Doc. A/100-3
09 March 2003

DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1)
PART 3: PROCEDURAL APPLICATIONS AND ENVIRONMENT

ATSC Standard

Note that this documents is past the customary 5-year review point. No update of the document is in process.

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 Tables

Blank Page

DASE-1 Procedural Applications and Environment

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/100-3) 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 specification defines an architecture and a collection of facilities by means of which procedural applications may be delivered in an ATSC data broadcast service to a procedural application environment embodied by a compliant receiver.[1]

A procedural application is an organization of information which primarily uses procedural as opposed to declarative mechanisms to express its information content and behavior. An example of a procedural application is a Java TV™ Xlet composed of Java™ class files and embedded graphics, video, and audio.

A procedural application environment is a software environment which decodes and executes a procedural application. An example of a procedural application environment is a Java™ Virtual Machine and its associated APIs. An alternate name for a procedural application environment is *application execution engine*.

> *Note:* In addition to supporting procedural applications, a procedural application environment supports features of declarative applications as defined by *DASE-1 Part 2: Declarative Applications and Environment*. In particular, a declarative application may make use of one or more *embedded* Xlets which make use of facilities defined by this specification.

[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. The patent holder has, however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

1.3 Application

The architecture and facilities of this specification 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 specification is organized as follows:

- Section 1 Describes purpose, application and organization of this specification
- Section 2 Enumerates normative and informative references
- Section 3 Defines acronyms, terminology, and conventions
- Section 4 Specifies procedural application and environment behavior
- Section 5 Specifies procedural application and environment facilities
- Annex A Specifies Java types
- Annex B Specifies Java constants
- Annex C Specifies Java system properties
- Annex D Specifies Xlet context properties
- Annex E Specifies international resource support
- Changes Cumulative changes to specification

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

This specification makes use of certain notational devices to provide valuable informative and explanatory information in the context of normative and, occasionally, informative sections. These devices take the form of paragraphs labeled as *Example* or *Note*. In each of these cases, the material is to be considered informative in nature.

2. REFERENCES

This section defines the normative and informative references employed by this specification. With the exception of Section 2.1, this section and its subsections are informative; in contrast, Section 2.1 is normative.

2.1 Normative References

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

When a conflict exists between this specification and a referenced document, this specification takes precedence.

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

[A/53]

ATSC Digital Television Standard, A/53, ATSC

[DASE]

DASE-1 Part 1: Introduction, Architecture, and Common Facilities, A/100-1, ATSC

[DASE-API]

DASE-1 Part 4: Application Programming Interface, A/100-4, ATSC

[DASE-ZIP]

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

[DOM2]

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

[DOM2-EVENTS]

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

[DOM2-HTML]

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

[DOM2-STYLE]

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

[DOM2-VIEWS]

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

[HAVI-UI]

The HAVi Specification, Chapter 8, Level 2 User Interface, Version 1.1, May 15, 2001, http://www.havi.org/home.html, HAVi Consortium

[HAVI-UI-API]

HAVi Level 2 User Interface APIs 1.1, May 15, 2001, http://www.havi.org/home.html, HAVi Consortium

[JAR]

JAR File Specification, http://java.sun.com/j2se/1.3/docs/guide/jar/jar.html, Sun Microsystems

[JAVATV]

Java TV API Specification, Version 1.0, http://java.sun.com/products/javatv/, Sun Microsystems

[JAVATV-INTRO]

Java TV™ API Technical Overview, Version 1.0, http://java.sun.com/products/javatv/, Sun Microsystems

[JDK1.1.8]

Java Development Kit, Version 1.1.8, http://java.sun.com/products/jdk/1.1/, Sun Microsystems

[JDK1.2.2]

Java Development Kit, Version 1.2.2, http://java.sun.com/products/jdk/1.2/, Sun Microsystems

[JMF]

Java Media Framework Specification, Version 1.0, http://java.sun.com/products/java-media/jmf/1.0/, Sun Microsystems

[JLS1]

Java Language Specification, First Edition, James Gosling et al., Addison Wesley, 1996, ISBN 0-201-63451-1

[JVM1]

Java Virtual Machine Specification, First Edition, Tim Lindholm and Frank Yellin, Addison Wesley, 1996, ISBN 0-201-63452-X

[JVM1-ERRATA]

Errata for the Java Virtual Machine Specification, Tim Lindholm and Frank Yellin, http://java.sun.com/docs/books/vmspec/errata.html, Sun Microsystems

[JVMX]

Inner Classes Specification, February 4, 1997, http://java.sun.com/products/jdk/1.1/docs/guide/innerclasses/, Sun Microsystems

[LANG-TAGS]

Tags for the Identification of Languages, RFC3066, IETF

[MIME-MEDIA]

Multimedia Internet Mail Extensions (MIME) Part Two: Media Types, RFC2046, IETF

[PJAE]

PersonalJava™ Application Environment Specification, Version 1.2A, http://java.sun.com/products/personaljava/, Sun Microsystems

[UNICODE]

Unicode Character Encoding Standard, Version 3.2, Unicode Consortium

[URI]

Uniform Resource Identifiers: Generic Syntax, RFC2396, IETF

[UTF-8]

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

[X.731]

Information Technology – Open Systems Interconnection – Systems Management: State Management Function, ITU-T Recommendation X.731, ITU

2.2 Informative References

[JNI]

Java Native Interface Specification, Version 1.1, May 16, 1997, http://java.sun.com/products/jdk/1.1/docs/guide/jni/, Sun Microsystems

[JLS1-ERRATA]

Clarifications and Amendments to The Java Language Specification, http://java.sun.com/docs/books/jls/clarify.html, Sun Microsystems

[JVM2]

Java Virtual Machine Specification, Second Edition, Tim Lindholm and Frank Yellin, Addison Wesley, 1999, ISBN 0-201-43294-3

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/.

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/.

HAVi Standards

The HAVi Organization, 2694 Bishop Drive, Suite 275, San Ramon, CA 94583, USA; Phone: +1 925 275 6615; Fax: +1 925 275 6691; http://www.havi.org/.

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/.

Java Standards

Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA; http://java.sun.com/.

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 and abbreviations, and terms as employed by this specification.

All acronyms, abbreviations, and terms defined by [DASE] apply to this specification. Only those acronyms, abbreviations, and terms specific to this document and not common to DASE in its entirety are defined herein.

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, that may or may not be present at the option of the content author or the procedural application environment implementer.

3.2 Acronyms and Abbreviations

bslbf	Bit Serial Leftmost Bit First
DAVIC	Digital Audio Video Council
HAVi	Home Audio Video Interoperability
JDK	Java Development Kit
JMF	Java Media Framework
PTS	Presentation Time Stamp
STC	System Time Clock
uimsbf	Unsigned Integer Most Significant Bit First

3.3 Terms

active object content: a category of content types which includes both `application/java` and `application/javatv-xlet` content types.

embedded Xlet: an Xlet that was loaded as a result of processing a markup content entity referenced by a declarative application; an embedded Xlet is specified by means of an XDML *object* element.

primary Xlet: the first Xlet that was loaded as a result of processing a procedural application's initial entity.

secondary Xlet: any Xlet explicitly registered and started by a primary Xlet, an embedded Xlet or another secondary Xlet.

Xlet: an element of active object content expressed as a Java class which implements the `javax.tv.xlet.Xlet` interface; a collection of Java class files and possibly related resources, one class file of which implements the `javax.tv.xlet.Xlet` interface; a collection of resources packaged as a Java archive which embodies the functionality of an Xlet.

4. BEHAVIOR

This section describes certain normative behavior for DASE Applications and Systems which employ the facilities defined by this specification.

4.1 State and Status Management

A procedural application environment shall support generic state and status management functions in adherence with [X.731] as extended and restricted by this specification. Furthermore, any DASE application which employs a Java TV Xlet shall maintain and report Xlet state and status information as described in the following subsections.

4.1.1 State Attributes

A procedural application environment shall support the following state attributes in accordance with [X.731] Clauses 7.1 and 8.1.1:

- *operational*
- *usage*
- *administrative*

State attributes are scalar valued. The value of a state attribute shall be exactly one of the values permitted for the attribute.

4.1.1.1 *operational* state

The *operational* state shall support the semantics of the following values in accordance with [X.731] Clauses 7.1.1 and 8.1.1.1:

- *enabled*
- *disabled*

Unless specified otherwise, the initial value of the *operational* state shall be *enabled*.

> *Note:* See [DASE-API] interface `org.atsc.management.OperationalState` for information on support of the *operational* state.

4.1.1.1.1 Use with Xlets

An Xlet should report changes in its *operational* state to the procedural application environment. The procedural application environment and other Xlets shall not rely upon an Xlet reporting changes in its *operational* state.

4.1.1.1.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements state management facilities shall maintain and report changes in its *operational* state.

> *Note:* See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports state changes.

A procedural application environment shall provide an interface to determine the overall *operational* state of the procedural application environment.

> *Note:* See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *operational* status is exposed to DASE applications.

4.1.1.2 *usage* state

The *usage* state shall support the semantics of the following values in accordance with [X.731] Clauses 7.1.2 and 8.1.1.2:

- *idle*
- *active*
- *busy*

Unless specified otherwise, the initial value of the *usage* state shall be *idle*.

> *Note:* See [DASE-API] interface `org.atsc.management.UsageState` for information on support of the *usage* state.

