally, one or more accompanying audio channels. Streaming video content shall adhere to one of the following content types as specified below:

Table 11 Streaming Video Content Types

`video/mpeg`	MPEG-2 Video Program
`video/mpv`	MPEG-2 Video Elementary Stream

If an entity of a DASE Application takes the form of a streaming video content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

An entity which takes the form of a streaming video content type is designated as an *implied resource*. The content of this entity is not required to be made manifest (exposed) directly to the DASE System or a DASE Application.

6.5.1 video/mpeg

An associated stream identified as content type `video/mpeg` shall consist of a specific MPEG-2 Program carried within an MPEG-2 Transport Stream which adheres to [A/53], where the identity of the MPEG-2 Program is either implied, in the case that only one MPEG-2 Program is carried, or is explicitly identified, in the case that multiple MPEG-2 Programs are carried. In addition to video information, the MPEG-2 Program may have one or more associated audio components as described in [A/53].

Note: In the case that an MPEG-2 Transport Stream carries multiple MPEG-2 Programs, the identity of the MPEG-2 Program may be indicated by a portion of the reference which indicates the use of a resource of this content type.

An associated stream which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding and system rules specified by [A/53].

If an associated stream of a DASE Application uses content type `video/mpeg`, is not valid, and the associated stream is processed, then a DASE System shall not abort the application.

Presentation of this content type shall cause rendition of the isochronous video content and optional audio content represented by the associated stream.

An entity which employs this content type shall be referenced in a DASE Application only by resource identifiers using the television identifier scheme `"tv:"`. Furthermore, such a resource

identifier shall either (1) not specify a query component, in which case an entire virtual channel (service) is referenced, or (2) specify a query which references an aggregate of video and audio program elements.

6.5.2 video/mpv

An associated stream identified as content type `video/mpv` shall consist of a single video elementary stream which adheres to [A/53]. This video elementary stream shall be one of possibly several components of an MPEG-2 Program embedded within an MPEG-2 Transport Stream which adheres to [A/53].

An associated stream which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding and system rules specified by [A/53].

If an associated stream of a DASE Application uses content type `video/mpv`, is not valid, and the associated stream is processed, then a DASE System shall not abort the application.

Presentation of this content type shall cause rendition of the isochronous video content represented by the associated stream.

An entity which employs this content type shall be referenced in a DASE Application only by resource identifiers using the television identifier scheme `"tv:"`. Furthermore, such a resource identifier shall specify a query component which references a single video elementary stream and no other program element.

6.6 Streaming Audio Content

Streaming audio content comprises content types that represent one or more audio frames to be progressively presented in a synchronous fashion. Streaming audio content shall adhere to one of the following content types as specified below:

Table 12 Streaming Audio Content Types

`audio/ac3`	Dolby AC-3 Audio

If a streaming audio content type other than one of the above specified types is encountered and is not supported by a DASE System, then no side effect should result. If an entity of a DASE Application takes the form of a streaming audio content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

6.6.1 audio/ac3

An associated stream identified as content type `audio/ac3` shall consist of a single audio elementary stream which adheres to [A/52]. This audio elementary stream shall be one of possibly several components of an MPEG-2 Program embedded within an MPEG-2 Transport Stream which adheres to [A/53].

An associated stream which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding format specified by [A/52] and to the encoding and system rules specified by [A/53].

If an associated stream of a DASE Application uses content type `audio/ac3`, is not valid, and the associated stream is processed, then a DASE System shall not abort the application.

Presentation of this content type shall cause rendition of the isochronous audio content represented by the associated stream.

An entity which employs this content type shall be referenced only by resource identifiers using the television identifier scheme `"tv:"`. Furthermore, such a resource identifier shall specify

a query component which references a single audio elementary stream and no other program element.

6.7 Font Content

Font content comprises content types that represent glyph shapes to be used in the rendering of textual content. Font content shall adhere to one of the following content types as specified below:

Table 13 Font Content Types

`application/font-tdpfr`	Portable Font Resource

If a font content type other than one of the above specified types is encountered and is not supported by a DASE System, then no side effect should result. If an entity of a DASE Application takes the form of a font content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

6.7.1 application/font-tdpfr

An application entity identified as content type `application/font-tdpfr` shall consist of a portable font resource which adheres to [PFR]. In addition, the following constraints shall apply:

- the *twoByteCharCode* field of the *physFontRecord()* and *pairKernData()* structures shall have the value 1 (one), indicating that 16-bit character codes are employed;
- the *charCode* field of the *charRecord()* and *bmapCharRecord()* structures and the *charCode1* and *charCode2* fields of the *pairKernData()* structure shall be a single, 16-bit code value defined by or permitted by [UNICODE], and furthermore, this code value shall not be a high-surrogate or a low-surrogate code value, as described by [UNICODE], Section 3.7, *Surrogates*;
- the *character[]* field of the *fontID()* structure shall be encoded using [UTF-8].

> *Note:* The design and specification of [PFR] does not adequately address the distinction between characters and glyphs which is required to accommodate the rendering of complex scripts. For further information, see [UNICODE], Section 2.2, *Unicode Design Principles*.

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding format and constraints specified above and by [PFR].

If an entity of a DASE Application uses content type `application/font-tdpfr`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

Presentation of this content type occurs only indirectly as a side effect of presenting textual content types which reference or otherwise require the use of portable font resources.

6.8 Archive Content

Archive content comprises content types that serve as packages for one or more application resources. Archive content shall adhere to one of the following content types as specified below:

Table 14 Archive Content Types

`application/zip`	ZIP Archive Resource

If an entity of a DASE Application takes the form of an archive content type other than one of the above specified types, and the entity is processed, then a DASE System shall not abort the application.

Application entities represented as archive content are not presented as such; in contrast, application resources embodied by archive content may be presented according to their specific content types.

6.8.1 application/zip

An application entity identified as content type `application/zip` shall consist of a ZIP Archive. A ZIP Archive shall adhere to [DASE-ZIP]. In addition, the following constraints shall apply:

- segmentation is not supported: the value of fields *deSegment*, *dtSegmentTrailer*, and *dtSegmentDirectory* fields shall be zero;
- compression methods 0 and 8 shall be supported by a DASE System, and may be used by a DASE Application;
- compression methods 1 through 6 need not be supported by a DASE System, and should not be used by a DASE Application;
- compression methods 7, 9, and 10 need not be supported by a DASE System, and shall not be used by a DASE Application;
- encrypted entries are not supported: the value of bit 0 of fields *ehFlags* and *deFlags* shall be zero;
- the character encoding of fields *ehPathname*, *dePathname*, and *deComment* shall adhere to [UTF-8];
- the fields *ehPathname* and *dePathname* shall be non-empty;
- the value of fields *ehVersionDecoder* and *deVersionDecoder* shall not be greater than 2.0;
- digital signatures are not supported;

An application entity which employs this content type shall be valid. An entity identified as this content type is valid if it adheres to the encoding format and constraints specified above and by [DASE-ZIP].

If an entity of a DASE Application uses content type `application/zip`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

Individual entries of a ZIP Archive shall specify a `"Content-Type"` MIME Header Extension and should specify a `"Last-Modified"` MIME Header Extension using the mechanisms defined by *DASE-1 Part 5: ZIP Archive Resource Format*, Section B.1, *MIME Header Extension*. The content type header value shall adhere to the syntax prescribed by Section 5.1.2.3 above. If specified, the last modified header value shall adhere to the *`rfc1123-date`* syntax as follows:

```
rfc1123-date : wkday "," SP date SP time SP "GMT"
date         : 2DIGIT SP month SP 4DIGIT
time         : 2DIGIT ":" 2DIGIT ":" 2DIGIT
wkday        : "Mon" | "Tue" | "Wed" | "Thu" | "Fri" | "Sat" | "Sun"
month        : "Jan" | "Feb" | "Mar" | "Apr" | "May" | "Jun" |
               "Jul" | "Aug" | "Sep" | "Oct" | "Nov" | "Dec"
```

Individual entries of a ZIP Archive may be referenced using the archive identifier scheme `"archive:"` in accordance with Section 5.1.2.3.1.1 above.

6.9 Trigger Content

This facility consists of a trigger content type. Trigger content is used to permit the delivery of events by the application delivery system to a DASE Application.

> *Note:* This form of event delivery may be contrasted with those events which are generated internally by an application environment, such as terminating an application, or by an end-user action, such as entering a key on a remote key device.

6.9.1 application/dase-trigger

Trigger content shall adhere to the *Extensible Markup Language, Version 1.0* [XML] and shall be identified as content type `application/dase-trigger`.