4.1.1.2.1 Use with Xlets

An Xlet should report changes in its *usage* state to the procedural application environment. The procedural application environment and other Xlets shall not rely upon an Xlet reporting changes in its *usage* state.

An Xlet whose *operational* state is *enabled* but which is not providing a service should report a *usage* status of *idle*. If an Xlet is *enabled* and is busy providing service or is otherwise not able to provide service, it should report a *usage* status of *busy*. If an Xlet is *enabled*, able to provide a service, but awaiting a client for its service, it should report a *usage* status of *active*.

An *enabled* Xlet whose purpose is not to provide a service to a client should report a constant *usage* state of *busy*.

> *Note:* See [DASE-API], `org.atsc.xlet.XletContextExt.stateChanged`, for information on how an Xlet reports state changes.

4.1.1.2.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements state management facilities shall maintain and report changes in its *usage* state.

> *Note:* See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports state changes.

A procedural application environment shall provide an interface to determine the overall *usage* state of the procedural application environment.

> *Note:* See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *usage* status is exposed to DASE applications.

4.1.1.3 *administrative* state

The *administrative* state shall support the semantics of the following values in accordance with [X.731] Clauses 7.1.3 and 8.1.1.3:

- *locked*
- *unlocked*
- *shutting down*

Unless specified otherwise, the initial value of the *administrative* state shall be *unlocked*.

> *Note:* See [DASE-API] interface `org.atsc.management.AdministrativeState` for information on support of the *administrative* state.

4.1.1.3.1 Use with Xlets

A procedural application environment shall maintain the *administrative* state of each Xlet; an Xlet shall not be permitted to alter its *administrative* state.

> *Note:* The determination of when a procedural application environment can or must transition the *administrative* state of an Xlet is not defined by this specification; furthermore, an Xlet is not required to take any action as a result of a transition in this state.

4.1.1.3.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements state management facilities shall maintain and report changes in its *administrative* state.

> *Note:* See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports state changes.

A procedural application environment shall provide an interface to determine the overall *administrative* state of the procedural application environment.

> *Note:* See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *administrative* status is exposed to DASE applications.

4.1.2 Status Attributes

A procedural application environment shall support the following status attributes in accordance with [X.731] Clauses 7.2 and 8.1.2:

- *alarm*
- *procedural*
- *availability*

Status attributes are set valued. If none of the specified attribute values applies, then a status attribute shall have the special value *none*.

4.1.2.1 *alarm* status

The *alarm* status shall support the semantics of the following values in accordance with [X.731] Clause 8.1.2.1:

- *under repair*
- *critical*
- *major*
- *minor*
- *alarm outstanding*

Unless specified otherwise, the initial value of the *alarm* status shall be *none*.

> *Note:* See [DASE-API] interface `org.atsc.management.AlarmStatus` for information on support of the *alarm* status.

4.1.2.1.1 Use with Xlets

An Xlet should report changes in its *alarm* status to the procedural application environment. The procedural application environment and other Xlets shall not rely upon an Xlet reporting its *alarm* status.

Note: See [DASE-API], `org.atsc.xlet.XletContextExt.statusChanged`, for information on how an Xlet reports status changes.

4.1.2.1.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements status management facilities shall maintain and report changes in its *alarm* status.

Note: See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports status changes.

A procedural application environment shall provide an interface to determine the overall *alarm* status of the procedural application environment.

Note: See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *alarm* status is exposed to DASE applications.

4.1.2.2 *procedural* status

The *procedural* status shall support the semantics of the following values in accordance with [X.731] Clause 8.1.2.2:

- *initializing*
- *initialization required*
- *not initialized*
- *reporting*
- *terminating*

Unless specified otherwise, the initial value of the *procedural* status shall be *none*.

Note: See [DASE-API] interface `org.atsc.management.ProceduralStatus` for information on support of the *procedural* status.

4.1.2.2.1 Use with Xlets

An Xlet should report changes in its *procedural* status to the procedural application environment. The procedural application environment and other Xlets shall not rely upon an Xlet reporting its *procedural* status.

Prior to performing `Xlet.initXlet`, a procedural application environment shall initialize an Xlet's *procedural* status to *not initialized*.

During the performance of `Xlet.initXlet`, an Xlet should report its *procedural* status as *initializing*. Upon successful completion of initialization, an Xlet should report an empty *procedural* status.

During the performance of `Xlet.destroyXlet`, an Xlet should report its *procedural* status as *terminating*.

Note: See [DASE-API], `org.atsc.xlet.XletContextExt.statusChanged`, for information on how an Xlet reports status changes.

4.1.2.2.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements status management facilities shall maintain and report changes in its *procedural* status.

Note: See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports status changes.

A procedural application environment shall provide an interface to determine the overall *procedural* status of the procedural application environment.

> *Note:* See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *procedural* status is exposed to DASE applications.

4.1.2.3 *availability* status

The *availability* status shall support the semantics of the following values in accordance with [X.731] Clause 8.1.2.3:

- *in test*
- *failed*
- *power off*
- *off line*
- *off duty*
- *dependency*
- *degraded*
- *not installed*
- *log full*

Unless specified otherwise, the initial value of the *availability* status shall be *none*.

> *Note:* See [DASE-API] interface `org.atsc.management.AvailabilityStatus` for information on support of the *availability* status.

4.1.2.3.1 Use with Xlets

An Xlet should report changes in its *availability* status to the procedural application environment. The procedural application environment and other Xlets shall not rely upon an Xlet reporting its *availability* status.

An Xlet which cannot provide its intended service(s) due to unavailability of a critical resource should report *dependency* in its *availability* status. An Xlet which can provide limited service due to unavailability of a critical resource should report *degraded* in its *availability* status.

An Xlet which performs a periodic function should report *off duty* in its *availability* status when not performing its intended function.

An Xlet which is undergoing testing should report *in test* in its *availability* status. An Xlet which has incurred a non-recoverable error should report *failed* in its *availability* status.

An Xlet should not report *power off* in its *availability* status.

> *Note:* See [DASE-API], `org.atsc.xlet.XletContextExt.statusChanged`, for information on how an Xlet reports status changes.

4.1.2.3.2 Use with Environment Resources

Every environment resource provided by a procedural application environment which implements status management facilities shall maintain and report changes in its *availability* status.

> *Note:* See [DASE-API], `org.atsc.management.ObjectStates`, for information on how an environment resource maintains and reports status changes.

A procedural application environment shall provide an interface to determine the overall *availability* status of the procedural application environment.

Note: See [DASE-API], `org.atsc.system.Receiver`, for information on how an environment's *availability* status is exposed to DASE applications.

4.2 Xlet Lifecycle Management

An Xlet shall exhibit a lifecycle as prescribed by [JAVATV-INTRO], Chapter 7, Application Lifecycle.

Note: The terminology used by [JAVATV-INTRO] assumes that an individual Xlet corresponds one-to-one with an individual application. This correspondence does not necessarily hold within a DASE Application, which permits the use of multiple Xlets.

The overall lifecycle of a DASE Application is specified in [DASE], Section 5.1.3, *Application Lifecycle*. Transitions in a DASE Application's lifecycle shall affect the lifecycle of individual Xlets loaded by the application in accordance with Table 1 Application Lifecycle State Mapping.

Table 1 Application Lifecycle State Mapping

Application Lifecycle	*Xlet Lifecycle Action*
during *initialized* state	`initXlet()`
enter *active* state	`startXlet()`
enter *suspended* state	`pauseXlet()`
enter *uninitialized* state	`destroyXlet()`

Note: No necessary relationship holds between an Xlet's lifecycle state and its X.731 state attribute values.

If a DASE Application makes use of multiple Xlets, then the *pause* and *destroy* mappings described above shall be applied to each *secondary* Xlet followed by each *embedded* Xlet or the *primary* Xlet, depending upon whether the application is a declarative or procedural application, respectively.

Note: The internal order of applying this mapping in the case of multiple secondary Xlets or multiple embedded Xlets is not defined.

Note: See Section 3.3 for definitions of *primary*, *secondary*, and *embedded* Xlets.

An Xlet may directly change its lifecycle state in certain cases: (1) an Xlet may pause itself by invoking `XletContext.notifyPaused()`; and (2) it may cause its destruction by invoking `XletContext.notifyDestroyed()`.

An Xlet which is paused may request that it be resumed (i.e., transitioned from paused to active states) by invoking `XletContext.resumeRequest()`. A procedural application environment is not required to satisfy a resume request, and may, instead, choose to leave the Xlet in a paused state or transition it to the destroyed state. However, if a procedural application environment can resume an Xlet which requests resumption and no administrative reason exists for keeping it suspended, then it should honor the request.

If a *primary* Xlet is destroyed, then the procedural application environment shall terminate the procedural application which resulted in the loading of this Xlet. Otherwise, Xlet state changes need not have a direct impact on a DASE Application's lifecycle.

4.3 Trigger Processing

A procedural application environment shall process trigger content as defined by [DASE], Section 6.9. Trigger content may declare one or more events of a specific type which require processing by a procedural application environment. A procedural application environment shall process all events with a *type* attribute of the following value:

- `generic`

4.3.1 Event Processing

An event whose type is defined as requiring processing by a procedural application environment is processed by creating an instance of the `org.atsc.trigger.TriggerEvent` class and dispatching this instance to applicable event listeners which implement the `org.atsc.trigger.TriggerListener` interface.

Note: No synchronization semantics hold for triggers as defined herein; that is, the decoding of triggers as well as any presentation side effects are considered to be asynchronous with respect to the application delivery system's clock.

Note: See [DASE-API] for the definition of the `org.atsc.trigger` package.

4.3.1.1 bubbles attribute

This attribute shall be ignored by a procedural application environment.

Note: See [DASE], Section 6.9.1.6.1.1, for more information on this attribute.

4.3.1.2 cancelable *attribute*

This attribute shall be ignored by a procedural application environment.

Note: See [DASE], Section 6.9.1.6.1.2, for more information on this attribute.

4.3.1.3 target attribute

This attribute shall have a value of which takes the form of a non-empty, application-defined string. The event shall be dispatched to (1) all trigger event listeners which specified an identical target string when the listener was registered and (2) all trigger event listeners which were registered without a target string.

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

Note: See [DASE], Section 6.9.1.6.1.3, for more information on this attribute.

Note: In the case of a DASE Application which makes use of multiple Xlets, a trigger event is dispatched to all appropriate trigger listeners of each Xlet. The order of dispatching to multiple Xlets is not defined by this specification.

4.3.2 Generic Event Processing

An event whose type attribute has a value of `"generic"` shall be processed as a *generic* event.

The *generic* event type provides support for arbitrary, pre-defined actions to be triggered in active object content under the control of the application emitter. The parameter set of an event of this type is not restricted in either names or values.

When a *generic* event is processed, the following steps shall occur:

(1) all event parameters consisting of name and value pairs shall be extracted from trigger content and used to instantiate an instance of `java.util.Properties`;
(2) the event's target attribute shall be extracted from trigger content;
(3) an instance of `org.atsc.trigger.TriggerEvent` shall be instantiated using the above information;
(4) the trigger event instance shall be dispatched to all applicable listeners.

4.4 Relative Identifier Resolution

A procedural application employs resource identifiers in order to reference various types of application resources in a variety of contexts. These identifiers take a form as described by [DASE], Section 5.1.2.3.1, *Resource Identifiers*. A reference to a resource may take an absolute or a relative form as described by [DASE], Section 5.1.2.3.2, *Resource References*. When a resource reference takes a relative form, the following interpretive rules shall be applied.

Given a relative identifier, the base identifier to be used to absolutize the relative identifier shall be determined by the following rules:

(1) for a relative identifier used with the constructor `java.net.URL(String)`, use the base identifier of the URI which corresponds to the external form of the current Xlet's locator;
(2) for a relative identifier used with the constructor `java.net.URL(URL,String)`, use the rules described by this constructor's defining specification;
(3) for a relative identifier used with the constructor `javax.media.MediaLocator(String)`, use the base identifier of the URI which corresponds to the external form of the current Xlet's locator;
(4) for a relative identifier used with the method `javax.tv.locator.LocatorFactory.createLocator(String)`, use the base identifier of URI which corresponds to the external form of the current Xlet's locator.

If none of the above rules permits resolution of a relative identifier, then an attempt shall be made to use rule three as specified by [DASE], Section 5.1.2.3.2.1, *Relative Resource Identifiers*.

4.5 Relative Name Resolution

A procedural application may directly or indirectly make use of relative names for referring to certain application resources or system defined resources. The procedures for resolving these relative names are described in the following subsections.

4.5.1 File System Names

Certain constructors of the `java.io.File` class as well as certain methods, e.g., `java.awt.Toolkit.getImage(String)`, permit the use of a relative name in order to designate a pathname with respect to some explicit or implied directory. If no directory context is provided, then a relative file name shall be resolved as follows:

(1) if the application is associated with exactly one application delivery file system, then a relative file name shall be resolved with respect to the mount point of that application delivery file system;
(2) if the application is associated with multiple application delivery file systems, then a relative file name shall be resolved with respect to the mount point of the first mounted application delivery file system;
(3) otherwise, a relative file name shall be resolved with respect to the root directory of the local file system.

4.5.2 Java Class Names

The resolution of Java types requires the implied loading of Java class files. When resolving a reference to an application-defined Java class or interface, the value of the application's *classpath* parameter shall be used to create a search list for resolving the fully qualified class name.

> *Note:* See Section 5.1.1.1.3 for more information on resolving class names using the classpath.

When resolving a reference to a system-defined Java class or interface, the value of the application's *classpath* parameter shall not be used; rather, a separate, implementation-defined classpath shall be used to create a search list for resolving the fully qualified class name.

For the purpose of mapping a fully qualified Java class name to a file name or resource identifier, each `"."` appearing in the class name shall be changed to an appropriate separator character and the extension `".class"` shall be appended as a suffix.

> *Note:* The appropriate separator character will depend upon the file or resource access protocol being employed; e.g., when the class file is being loaded from a file system, the value of `java.lang.System.getProperty("file.separator")` is used, when the class file is being loaded from an archive resource, the file or pathname separator employed by the archive resource format is used, etc.

4.5.3 Java Resource Names

When a relative name is used with one of the following methods, the same rules used to resolve an application-defined Java class name (as described above in Section 4.5.2) shall apply:

- `java.lang.ClassLoader.getResource(String)`
- `java.lang.ClassLoader.getResourceAsStream(String)`

When a relative name is used with one of the following methods, the same rules used to resolve a system-defined Java class name (as described above in Section 4.5.2) shall apply:

- `java.lang.ClassLoader.getSystemResource(String)`
- `java.lang.ClassLoader.getSystemResourceAsStream(String)`

4.6 Local File System

A procedural application environment should provide local file system storage for use by DASE Applications which employ active object content types.

A DASE System may purge local file system storage usage at the discretion of either the end-user or the system itself. A purge of the local file system shall not occur during a period that a DASE Application is in a state other than the *uninitialized* state.

> *Note*: This specification does not define either the minimum local file system size or the minimum persistence of files created within this file system beyond the lifecycle of an application.

If a DASE Application creates a directory or file in a local file system, then it should employ the following convention to reduce potential collisions of directory and file names: an application should create files in a directory whose pathname corresponds to the reversed namespace and path of the base identifier of the application's root resource; for example, if the base identifier is `"lid://xyz.com/app1/"`, then the pathname `"/com/xyz/app1/"` should be used.

If a local file system is employed by a DASE System to retain persistent information other than directories or files explicitly created by a DASE Application, then this persistent information shall not be exposed to a DASE Application by means of local file system functionality. Any attempt by a DASE Application to access such information shall cause a runtime exception to be raised.

5. FACILITIES

This specification defines a number of facilities, each of which defines a category of content types by enumerating a set of one or more specific content types. The following categories are defined:

- active object content
- application defined content
- text content
- java archive content

A procedural application environment shall implement all the facilities specified in this section, and both procedural application and declarative application entities may use these facilities.

> *Note*: See *DASE-1 Part 2: Declarative Applications and Environment*, Section 5.1.1.6.8.1, *Active Content Object Element*, for more information on the use of these facilities by a declarative application.

5.1 Active Object Content

This facility consists of a primary active object content type, `application/java`, and a specialized form of this content type, `application/javatv-xlet`. Every procedural application shall contain one or more application entities which take the form of these active object content types. In particular, the initial entity of a procedural application shall take the form of the specialized active object content type, `application/javatv-xlet`.

If the initial entity of a procedural application is a content type other than `application/javatv-xlet` and this application is invoked, then the procedural application environment shall abort the application.

5.1.1 application/java

This active object content type shall adhere to [JVM1], Chapter 4, Class File Format, as extended and restricted below and shall be identified as content type `application/java`.

An application resource of this content type shall specify a Java class file version number in the range 45.3 through 45.65535.

A Java class file whose version is 45.4 or higher may make use of the following extensions as defined by [JVMX]:

- `InnerClasses` attribute
- `Synthetic` attribute

5.1.1.1 Java Virtual Machine

A procedural application environment shall execute active object content in a manner consistent with [JVM1], as amended by [JVM1-ERRATA] and [JVMX], and as further clarified and constrained below.

The Java Virtual Machine shall support Java class files whose version numbers are in the range 45.3 through 45.65535.

When a class is resolved by the Java Virtual Machine, all superclasses (or superinterfaces) shall be resolved.

The Java Virtual Machine may use either *eager* (early or static) or *lazy* (late) resolution of symbolic references; however, if eager resolution would result in an error condition, then any

exception produced by such condition shall not be raised until the execution of the first Java virtual machine instruction which requires resolution to succeed.

> *Note:* A Java Virtual Machine which performs eager resolution must appear to applications to operate as if it performs lazy resolution. See [JLS1], Section 12.3, *Linking of Classes and Interfaces*, for additional information.

If the target of an `invokeinterface` instruction does not support the referenced interface and the instruction is executed, then a `java.lang.IncompatibleClassChangeError` shall be raised as described by [JLS1] Section 13.1.

The Java Virtual Machine is *not* required to support class finalization as required by [JVM1], Section 2.16.8.

> *Note:* Implementers are advised to carefully review [JLS1-ERRATA], *Java Language Specification Clarifications*, and [JVM2], *Summary of Clarifications and Amendments*, for important information regarding Java Virtual Machine semantics.

A procedural application environment is not required to implement support for Java to native compilation services. If a procedural application environment does implement such services, then it shall do so in a manner that is transparent to the processing of an application entity with the exception of differences in elapsed execution time.