Furthermore, trigger content shall adhere to the following document type, labeled here according to its formal public identifier, and defined in ANNEX A Document Type Definitions according to procedures set forth in [XML] and [XMLNAMES]:

- `"-//ATSC//DTD DASE Trigger 1.0//EN"`

6.9.1.1 Well Formedness

An application entity which employs this content type shall be well formed as prescribed by [XML], Section 2.1.

If an entity of this content type is not well formed, then a DASE System shall ignore the trigger. If an entity of an application uses content type `application/dase-trigger`, is not well formed, and the entity is processed, then a DASE System shall not abort the application.

6.9.1.2 Validity

An application entity which employs this content type shall be valid as prescribed by [XML], Section 2.8.

If an entity of this content type is not valid, then a DASE System shall ignore the trigger. If an entity of an application uses content type `application/dase-trigger`, is not valid, and the entity is processed, then a DASE System shall not abort the application.

6.9.1.3 XML Declaration

An application entity which employs this content type shall specify a valid XML declaration [XML:23].

Note: The notation [XML:23] refers to [XML] grammar production 23.

If an entity of an application uses content type `application/dase-trigger`, does not specify a valid XML declaration, and the entity is processed, then a DASE System shall not abort the application.

An application entity which employs this content type shall specify an XML declaration with an encoding declaration [XML:80] according to one of the following character encoding systems, and, furthermore, the entity's representation shall employ the specified encoding system as its actual character encoding system:

- `"UTF-8"`
- `"ISO-8859-1"`

If an entity of an application uses content type `application/dase-trigger`, does not specify one of the preceding encoding declarations in the XML declaration according to the actual employed character encoding system, and the entity is processed, then a DASE System shall not abort the application.

An application entity which employs this content type shall not specify an XML declaration with a standalone document declaration [XML:32] with the value `"yes"`.

If an entity of an application uses content type `application/dase-trigger`, does specify a standalone document declaration with the value `"yes"`, and the entity is processed, then a DASE System shall not abort the application.

If an entity of this content type does not satisfy the above constraints, then a DASE System shall ignore the trigger.

6.9.1.4 Document Type Declaration

An application entity which employs this content type shall specify a valid document type declaration [XML:28].

If an entity of an application uses content type `application/dase-trigger`, does not specify a valid document type declaration, and the entity is processed, then a DASE System shall not abort the application.

An application entity which employs this content type shall specify a document type declaration with an external identifier [XML:75] containing one of the following public identifiers [XML:12]:

- `"-//ATSC//DTD DASE Trigger 1.0//EN"`

If an entity of an application uses content type `application/dase-trigger`, does not specify one of the preceding public identifiers in an external identifier in the document type declaration, and the entity is processed, then a DASE System shall not abort the application.

An application entity which employs this content type shall not specify an internal declaration subset. An internal declaration subset is that part of [XML:28] consisting of the delimiters `'['`, `']'`, and all intervening delimiters and non-terminals.

If an entity of an application uses content type `application/dase-trigger`, does specify an internal declaration subset (whether empty or not), and the entity is processed, then a DASE System shall not abort the application.

If an entity of this content type does not satisfy the above constraints, then a DASE System shall ignore the trigger.

6.9.1.5 Attribute Semantics

This section defines the semantics for the common attributes specified for use with all elements permitted by this content type.

6.9.1.5.1 *id* attribute

This common attribute permits identifying a unique element in a document instance.

6.9.1.5.2 *xmlns* attribute

This common attribute permits the specification of an XML Namespace. In this content type, this attribute has a `#FIXED` value and shall not be specified in a document instance.

6.9.1.5.3 *xml:lang* attribute

This common attribute permits the association of a natural language with an element and its children.

The values taken by this attribute shall adhere to the syntax prescribed by [LANG-TAGS].

6.9.1.6 Element Semantics

This section defines the semantics for all elements specified for use with this content type.

6.9.1.6.1 *event* element

The *event* element encapsulates all information that pertains to an event to be dispatched by a DASE System for subsequent event processing.

The content of an *event* element consists of zero or more *param* elements.

The attributes of an *event* element consist of the common attributes specified above and the following attributes.

6.9.1.6.1.1 bubbles attribute

This attribute has the semantics specified for the `Event::bubbles` property by [DOM2-EVENTS], Section 1.4. Its default value is `true`.

6.9.1.6.1.2 cancelable attribute

This attribute has the semantics specified for the `Event::cancelable` property by [DOM2-EVENTS], Section 1.4. Its default value is `false`.

6.9.1.6.1.3 target attribute

This required attribute is used to specify the target of the event. The value of this attribute takes the form of a URI.

> *Note:* See *DASE-1 Part 2: Declarative Applications and Environment*, Section 4.5, for more information on the use of the *target* attribute with a declarative application. See *DASE-1 Part 3: Procedural Applications and Environment*, Section 4.3, for more information on the use of the *target* attribute with a procedural application.

6.9.1.6.1.4 type attribute

This required attribute specifies the event type. It used to determine the processing semantics of the event as well as to determine which parameter set applies for the event. The parameter set which applies to the event determines the form of the *param* element children of the *event* element. The value of this type shall be one of the following standard values or a non-standard value which starts with the prefix `"x-"`:

- `generic`
- `script`

> *Note:* Non-standard values of this attribute are intended for use by application emitters and DASE System implementers who employ a private agreement. When a non-standard value is used, its user should take care to avoid collision with other non-standard values, such as by making use of an organization unique infix.

If an *event* element's *type* attribute does not satisfy the above constraints, then a DASE System shall ignore the event. If an entity of an application uses content type `application/dase-trigger`, does specify an *event* element with a *type* attribute value other than one of the above, and the event is processed, then a DASE System shall not abort the application.

> *Note:* See *DASE-1 Part 2: Declarative Applications and Environment*, Section 4.5, for more information on the use of the *script* event type. See *DASE-1 Part 3: Procedural Applications and Environment*, Section 4.3, for more information on the use of the *generic* event type.

6.9.1.6.2 *param* element

The *param* element is used to provide additional information for events according to the event's type. Each event type specifies a (possibly empty) parameter set which governs the parameters that may be specified by the *param* element as children of the *event* element. The actual parameters are constructed as a collection of name and value pairs according to the information provided by the *param* element's *name* and *value* attributes.

The content of a *param* element is `EMPTY`.

The attributes of a *param* element consist of the common attributes specified above and the following attributes.

6.9.1.6.2.1 name attribute

This attribute specifies the name of a parameter.

The value of this attribute shall be treated as case-insensitive for the purpose of determining value equality.

> *Note:* For information on permitted values of this attribute, see the definitions of specific event types.

6.9.1.6.2.2 value attribute

This attribute specifies the value of a parameter.

> *Note:* For information on permitted values of this attribute, see the individual parameter definitions under the definitions of specific event types.

6.9.1.6.3 *trigger* element

The *trigger* element is the top-most, document element of a DASE Trigger document instance. Its content consists of one or more *event* elements. Its attributes consist only of common attributes specified above in Section 6.9.1.5.

ANNEX A. DOCUMENT TYPE DEFINITIONS

The entirety of this section and its subsections is normative.

This annex specifies two document type definitions according to [XML] for use with application metadata and trigger content facilities.

A.1 DASE Application Metadata Document Type

This document type supports application metadata content.

A.1.1 DTD Driver

```
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                  DASE Application Metadata 1.0 DTD                    -->
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!--
     This is DASE Application Metadata 1.0, an XML Document Type specified
     for use with ATSC DASE Applications.

     This module shall be idenitifed by the following formal public identifier:

     "-//ATSC//DTD DASE Application Metadata 1.0//EN"
-->

<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                              Parameters                               -->
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % XMLNS "http://www.atsc.org/dase-1#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                                 -->
<!-- ....................................................................... -->
```

A.1.2 Document (Content) Model

This document type does not use an external document (content) model.

A.2 DASE Trigger Document Type

This document type supports trigger content.