5.1.1.1.1 Byte Code Verification

A procedural application environment shall verify `application/java` content as a Java class file according to [JVM1], Section 4.9, Verification of Class Files, and [JLS1], Sections 12.1.2 and 12.3.1.

If an entity of an Xlet uses content type `application/java`, does not correctly verify according to [JVM1], Section 4.9, then the procedural application environment shall not execute the entity, shall raise an appropriate runtime exception, and, if the exception is not caught, shall destroy the Xlet.

5.1.1.1.2 Java Native Interface (JNI)

The Java Native Interface may be, but need not be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this functionality.

> *Note:* See [JNI] for further information on the Java Native Interface.

5.1.1.1.3 Classpath

A procedural application environment shall load application-defined Java class files, i.e., application resources of active object content types, by performing an ordered search of the entries specified by an Xlet's *effective classpath*, as partially determined by the containing application's *classpath* parameter.

> *Note:* See [DASE], Section 6.1.1.6.13.2, for more information on how an application's *classpath* parameter is specified in a DASE Application.

The value of an application's *classpath* parameter shall take the form of a (possibly empty) list of entries, where each entry specifies either (1) a base URI or (2) the URI of a resource of an archive content type, and where multiple entries are separated by the token `";"`.

> *Note:* The token used to delimit entries in the application specified classpath parameter is typically converted to an implementation dependent *path separator* token during the process of parsing the parameter.

If no *classpath* parameter is specified or if its value is an empty string, then it shall be considered to have been specified with a value taking the form of a single entry consisting of the base URI of the application's root resource.

For a primary Xlet, the *effective classpath* shall be the value of the *classpath* parameter as determined above.

For an embedded Xlet, the non-empty value of the *codebase* attribute of the referencing XDML *object* element shall be prepended to the value of the *classpath* parameter as determined above with an appropriate, intervening path separator token. The resulting value shall be treated as the *effective classpath* of the embedded Xlet.

> *Note:* See *DASE-1 Part 2: Declarative Applications and Environment*, Section 5.1.1.6.8, for more information about the XDML *object* element and its *codebase* attribute.

For a secondary Xlet, the *effective classpath* shall be the same as the *effective classpath* of the Xlet which starts the secondary Xlet.

A procedural application environment shall not use an Xlet's *effective classpath* to search for or load system (non-application) defined Java class files. Furthermore, a procedural application environment shall not load any Java class file from any portion of a local file system which is exposed to a DASE application for read or write access.

> *Note:* See Section 5.4 below for more information on Java archive content types.

Example: Given the fully qualified Java class name `com.acme.Class`, and given the following effective classpath:

```
"lid://acme.com/myxlet.jar;lid://acme.com/myxlet/"
```

then the procedural application environment would search for the class file using the following URIs in the order indicated:

1. `"lid://acme.com/myxlet.jar!/com/acme/Class.class"`
2. `"lid://acme.com/myxlet/com/acme/Class.class"`

5.1.1.2 Java Application Programming Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall implement the following application programming interfaces (APIs) as specified in the following subsections:

- Personal Java Application Environment (PJAE)
- Java Media Framework (JMF)
- Java Television (Java TV)
- Home Audio Video Interoperability User Interface (HAVi UI)
- Digital Audio Video Council (DAVIC)
- W3C Document Object Model (DOM)
- DASE Specific (ATSC)

A procedural application environment is not required to implement any deprecated Java class, interface, method, or field; furthermore, an application entity shall not rely upon the presence of support for such deprecated functionality.

An application entity shall not synchronize on any system class or upon any system static object.

If the specification of a Java class does not define any constructor, then the implementation of the class shall explicitly define either a package or private constructor in order to prevent the generation of a default constructor.

5.1.1.2.1 Personal Java Application Environment (PJAE) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall support the use of Personal Java Application Environment APIs in a manner consistent with [PJAE], [JDK1.1.8], and [JDK1.2.2] as extended and restricted below.

If an application entity of this content type requires resolution of a reference to a PJAE class for which support is not required and which is not present in a procedural application environment, then the procedural application environment shall raise a `java.lang.NoClassDefFoundError`, and, if the exception is not caught by the application, shall destroy the Xlet in whose context this reference occurs.

If an application entity of this content type invokes a PJAE method for which support is not required and which is not present in a procedural application environment, then the procedural application environment shall raise a `java.lang.NoSuchMethodError`, and, if the exception is not caught by the application, shall destroy the Xlet in whose context this reference occurs.

If an application entity of this content type invokes a PJAE method for which support is not required and which is partially implemented in the sense that the method is present in, but its semantics are not supported by a procedural application environment, then the procedural application environment should raise a `java.lang.UnsupportedOperationException`, and, if the exception is not caught by the application, shall destroy the Xlet in whose context this reference occurs.

If an application entity of this content type requires resolution of a reference to a PJAE field for which support is not required and which is not present in a procedural application environment, then the procedural application environment shall raise a `java.lang.NoSuchFieldError`, and, if the exception is not caught by the application, shall destroy the Xlet in whose context this reference occurs.

If a *PJAE optional* class or method designated to be not required by this specification is implemented, then it shall be implemented according to [PJAE].

> *Note:* See [PJAE], Section 3, *Definitions*, for the meaning of *optional* as used in the above paragraph.

A procedural application environment shall use the values of constants defined in ANNEX B in the case that a PJAE defined constant field does not specify a value.

> *Note:* See Annexes A.1 through A.13 for all required PJAE types.

5.1.1.2.1.1 Modified PJAE Required Features

The following *PJAE required features* are modified or restricted by this specification:

- `java.awt` package, [PJAE], Section 5.2
- `java.awt.event` package, [PJAE], Section 5.4
- `java.awt.image` package, [PJAE], Section 5.5
- `java.beans` package, [PJAE], Section 5.7
- `java.io` package, [PJAE], Section 5.8
- `java.lang` package, [PJAE], Section 5.9
- `java.net` package, [PJAE], Section 5.12
- `java.security` package, [PJAE], Section 5.17
- `java.text` package, [PJAE], Section 5.23
- `java.util` package, [PJAE], Section 5.25
- `java.util.zip` package, [PJAE], Section 5.27

5.1.1.2.1.1.1 Modifications to and Constraints on `java.awt` package

The following types in the `java.awt` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these types:

- `Button`
- `Canvas`
- `Checkbox`
- `CheckboxGroup`
- `CheckboxMenuItem`
- `Choice`
- `Dialog`
- `Event`
- `FileDialog`
- `Frame`
- `GridBagLayoutInfo`
- `Label`
- `List`
- `Menu`
- `MenuBar`
- `MenuComponent`
- `MenuContainer`
- `MenuItem`
- `MenuShortcut`
- `Panel`
- `PopupMenu`
- `PrintGraphics`
- `PrintJob`
- `Scrollbar`
- `ScrollPane`
- `SystemColor`
- `TextArea`
- `TextComponent`
- `TextField`
- `Window`

Note: The above excluded functionality is satisfied by HAVi UI functionality. See Section 0.

5.1.1.2.1.1.1.1 Modifications to `java.awt.AWTEvent` class

The following constructor in the `java.awt.AWTEvent` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this constructor:

- `AWTEvent(Event)`

5.1.1.2.1.1.1.2 Modifications to `java.awt.AWTEventMulticaster` class

The following methods in the `java.awt.AWTEventMulticaster` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `save(ObjectOutputStream,String)`
- `saveInternal(ObjectOutputStream,String)`

5.1.1.2.1.1.1.3 Modifications to `java.awt.BorderLayout` class

The following deprecated method in the `java.awt.BorderLayout` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `addLayoutComponent(String, Component)`

5.1.1.2.1.1.1.4 Modifications to `java.awt.CardLayout` class

The following deprecated method in the `java.awt.CardLayout` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `addLayoutComponent(String, Component)`

5.1.1.2.1.1.1.5 Constraints on `java.awt.Color` class

The following constraints shall apply to the constant fields specified by `java.awt.Color`:

- `black == new Color ( 0, 0, 0 )`
- `blue == new Color ( 0, 0, 255 )`
- `cyan == new Color ( 0, 255, 255 )`
- `darkGray == new Color ( 64, 64, 64 )`
- `gray == new Color ( 128, 128, 128 )`
- `green == new Color ( 0, 255, 0 )`
- `lightGray == new Color ( 192, 192, 192 )`
- `magenta == new Color ( 255, 0, 255 )`
- `orange == new Color ( 255, 200, 0 )`
- `pink == new Color ( 255, 175, 175 )`
- `red == new Color ( 255, 0, 0 )`
- `white == new Color ( 255, 255, 255 )`
- `yellow == new Color ( 255, 255, 0 )`

5.1.1.2.1.1.1.6 Modifications to `java.awt.Component` class

The `java.awt.Component` class is not required to implement the `java.awt.MenuContainer` interface; however, the `getFont()` method specified by `java.awt.Component` shall be implemented by a procedural application environment.