A.2.1 DTD Driver

```
<!-- ....................................................................... -->
<!--                          DASE Trigger 1.0 DTD                         -->
<!-- ....................................................................... -->

<!--
     This is DASE Trigger 1.0, an XML Document Type specified for use with
     ATSC DASE Applications.

     This module shall be idenitifed by the following formal public identifier:

     "-//ATSC//DTD DASE Trigger 1.0//EN"
-->

<!-- ....................................................................... -->
<!--                              Parameters                               -->
<!-- ....................................................................... -->

<!ENTITY % XMLNS "http://www.atsc.org/dase-1#trigger" >

<!-- ....................................................................... -->
<!--                    Data Type Entity Declarations                      -->
<!-- ....................................................................... -->

<!-- a boolean value -->
<!ENTITY % Boolean.datatype "(true|false)" >

<!-- media type, as per [MIME] -->
<!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 % xmlns.attrib
     "xmlns         %URI.datatype;          #FIXED '%XMLNS;'"
>

<!ENTITY % lang.attrib
     "xml:lang      %LanguageCode.datatype; #IMPLIED"
>

<!ENTITY % id.attrib
     "id            ID                      #IMPLIED"
>

<!-- ....................................................................... -->
<!--               Qualified Element Name Entity Declarations              -->
```

```
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % trigger.qname    "trigger" >
<!ENTITY % event.qname      "event" >
<!ENTITY % param.qname      "param" >

<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
<!--                        Element Declarations                         -->
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!ENTITY % trigger.content  "( %event.qname; )*" >
<!ELEMENT %trigger.qname;   %trigger.content; >
<!ATTLIST %trigger.qname;
      %xmlns.attrib;
      %lang.attrib;
      %id.attrib;
>

<!ENTITY % event.content    "( %param.qname; )*">
<!ELEMENT %event.qname; %event.content; >
<!ATTLIST %event.qname;
      %xmlns.attrib;
      %lang.attrib;
      %id.attrib;
      type          NMTOKEN                 #REQUIRED
      target        %URI.datatype;          #REQUIRED
      cancelable    %Boolean.datatype;      'false'
      bubbles       %Boolean.datatype;      'true'
>

<!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                             -->
<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->
```

A.2.2 Document (Content) Model

This document type does not use an external document (content) model.

ANNEX B. CONTENT TYPES

The entirety of this section is normative.

The content types specified in Table 15 Content Types may be used by a DASE-1 Application and shall be supported by a DASE-1 System.

The last column of Table 15 designates zero or more extensions (separated by semicolons) to be used with each content type in contexts where an extension is required or useful; e.g., when constructing a resource identifier or an archive pathname. An extension marked as *N/A* denotes that the use of an extension is not applicable to this content type due to constraints on referencing mechanisms. An extension shall not be used to determine the content type of a resource except in an exceptional case where no content type metadata is available.

> *Note*: The extensions recommended below apply when the DASE Standard does not indicate the use of a more specific extension.

Table 15 Content Types

Content Type	Defined At	Extensions
`application/dase`	DASE-1 Part 1, Section 6.1	`.xml`
`application/dase-permission`	DASE-1 Part 6, Section 5.1	`.xml`
`application/dase-trigger`	DASE-1 Part 1, Section 6.9	`.xml`
`application/font-tdpfr`	DASE-1 Part 1, Section 6.7	`.pfr`
`application/jar`	DASE-1 Part 3, Section 5.4	`.jar`
`application/java`	DASE-1 Part 3, Section 5.1	`.class`
`application/javatv-xlet`	DASE-1 Part 3, Section 5.1	`.class`
`application/octet-stream`	DASE-1 Part 3, Section 5.2	`.dat`
`application/xhtml+xml`	DASE-1 Part 2, Section 5.1	`.xht;.xhtml`
`application/zip`	DASE-1 Part 1, Section 6.8	`.zip`
`audio/ac3`	DASE-1 Part 1, Section 6.6	*N/A*
`audio/basic`	DASE-1 Part 1, Section 6.4	`.au`
`image/jpeg`	DASE-1 Part 1, Section 6.2	`.jpg;.jpeg`
`image/png`	DASE-1 Part 1, Section 6.2	`.png`
`text/css`	DASE-1 Part 2, Section 5.2	`.css`
`text/ecmascript`	DASE-1 Part 2, Section 5.3	`.es`
`text/plain`	DASE-1 Part 3, Section 5.3	`.txt`
`video/mng`	DASE-1 Part 1, Section 6.3	`.mng`
`video/mpeg`	DASE-1 Part 1, Section 6.5	*N/A*
`video/mpv`	DASE-1 Part 1, Section 6.5	*N/A*

ANNEX C. MINIMUM COLOR SUPPORT

The entirety of this section is normative.

This annex specifies the minimum set of colors for which display support is required.