The following deprecated methods in the `java.awt.Component` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `action(Event,Object)`
- `bounds()`
- `deliverEvent(Event)`
- `disable()`
- `enable()`
- `enable(boolean)`
- `getPeer()`
- `gotFocus(Event,Object)`
- `handleEvent(Event)`
- `hide()`
- `inside(int,int)`
- `keyDown(Event,int)`
- `keyUp(Event,int)`
- `layout()`

- `locate(int,int)`
- `location()`
- `lostFocus(Event,Object)`
- `minimumSize()`
- `mouseDown(Event,int,int)`
- `mouseDrag(Event,int,int)`
- `mouseEnter(Event,int,int)`
- `mouseExit(Event,int,int)`
- `mouseMove(Event,int,int)`
- `mouseUp(Event,int,int)`
- `move(int,int)`
- `nextFocus()`
- `postEvent(Event)`
- `preferredSize()`
- `reshape(int,int,int,int)`
- `resize(Dimension)`
- `resize(int,int)`
- `show()`
- `show(boolean)`
- `size()`

The following methods in the `java.awt.Component` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `add(PopupMenu)`
- `list()`
- `list(PrintStream)`
- `list(PrintStream,int)`
- `list(PrintWriter)`
- `list(PrintWriter,int)`
- `paramString()`
- `print(Graphics)`
- `printAll(Graphics)`
- `remove(MenuComponent)`

The following method in the `java.awt.Component` class shall be present in a procedural application environment; however, its semantics are not required to be fully implemented by a procedural application environment. Furthermore, an application entity shall not rely on the presence of support for the full semantics of this method:

- `setCursor(Cursor)`

If a procedural application environment cannot fulfill the semantics of this method and this method is invoked, then a `java.lang.UnsupportedOperationException` should be raised, and, if the exception is not caught by the application, the application shall be aborted.

5.1.1.2.1.1.1.7 Modifications to `java.awt.Container` class

The following deprecated methods in the `java.awt.Container` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `add(String,Component)`
- `countComponents()`
- `deliverEvent(Event)`
- `insets()`

- `layout()`
- `locate(int,int)`
- `minimumSize()`
- `preferredSize()`

The following methods in the `java.awt.Container` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `addImpl(Component,Object,int)`
- `list(PrintStream,int)`
- `list(PrintWriter,int)`
- `printComponents(Graphics)`

5.1.1.2.1.1.1.8 Modifications to `java.awt.Dimension` class

The `java.awt.Dimension` class should override the method `java.lang.Object.-hashCode()` in order to conform to semantic interdependencies with `java.lang.Object.-equals(Object)`.

5.1.1.2.1.1.1.9 Modifications to `java.awt.Font` class

The following method in the `java.awt.Font` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getPeer()`

5.1.1.2.1.1.1.10 Modifications to and Constraints on `java.awt.FontMetrics` class

The following deprecated method in the `java.awt.FontMetrics` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getMaxDecent()`

The following method shall employ the default platform character encoding system when interpreting the byte array represented by the first argument:

- `bytesWidth(byte[],int,int)`

  Note: The default platform character encoding system may be determined by evaluating the expression `new InputStreamReader(System.in).getEncoding()`.

5.1.1.2.1.1.1.11 Modifications to and Constraints on `java.awt.Graphics` class

The following deprecated method in the `java.awt.Graphics` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getClipRect()`

The following method in the `java.awt.Graphics` class shall be present in a procedural application environment; however, its semantics are not required to be fully implemented by a procedural application environment. Furthermore, an application entity shall not rely on the presence of support for the full semantics of this method:

- `setXORMode(Color)`

If a procedural application environment cannot fulfill the semantics of this method and this method is invoked, then a `java.lang.UnsupportedOperationException` should be raised, and, if the exception is not caught by the application, the application shall be aborted.

The following method shall employ the default platform character encoding system when interpreting the byte array represented by the first argument:

- `drawBytes(byte[],int,int,int,int)`

5.1.1.2.1.1.12 Modifications to `java.awt.GridBagLayout` class

The following protected fields in the `java.awt.GridBagLayout` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these fields:

- `comptable`
- `defaultConstraints`
- `layoutInfo`
- `MAXGRIDSIZE`
- `MINSIZE`
- `PREFERREDSIZE`

The following protected methods in the `java.awt.GridBagLayout` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `AdjustForGravity(GridBagConstraints,Rectangle)`
- `ArrangeGrid(Container)`
- `GetLayoutInfo(Container,int)`
- `GetMinSize(Container,GridBagLayoutInfo)`

5.1.1.2.1.1.13 Constraints on `java.awt.Image` class

The `java.awt.Image.getProperty(String)` method provides a mechanism for querying a variety of image specific or image category specific properties. The following subsections describe a set of predefined properties, including their values and semantics. An implementation may make use of additional, non-standard properties provided the non-standard property name is prefixed with a reversed domain name or with `"x-"`.

5.1.1.2.1.1.13.1 `comment` *image property*

The `"comment"` property may be used to provide a description of an image, its author, or source. If a value other than `Image.UndefinedProperty` or `null` is returned, it shall be an instance of `java.lang.String`. No other constraints are placed on the value.

5.1.1.2.1.1.13.2 `coprect` *image property*

The `"coprect"` property may be used to indicate what crop rectangle has been applied to the original image.

Note: See Section 5.1.1.2.1.1.3.2 for additional information on this property.

5.1.1.2.1.1.13.3 `filters` *image property*

The `"filters"` property may be used to provide a description of a set of image filters which are applied to an image.

Note: See Section 5.1.1.2.1.1.3.4 for additional information on this property.

5.1.1.2.1.1.1.13.4 `rescale` *image property*

The `"rescale"` property may be used to indicate what scaling transformations have been applied to the original image.

Note: See Section 5.1.1.2.1.1.3.6 for additional information on this property.

5.1.1.2.1.1.1.14 Modifications to `java.awt.Insets` class

The `java.awt.Insets` class should override the method `java.lang.Object.hashCode()` in order to conform to semantic interdependencies with `java.lang.Object.equals(Object)`.

5.1.1.2.1.1.1.15 Modifications to `java.awt.Polygon` class

The following deprecated methods in the `java.awt.Polygon` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getBoundingBox()`
- `inside(int,int)`

5.1.1.2.1.1.1.16 Modifications to `java.awt.Rectangle` class

The following deprecated methods in the `java.awt.Rectangle` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `inside(int,int)`
- `move(int,int)`
- `reshape(int,int,int,int)`
- `resize(int,int)`

5.1.1.2.1.1.1.17 Modifications to and Constraints on `java.awt.Toolkit` class

The following methods in the `java.awt.Toolkit` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `createButton(Button)`
- `createCanvas(Canvas)`
- `createCheckbox(Checkbox)`
- `createCheckboxMenuItem(CheckboxMenuItem)`
- `createChoice(Choice)`
- `createComponent(Component)`
- `createDialog(Dialog)`
- `createFileDialog(FileDialog)`
- `createFrame(Frame)`
- `createLabel(Label)`
- `createList(List)`
- `createMenu(Menu)`
- `createMenuBar(MenuBar)`
- `createMenuItem(MenuItem)`
- `createPanel(Panel)`
- `createPopupMenu(PopupMenu)`

- `createScrollbar(Scrollbar)`
- `createScrollPane(ScrollPane)`
- `createTextArea(TextArea)`
- `createTextField(TextField)`
- `createWindow(Window)`
- `getFontPeer(String,int)`
- `getMenuShortcutKeyMask()`
- `getNativeContainer(Component)`
- `getPrintJob(Frame,String,Properties)`
- `getScreenResolution()`
- `getSystemClipboard()`
- `getSystemEventQueue()`
- `getSystemEventQueueImpl()`
- `loadSystemColors(int[])`

Note: The above excluded functionality is satisfied by HAVi UI functionality or is not required for application interoperability. See Section 5.1.1.2.4.

The value returned by the `java.awt.Toolkit.getScreenSize()` method shall be equivalent to the pixel resolution of the current configuration of the default screen device returned by the `org.havi.ui.HScreen.getDefaultGraphicsDevice()` method.

The first argument to the `java.awt.Toolkit.createImage(byte[])` and `createImage(byte[],int,int)` methods shall be a byte array containing the contents of an application resource which conforms to one of the following content types as constrained by [DASE], Sections 6.2 and 6.3:

- `image/jpeg`
- `image/png`
- `video/mng`

Due to the lack of content type information provided to these two methods, an implementation shall determine the content type by employing implementation dependent heuristics. A procedural application environment may optionally support the heuristic detection and decoding of content types other than those specified above; however, an application shall not rely upon support for the detection or decoding of any other image content types.

Note: If an image format error is detected when parsing the image data associated with an instance of `java.awt.Image` created by one of the `createImage(…)` methods, then the error is reported through an applicable `ImageObserver`.

Note: See [PJAE], Section 5.2, for more information on methods `createImage(String)` and `createImage(URL)`, which are adopted from [JDK1.2.2]. Note that the text in [PJAE] is not clear that these methods are to be implemented by the `java.awt.Toolkit` class.

The `java.awt.Toolkit.getProperty(String)` method provides a mechanism for querying a variety of toolkit properties. No predefined properties are defined by this specification. An implementation may make use of additional, non-standard properties provided the non-standard property name is prefixed with a reversed domain name or with `"x-"`.

A procedural application environment is not required to support any specific toolkit property; furthermore, an application entity shall not rely on the presence of support for any specific toolkit property.

5.1.1.2.1.1.2 Modifications to `java.awt.event` package

5.1.1.2.1.1.2.1 Modifications to `java.awt.event.KeyEvent` class

The following deprecated constructor in the `java.awt.event.KeyEvent` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this constructor:

- `KeyEvent(Component,int,long,int,int)`

The method `java.awt.event.KeyEvent.isAction()` shall return true for the following key codes:

- `VK_F1` through `VK_F12`
- `VK_LEFT`, `VK_RIGHT`, `VK_UP`, `VK_DOWN`, `VK_HOME`, `VK_END`
- `VK_PAGE_UP`, `VK_PAGE_DOWN`
- `VK_PRINTSCREEN`, `VK_PAUSE`
- `VK_INSERT`
- `VK_SCROLL_LOCK`, `VK_CAPS_LOCK`, `VK_NUM_LOCK`

5.1.1.2.1.1.2.2 Modifications to `java.awt.event.WindowEvent` class

The following constructor in the `java.awt.event.WindowEvent` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this constructor:

- `WindowEvent(Window,int)`

The following method in the `java.awt.event.WindowEvent` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getWindow()`

5.1.1.2.1.1.3 Modifications to and Constraints on `java.awt.image` package

5.1.1.2.1.1.3.1 Modifications to `java.awt.image.ColorModel` class

The method `java.awt.image.ColorModel.finalize()` need not be overridden by an implementation of a procedural application environment.

5.1.1.2.1.1.3.2 Constraints on `java.awt.image.CropImageFilter` class

The method `setProperties(Hashtable)` shall add a property with the name `"croprect"` and a value that is an instance of `java.awt.Rectangle` initialized to the crop rectangle.

5.1.1.2.1.1.3.3 Modifications to `java.awt.image.DirectColorModel` class

The method `java.awt.image.DirectColorModel.getRGB()` need not be overridden by an implementation of a procedural application environment if it is implemented generically in a base class in such a manner as to respect the indicated semantics.

5.1.1.2.1.1.3.4 Constraints on `java.awt.image.ImageFilter` class

The following methods of the `java.awt.image.ImageFilter` class and its direct and indirect subclasses are intended to be used only by the image producer of the image being filtered. An application should not invoke any of these methods directly except in the limited case that the application implements a subclass of an existing image filter.

- `getFilterInstance(ImageConsumer)`
- `imageComplete(int)`
- `setColorModel(ColorModel)`
- `setDimensions(int,int)`
- `setHints(int)`
- `setPixels(int,int,int,int,ColorModel,byte[],int,int)`
- `setPixels(int,int,int,int,ColorModel,int[],int,int)`
- `setProperties(Hashtable)`

The method `setProperties(Hashtable)` shall add a property with the name `"filters"` and a value that is an instance of `java.awt.String` which adheres to the following syntax:

`[` *`previous-value`* `", " ]` *`filter`*

where *previous-value* is the previous value of the `"filters"` property, if non-null, and where *filter* is the result of invoking `java.lang.Object.toString()` on the current filter.

5.1.1.2.1.1.3.5 Modifications to `java.awt.image.PixelGrabber` class

The following deprecated method in the `java.awt.image.PixelGrabber` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `status()`

5.1.1.2.1.1.3.6 Constraints on `java.awt.image.ReplicateScaleFilter` class

The method `setProperties(Hashtable)` shall add a property with the name `"rescale"` and a value that is an instance of `java.awt.String` which adheres to the following syntax:

`width` `"x"` *`height`* `[ ", "` *`previous-value`* `]`

where *width* and *height* are the new scaled width and height, and where *previous-value* is the previous value of the `"rescale"` property, if non-null.

5.1.1.2.1.1.4 Modifications to `java.beans` package

The following types in the `java.beans` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these types:

- `BeanDescriptor`
- `BeanInfo`
- `Customizer`
- `EventSetDescriptor`
- `FeatureDescriptor`
- `IndexedPropertyDescriptor`
- `IntrospectionException`
- `Introspector`
- `MethodDescriptor`
- `ParameterDescriptor`
- `PropertyDescriptor`
- `PropertyEditor`
- `PropertyEditorManager`
- `PropertyEditorSupport`
- `SimpleBeanInfo`

Note: The above functionality is excluded on the basis that it is intended solely for design-time support of beans, which is not required by DASE applications.

5.1.1.2.1.1.4.1 Modifications to and Constraints on `java.beans.Beans` class

The following methods in the `java.beans.Beans` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getInstanceOf(Object,Class)`
- `isInstanceOf(Object,Class)`
- `setDesignTime(boolean)`
- `setGuiAvailable(boolean)`

The following method in the `java.beans.Beans` class shall not instantiate a bean which is a subclass of `java.applet.Applet`:

- `instantiate(ClassLoader,String)`

The method `java.beans.Beans.isDesignTime()` shall always return `false`.

The method `java.beans.Beans.isGuiAvailable()` shall always return `true`.

5.1.1.2.1.1.5 Modifications to and Constraints on `java.io` package

The following deprecated classes in the `java.io` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these classes:

- `LineNumberInputStream`
- `StringBufferInputStream`

5.1.1.2.1.1.5.1 Constraints on `java.io.BufferedInputStream` class

When the `java.io.BufferedInputStream` class is constructed using the constructor `BufferedInputStream(InputStream)`, the size of the input buffer shall be implementation dependent.

5.1.1.2.1.1.5.2 Modifications to `java.io.ByteArrayOutputStream` class

The following deprecated method in the `java.io.ByteArrayOutputStream` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `toString(int)`

5.1.1.2.1.1.5.3 Modifications to `java.io.DataInput` interface

The deserialization semantics of the `java.io.DataInput` interface shall adhere to those specified by [JDK1.2.2].

The following method in the `java.io.DataInput` interface is not required in any class implementing this interface in a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `readLine()`

5.1.1.2.1.1.5.4 Modifications to `java.io.DataInputStream` class

The following deprecated method in the `java.io.DataInputStream` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `readLine()`

5.1.1.2.1.1.5.5 Modifications to `java.io.DataOutput` interface

The serialization semantics of the `java.io.DataOutput` interface shall adhere to those specified by [JDK1.2.2].

5.1.1.2.1.1.5.6 Modifications to `java.io.File` class

The method `lastModified()` shall return a value which represents the number of milliseconds from 00:00:00 GMT, January 1, 1970.

5.1.1.2.1.1.5.7 Constraints on `java.io.FileInputStream` class

A procedural application environment shall support the fine-grained access control semantics specified by [JDK1.2.2] for the `java.io.FileInputStream` class.

5.1.1.2.1.1.5.8 Constraints on `java.io.FileOutputStream` class

A procedural application environment shall support the fine-grained access control semantics specified by [JDK1.2.2] for the `java.io.FileOutputStream` class.

5.1.1.2.1.1.5.9 Modifications to and Constraints on `java.io.ObjectInputStream` class

The following method in the `java.io.ObjectInputStream` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `readLine()`

An application entity shall not rely on interoperability of serialized objects between distinct implementations of a procedural application environment; nevertheless, a procedural application environment shall support local serialization and deserialization.

The external, serialized format of an object is not defined by this specification.

> *Note:* Support for implementation independent exchange of serialized objects is not required due to lack of specificity of the external, serialized object format such that two independent implementations would be able to serialize and deserialize in an interoperable manner.
>
> *Note:* Even without an interoperable interchange format for serialized objects, an application entity may nevertheless serialize and deserialize objects within a specific procedural application environment implementation provided it supports the semantics of the above types. For example, an application may serialize an object to a persistent file on a receiver's local file storage system, then deserialize that object at a later time.

A procedural application environment shall support the fine-grained access control semantics specified by [JDK1.2.2] for the `java.io.ObjectInputStream` class.

> *Note:* The `ObjectInputStream(InputStream)` constructor and the `enableResolve-Object(boolean)` method check for permission using the `SerializablePermission` class and its `enableSubclassImplementation` and `enableSubstitution` targets, respectively.

5.1.1.2.1.1.5.10 Constraints on `java.io.ObjectOutputStream` class

A procedural application environment shall support the fine-grained access control semantics specified by [JDK1.2.2] for the `java.io.ObjectOutputStream` class.

> *Note:* The `ObjectOutputStream(OutputStream)` constructor and the `enableReplace-Object(boolean)` method check for permission using the `SerializablePermission` class and its `enableSubclassImplementation` and `enableSubstitution` targets, respectively.

5.1.1.2.1.1.5.11 Clarifications to `java.io.PipedInputStream` class

The following fields of the java.io.PipedInputStream class shall adhere to the semantics specified in [JDK1.2.2]:

- `in`
- `out`
- `PIPE_SIZE`

5.1.1.2.1.1.5.12 Modifications to `java.io.PrintStream` class

The following deprecated constructors in the `java.io.PrintStream` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these constructors:

- `PrintStream(OutputStream)`
- `PrintStream(OutputStream,boolean)`

5.1.1.2.1.1.5.13 Modifications to `java.io.RandomAccessFile` class

The following method in the `java.io.RandomAccessFile` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `readLine()`

5.1.1.2.1.1.5.14 Modifications to `java.io.StreamTokenizer` class

The following deprecated constructor in the `java.io.StreamTokenizer` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this constructor:

- `StreamTokenizer(InputStream)`

5.1.1.2.1.1.6 Modifications to and Constraints on `java.lang` package

The following classes in the `java.lang` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these classes:

- `Compiler`
- `Process`

5.1.1.2.1.1.6.1 Modifications to `java.lang.Character` class

The following deprecated methods in the `java.lang.Character` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `isJavaLetter(char)`
- `isJavaLetterOrDigit(char)`
- `isSpace(char)`

5.1.1.2.1.1.6.2 Clarifications to `java.lang.Class` class

The following methods in the `java.lang.Class` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getClasses()`
- `getDeclaredClasses()`

The methods required by `java.lang.Class` class shall adhere to the semantics specified by [JDK1.2.2].