Table 16 Minimum Color Set

	Black = "#000000"		Green = "#008000"
	Silver = "#C0C0C0"		Lime = "#00FF00"
	Gray = "#808080"		Olive = "#808000"
	White = "#FFFFFF"		Yellow = "#FFFF00"
	Maroon = "#800000"		Navy = "#000080"
	Red = "#FF0000"		Blue = "#0000FF"
	Purple = "#800080"		Teal = "#008080"
	Fuchsia = "#FF00FF"		Aqua = "#00FFFF"

ANNEX D. EXAMPLES

The entirety of this section and its subsections is informative.

This annex presents examples of the use of the content types defined by this specification.

D.1 Application Metadata Content Example

This example depicts an application metadata entity for a declarative application.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE application PUBLIC "-//ATSC//DTD DASE Application Metadata 1.0//EN">
<application>
  <identifier uuid="7cd89e0a-ad29-4145-8b31-013635801f8d"/>
  <entityset>
    <entity entitytype="initial" uri="app.xhtml"/>
    <entity entitytype="permissionRequest" uri="prm.xml"/>
  </entityset>
  <descset xml:lang="en">
    <name>my declarative application</name>
    <desc>my application is an example dase-1 declarative application.</desc>
  </descset>
  <condset>
    <cond qualifier="require" capability="system">
      <param name="level" value="1"/>
    </cond>
    <cond qualifier="require" capability="graphics">
      <param name="widthpx" value="640"/>
      <param name="heightpx" value="480"/>
      <param name="samplebitsr" value="2"/>
      <param name="samplebitsg" value="3"/>
      <param name="samplebitsb" value="2"/>
      <param name="samplebitsa" value="1"/>
    </cond>
    <cond qualifier="request" capability="cache">
      <param name="minsize" value="2m"/>
    </cond>
    <cond qualifier="request" capability="memory">
      <param name="minsize" value="1m"/>
    </cond>
    <cond qualifier="request" capability="graphics">
      <param name="widthpx" value="960"/>
      <param name="heightpx" value="540"/>
      <param name="samplebitsr" value="4"/>
      <param name="samplebitsg" value="4"/>
      <param name="samplebitsb" value="4"/>
      <param name="samplebitsa" value="4"/>
    </cond>
  </condset>
  <cacheset>
    <cache target="-" directives="no-cache"/>
    <cache target="images/splash.jpg" directives="preload"/>
  </cacheset>
</application>
```

D.2 Trigger Content Example

This example depicts a trigger entity containing two *script* events whose targets are different elements within a markup content entity of a declarative application.

> Note: See *DASE-1 Part 2: Declarative Applications and Environment*, Section 4.5, *Trigger Processing*, for more information on the trigger event of type *script*.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE trigger PUBLIC "-//ATSC//DTD DASE Trigger 1.0//EN">
<trigger>
  <event type="script" target="lid://fbs.com/app.xhtml#t1">
    <param name="code" value="fire(1)"/>
  </event>
  <event type="script" target="lid://fbs.com/app.xhtml#t2">
    <param name="code" value="fire(2)"/>
  </event>
</trigger>
```

CHANGES

This section is informative.

Changes from Candidate Standard to Standard

The following table enumerates the changes between the issuance of the candidate standard edition of this specification and the standard edition.

Table 17 Changes from Candidate Standard

Section	Description
1	Change status to standard.
2	Correct title of [URI-LID] reference.
2	Add informative reference to PNG: The Definitive Guide, by Greg Roelofs, [PNG-GUIDE].
2	Update [UNICODE] to reference Unicode Version 3.2.
3	Add *application delivery file system* definition.
3	Add *DASE Extension* definition.
3	Add *DASE Standard* definition.
3	Add *local file system* definition.
3	Add *persistent file system* definition.
5	Add ecmascript resource identifier scheme.
5	Clarify note describing terminate event.
5	Add four anonymous color gestures to minimum user input capabilities.
6.1	Change FPI to include "Application Metadata" to improve description.
6.1	Define *extension* capability token for the purpose of supporting future extensions.
6.1	Clarify that zero or one permission request entity be specified in root resource.
6.1	Constrain use of multiple parameter elements containing same name.
6.1	Correct description of identifier element content model to permit no child elements.
6.1	Add noautoload application parameter.
6.[23]	Add note referencing [PNG-GUIDE] for additional information on PNG and MNG formats.
A.1	Correct specification of identifier element content model to permit no child elements.
B	Clarify description of application resource extensions.
B	Change application/xdml+xml to application/xhtml+xml.
B	Add text/plain content type.
D	Change declarative document file name extensions in examples from .xml to .xhtml.
D.1	Correct parameter name used with *memory* capability in application metadata example.