5.1.1.2.1.1.6.3 Modifications to `java.lang.ClassLoader` class

The following deprecated method in the `java.lang.ClassLoader` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `defineClass(byte[],int,int)`

  Note: The [JDK1.2.2] method `defineClass(String,byte[],int,int,java.-security.ProtectionDomain)` is to be added to this class as described in [PJAE], Section 5.9.

The method `java.lang.ClassLoader.loadClass(String,boolean)` may be concrete, and not abstract. In the case that an implementation of this class does not implement the semantics of this method, then a default implementation shall be provided which, if invoked, shall cause a `java.lang.ClassNotFoundException` to be raised.

The method `java.lang.ClassLoader.findLoadedClass(String)` shall adhere to the semantics specified by [JDK1.2.2].

5.1.1.2.1.1.6.4 Constraints on `java.lang.Double` class

The following constraints shall apply to the constant fields specified by `java.lang.Double`:

- `MIN_VALUE == longBitsToDouble(0x0000000000000001L)`
- `MAX_VALUE == longBitsToDouble(0x7FEFFFFFFFFFFFFFL)`
- `NaN == longBitsToDouble(0x7FF8000000000000L)`
- `NEGATIVE_INFINITY == longBitsToDouble(0xFFF0000000000000L)`
- `POSITIVE_INFINITY == longBitsToDouble(0x7FF0000000000000L)`

5.1.1.2.1.1.6.5 Constraints on `java.lang.Float` class

The following constraints shall apply to the constant fields specified by `java.lang.Float`:

- `MIN_VALUE == intBitsToFloat(0x00000001)`
- `MAX_VALUE == intBitsToFloat(0x7F7FFFFF)`
- `NaN == intBitsToFloat(0x7FC00000)`
- `NEGATIVE_INFINITY == intBitsToFloat(0xFF800000)`
- `POSITIVE_INFINITY == intBitsToFloat(0x7F800000)`

5.1.1.2.1.1.6.6 Constraints on `java.lang.Math` class

Non-strict floating point rules shall apply to the functions supported by `java.lang.Math`.

5.1.1.2.1.1.6.7 Constraints on `java.lang.Object` class

Unless explicitly defined by a subclass of `java.lang.Object`, the form taken by the return value of the method `java.lang.Object.toString()` is implementation dependent.

5.1.1.2.1.1.6.8 Modifications to `java.lang.Runtime` class

The following deprecated methods in the `java.lang.Runtime` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getLocalizedInputStream(InputStream)`
- `getLocalizedOutputStream(OutputStream)`

The following methods in the `java.lang.Runtime` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `runFinalizersOnExit(boolean)`
- `traceInstructions()`
- `traceMethodCalls()`

The following methods in the `java.lang.Runtime` class are not required be implemented by a procedural application environment; if they are implemented, then they shall cause a `SecurityException` to be raised if invoked by an application:

- `exec(String)`
- `exec(String,String[])`
- `exec(String[])`
- `exec(String[],String[])`
- `exit(int)`
- `load(String)`
- `loadLibrary(String)`

5.1.1.2.1.1.6.9 Modifications to `java.lang.SecurityManager` class

The following deprecated methods in the `java.lang.SecurityManager` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `classDepth(String)`
- `classLoaderDepth()`
- `currentClassLoader()`
- `currentLoadedClass()`
- `getInCheck()`
- `inClass(String)`
- `inClassLoader()`

The following deprecated field in the `java.lang.SecurityManager` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this field:

- `inCheck`

5.1.1.2.1.1.6.10 Modifications to `java.lang.String` class

The following deprecated constructors in the `java.lang.String` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these constructors:

- `String(byte[],int)`
- `String(byte[],int,int,int)`

The following constructor shall employ the default platform character encoding system when interpreting the byte array represented by the first argument:

- `String(byte[],int,int)`

The following deprecated method in the `java.lang.String` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getBytes(int,int,byte[],int)`

A procedural application environment shall support the character encodings specified by Annex E.1 for those methods of the `java.lang.String` class which use a parameter that specifies a character encoding name.

5.1.1.2.1.1.6.11 Modifications to `java.lang.System` class

The following deprecated method in the `java.lang.System` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getenv(String)`

The following field in the `java.lang.System` class shall be implemented by a procedural application environment; however, an application entity shall not reference this field:

- `in`

  Note: This field is solely provided for use by the implementation of a procedural application environment or other components of a DASE System implementation.

The following fields in the `java.lang.System` class shall be implemented by a procedural application environment. Any output produced as a side effect of writing to the `PrintStream` values of these fields shall not affect the system display; however, a procedural application environment may provide an implementation defined means for displaying such output.

- `err`
- `out`

The following method in the `java.lang.System` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `runFinalizersOnExit(boolean)`

The following methods in the `java.lang.System` class are not required to be implemented by a procedural application environment; if they are implemented, then they shall cause a `SecurityException` to be raised if invoked by an application:

- `exit(int)`
- `load(String)`
- `loadLibrary(String)`
- `setIn(InputStream)`
- `setErr(PrintStream)`
- `setOut(PrintStream)`

5.1.1.2.1.1.6.12 Modifications to `java.lang.Thread` class

The following methods in the `java.lang.Thread` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `countStackFrames()`
- `destroy()`
- `resume()`
- `stop()`
- `stop(Throwable)`
- `suspend()`

5.1.1.2.1.1.6.13 Modifications to `java.lang.ThreadGroup` class

The following methods in the `java.lang.ThreadGroup` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `allowThreadSuspension(boolean)`
- `list()`
- `resume()`
- `stop()`
- `suspend()`

5.1.1.2.1.1.6.14 Addition of `java.lang.UnsupportedOperationException` class

The `java.lang.UnsupportedOperationException` class defined by [JDK1.2.2] shall be implemented by a procedural application environment.

5.1.1.2.1.1.7 Modifications to `java.net` package

The following types in the `java.net` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these types:

- `ConnectException`
- `ContentHandler`
- `ContentHandlerFactory`
- `DatagramSocketImpl`
- `FileNameMap`
- `HttpURLConnection`
- `NoRouteToHostException`
- `ServerSocket`
- `Socket`
- `SocketImpl`
- `SocketImplFactory`
- `UnknownServiceException`
- `URLConnection`
- `URLStreamHandler`
- `URLStreamHandlerFactory`

Note: The above functionality is excluded on the basis that DASE-1 does not require support for a return channel, and, consequently, no reliable use may be made of connection oriented network facilities. In contrast, DASE-1 does support unidirectional, connection-less networking facilities through the `DatagramSocket` and `Multicast-Socket` types.

5.1.1.2.1.1.7.1 Modifications to `java.net.MulticastSocket` class

The following deprecated methods in the `java.net.MulticastSocket` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getTTL()`
- `send(DatagramPacket,byte)`
- `setTTL(byte)`

5.1.1.2.1.1.7.2 Modifications to `java.net.URL` class

When constructing an instance of the `java.net.URL` class, a *stream protocol handler* object need not be constructed.

> *Note*: A stream protocol handler is an instance of `java.net.URLStreamHandler`, whose support is not required by this specification.

The following methods in the `java.net.URL` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getContent()`
- `openConnection()`
- `openStream()`
- `sameFile(URL)`
- `setURLStreamHandlerFactory(URLStreamHandlerFactory)`

5.1.1.2.1.1.8 Modifications to `java.security` package

5.1.1.2.1.1.8.1 Modifications to `java.security.Key` interface

The following field in the `java.security.Key` interface is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this field:

- `serialVersionUID`

5.1.1.2.1.1.8.2 Modifications to `java.security.Provider` class

The following methods in the `java.security.Provider` class shall adhere to the semantics specified by [JDK1.2.2]:

- `clear()`
- `put(Object,Object)`
- `remove(Object)`

5.1.1.2.1.1.8.3 Modifications to `java.security.PublicKey` interface

The following field in the `java.security.PublicKey` interface is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this field:

- `serialVersionUID`

5.1.1.2.1.1.8.4 Modifications to `java.security.Security` class

The following deprecated method in the `java.security.Security` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `getAlgorithmProperty(String,String)`

5.1.1.2.1.1.9 Modifications to `java.text` package

The following types in the `java.text` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these types:

- `BreakIterator`

- `CharacterIterator`
- `CollationElementIterator`
- `CollationKey`
- `Collator`
- `RuleBasedCollator`
- `StringCharacterIterator`

5.1.1.2.1.1.10 Modifications to `java.util` package

5.1.1.2.1.1.10.1 Modifications to and Constraints on `java.util.Calendar` class

If the argument to the method `java.util.Calendar.after(Object)` or `before(Object)` is not an instance of `java.util.Calendar`, then this method shall return `false`.

If the `java.util.Calendar` class or a subclass overrides the method `java.lang.Object.equals(Object)`, then the method `java.lang.Object.hashCode()` should also be overridden.

5.1.1.2.1.1.10.2 Modifications to `java.util.Date` class

The following deprecated constructors in the `java.util.Date` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these constructors:

- `Date(String)`
- `Date(int,int,int)`
- `Date(int,int,int,int,int)`
- `Date(int,int,int,int,int,int)`

The `Date()` constructor should derive time of day from accurate time information available from the application delivery system.