ACKNOWLEDGEMENTS

This section is informative.

The primary authors of the DASE Standard were: Glenn Adams (Parts 1-3 and 5-8), and Petr Peterka (Part 4). Final editing was performed by Glenn Adams (T3/S17 chair and editor).

Many dedicated people contributed to this work, most of whom served as regular members of the ATSC T3/S17 Specialist Group. We especially thank the following: Art Allison, Wendy Aylsworth, Jonathan Courtney, Kavitha Devara, Mike Dolan, Craig Finseth, Bill Foote, Alan Goldfine, Pat Griffis, Guy Hadland, Edwin Heredia, Jian Huang, Taylor Kidd, Kwangkee Lee, Philippe Perrot, Petr Peterka, Panagiotis Reveliotis, David Rivas, Eddie Schwalb, Rob Snelick, Jim Van Loo, and Ted Wugofski.

For their effort in overseeing the activities of various current and past adhoc groups of T3/S17, we wish to thank: Mike Dolan (End-to-End), Mel Engel (Profile), Alan Goldfine (Conformance), Edwin Heredia (Security), Taylor Kidd (Security), Petr Peterka (API), Satish Thatte (Glossary), and Ted Wugofski (Presentation Engine).

In addition, we wish to thank the following for their contributions to and support for this work: Anne Antoszkiewicz, Rod Archer, Christopher Atienza, John Barkley, Jeff Brown, Bartley Calder, Mike Capuano, Wayne Carr, Lee Chen, Rich Chernock, Jin-Soo Choi, John Collins, Michel Courson, Joseph Cozad, Luke Crook, David Cutts, Peter Dare, Randall Dark, Mike Derrenberger, Aaron Dinwiddie, Rick Doherty, Brian Dougherty, Vincent Dureau, Mike Ellis, Mike Epstein, Bill Feininger, Gus Fernandez, Gerry Field, Wendy Fong, Don Fowler, Martin Freeman, Jud French, Alexander Gelman, Simon Gibbs, Rob Glidden, Adam Goldberg, Iain Hackett, Scott Hamilton, Jeff Huckins, Kilroy Hughes, Alan Kaplan, Jaeryong Kim, Lakshman Krishnamurthy, Adam Kunzman, Tien-Ying Kuo, Jim Kutzner, O-Hyung Kwon, Dave Johnston, Warner Johnston, Robert Joseph, David Lau-Kee, Guillame Lathoud, Chieh-Li Lee, Sang Gil Lee, Victor Liang, John Litke, Gordon Lyon, Christian Maciocco, Andrew McCaffrey, Tom McMahon, Branislav Meandzija, Milan Milenkovic, Bill Miller, Sam Narasimhan, Jon Piesing, Auri Rahimzadeh, Dave Raggett, Lee Raguz, Dave Reynolds, Mike Richmond, Glenn Ripple, Audrey Ruelas, Wayne Salamon, Laurie Schwartz, Elizabeth Seip, Bhavan Shah, Bill Sheppard, Robert Shideleff, Eran Sitnik, Mark Skall, Dick Streeter, Nikola Stojanovic, Larry Taymor, Satish Thatte, Gomer Thomas, Craig Todd, Andrew Twigger, John Ware, Scott Watson, Steve Weinstein, Carmi Weinsweig, Eric Whitman, Tom Wlodowski, Bumjin Yang, Jae-Soo Yoon, Hui Yu, and Dan Zigmond.

Thanks to Skip Pizzi for his dedicated secretarial work for T3/S17 and to Eddie Schwalb for his service as Secretary Pro Tempore.

Thanks to Petr Peterka and Craig Smithpeters for their service as T3/S17 vice chair.

Special thanks are due to Alan Mink for his work on organizing the DASE Symposiums and managing the development of a reference implementation of the DASE PAE.

Thanks to Mark Richer (ATSC Executive Director) for his patience and guidance throughout a long, but ultimately fruitful effort.

Thanks to Craig Tanner (former ATSC Executive Director) for his encouragement and guidance during the formative period of T3/S17.

Thanks to both Ralph Justus (T3 chair) and Stan Barron (former T3 chair) for their support and guidance of T3/S17.

Finally, the development of DASE would not have been possible were it not for the work of Aninda Dasgupta, the former (and founding) chair of T3/S17.