The following deprecated methods in the `java.util.Date` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getDate()`
- `getDay()`
- `getHours()`
- `getMinutes()`
- `getMonth()`
- `getSeconds()`
- `getTimezoneOffset()`
- `getYear()`
- `parse(String)`
- `setDate(int)`
- `setHours(int)`
- `setMinutes(int)`
- `setMonth(int)`
- `setSeconds(int)`
- `setYear(int)`
- `toGMTString()`
- `toLocaleString()`
- `UTC(int,int,int,int,int,int)`

An implementation of the `java.util.Date` class should override the method `java.lang.Object.clone()`.

5.1.1.2.1.1.10.3 Modifications to `java.util.Locale` class

A procedural application environment shall support all locales specified in Annex E.2. Any `Locale` typed field of the `Locale` class which is not required by Annex E.2 is not required to have a value other than `null`. Except for the predefined `Locale` fields specified by Annex E.2, an application entity which uses a `Locale` typed field of the `Locale` class shall not rely on its value being any value other than `null`.

The method `java.util.Locale.getDefault()` shall return a clone of the default locale instance, which shall reflect the current value of the Java system properties `"user.language"` and `"user.region"`.

The method `java.util.Locale.setDefault(Locale)` shall check for permission to write the Java system property `"user.language"`; if denied, a `java.lang.SecurityException` shall be raised.

> *Note*: In order to obtain information about available locales supported by a procedural application environment, the `org.atsc.util.locales` Xlet context property may be used. See Annex D.3 for further information.

5.1.1.2.1.1.10.4 Modifications to `java.util.Properties` class

The following methods in the `java.util.Properties` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `save(OutputStream,String)`
- `list(PrintStream)`
- `list(PrintWriter)`

If an application entity invokes the method `java.util.Properties.load(InputStream)`, then the content of the referenced input stream shall consist of a sequence of US-ASCII characters encoded as individual octets that adhere to the following *property list resource* syntax:

```
properties:         line*
line:               ( empty | comment | property ) nl
empty:              sp*
comment:            ( "#" | "!" ) pchar*
property:           key sp* value-separator [ sp* value ]
nl:                 "\n" | "\r" | "\r\n"
sp:                 " " | "\t"
key:                kchar+
value-separator:    "=" | ":"
value:              vchar*
char:               { any ascii character except NUL, LF, CR }
kchar:              char - { '\\', '=', ':', ' ', '\t' } | escape
vchar:              char - { '\\' } | escape
escape:             "\\n" | "\\r" | "\\t" | unicode-escape | ascii-escape
unicode-escape:     "\\u" 4*[0-9a-fA-F]
ascii-escape:       "\\" { char - { 'n', 'r', 't', 'u' } }
```

Prior to parsing a line according to the above syntax, any line continuations shall first be appended together to form a whole continued line in accordance with the following syntax. When appending line continuations to form a continued line, the sequence matching the `remove-on-append` syntactic token shall be removed at the boundary between a line and its subsequent continuation.

```
continued-line:     line-start line-continuation
line-start:         char* "\\" nl
line-continuation:  sp* char* ( nl | "\\" nl line-continuation )
remove-on-append:   "\\" nl sp*
```

The content type of a property list resource encoded shall be specified as a plain text content type as described below in Section 5.3.1. When a file or resource name extension is used to identify a property list resource, the extension `".properties"` should be used.

If the content of the input stream provided to the `java.util.Properties.load(Input-Stream)` method does not adhere to the above syntax, then a procedural application environment may cause an `IllegalArgumentException` to be raised.

The semantics of the method of `java.util.Properties.put(Object,Object)`, as inherited from `java.util.Hashtable`, shall be constrained as follows: if the first or the second argument is non-null and the type of the argument is not `java.lang.String`, then an `IllegalArgumentException` may be raised.

5.1.1.2.1.1.10.5 Clarifications to `java.util.PropertyResourceBundle` class

The constructor `PropertyResourceBundle(InputStream)` shall employ `Properties.load(InputStream)` to effect the deserialization of a property resource bundle.

5.1.1.2.1.1.10.6 Modifications to `java.util.ResourceBundle` class

A resource bundle should contain a key whose value is `"LocaleString"` and whose value is a locale identifier adhering to the syntax of the value of `java.util.Locale.toString()`.

> *Note*: The presence of this key permits the user of a resource bundle to determine if the bundle returned by `java.util.ResourceBundle.getBundle(…)` corresponds to the requested bundle.

5.1.1.2.1.1.10.7 Constraints on `java.util.TimeZone` class

The value returned by `java.util.TimeZone.getID()` and the argument provided to `java.util.TimeZone.setID(String)` shall adhere to the *timezone-id* syntax as follows:

```
timezone-id:        tz-custom-id
tz-custom-id:       "GMT" sign hours [ [":"] minutes ]
sign:               "+" | "-"
hours:              digit | digit digit
minutes:            digit digit
digit:              [0-9]
```

The method `java.util.TimeZone.getDefault()` shall return a clone of the default timezone instance, which shall reflect the current value of the Java system property `"user.timezone"`. The method `java.util.TimeZone.setDefault(TimeZone)` shall check for permission to write the Java system property `"user.timezone"`; if denied, a `java.lang.-SecurityException` shall be raised.

5.1.1.2.1.1.11 Modifications to `java.util.zip` package

The following class in the `java.util.zip` package is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this class:

- `GZIPInputStream`

  Note: The GZIP archive content type is not supported by DASE-1.

5.1.1.2.1.1.11.1 Clarifications to `java.util.zip.CRC32` class

The class `java.util.zip.CRC32` shall implement the checksum algorithm specified in [DASE-ZIP], Annex C.

5.1.1.2.1.1.11.2 Constraints on `java.util.zip.Inflater` class

The following constructor in the `java.util.zip.Inflater` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this constructor:

- `Inflater()`

The following methods in the `java.util.zip.Inflater` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `getAdler()`
- `setDictionary(byte[])`
- `setDictionary(byte[],int,int)`

The class `java.util.zip.Inflater` need not implement support for the ZLIB header or checksum; furthermore, an application entity shall not rely on the presence of support for the ZLIB header. In particular, the following constructor invocation should cause a `java.lang.-UnsupportedOperationException` to be raised:

- `new Inflater(false)`

5.1.1.2.1.1.11.3 Constraints on `java.util.zip.InflaterInputStream` class

An implementation of the `java.util.zip.InflaterInputStream` class should override the methods `available()` and `close()`.

The `java.util.zip.Inflater` instance employed by the `java.util.zip.Inflater-InputStream` class need not support the ZLIB header or syntax as specified in Section 5.1.1.2.1.1.11.2 above.

5.1.1.2.1.2 Non-Required PJAE Features

The following *PJAE required features* are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these features:

- `java.applet` package, [PJAE], Section 5.1
- `audio/au` audio format, [PJAE], Section 5.1
- `java.awt.datatransfer` package, [PJAE], Section 5.3
- `image/gif` image format, [PJAE], Section 5.5
- `image/xbm` image format, [PJAE], Section 5.5
- `java.awt.peer` package, [PJAE], Section 5.6
- `http` protocol, [PJAE], Section 5.12
- `java.text.resources` package, [PJAE], Section 5.24
- `com.sun.awt` package, [PJAE], Section 6.2
- `com.sun.lang` package, [PJAE], Section 6.3
- `com.sun.util` package, [PJAE], Section 6.4

5.1.1.2.1.3 Required PJAE Optional Features

The following *PJAE optional features* shall be implemented by a procedural application environment:

- `java.io` user-visible file system group, [PJAE], Section 5.8

5.1.1.2.1.3.1.1 Definition of `java.util.zip.ZipConstants` class

The class `java.util.zip.ZipConstants` is defined as follows:

```
interface ZipConstants {
    /* Header signatures */
    static long LOCSIG = 0x04034b50L;   // "PK\003\004"
    static long EXTSIG = 0x08074b50L;   // "PK\007\008"
    static long CENSIG = 0x02014b50L;   // "PK\001\002"
    static long ENDSIG = 0x06054b50L;   // "PK\005\006"
    /* Header sizes in bytes (including signatures) */
    static final int LOCHDR = 30;       // LOC header size
    static final int EXTHDR = 16;       // EXT header size
    static final int CENHDR = 46;       // CEN header size
    static final int ENDHDR = 22;       // END header size
    /* Local file (LOC) header field offsets */
    static final int LOCVER = 4;        // version needed to extract
    static final int LOCFLG = 6;        // general purpose bit flag
    static final int LOCHOW = 8;        // compression method
    static final int LOCTIM = 10;       // modification time
    static final int LOCCRC = 14;       // uncompressed file crc-32 value
    static final int LOCSIZ = 18;       // compressed size
    static final int LOCLEN = 22;       // uncompressed size
    static final int LOCNAM = 26;       // filename length
    static final int LOCEXT = 28;       // extra field length
    /* Extra local (EXT) header field offsets */
    static final int EXTCRC = 4;        // uncompressed file crc-32 value
    static final int EXTSIZ = 8;        // compressed size
    static final int EXTLEN = 12;       // uncompressed size
    /* Central directory (CEN) header field offsets */
    static final int CENVEM = 4;        // version made by
    static final int CENVER = 6;        // version needed to extract
    static final int CENFLG = 8;        // encrypt, decrypt flags
    static final int CENHOW = 10;       // compression method
    static final int CENTIM = 12;       // modification time
    static final int CENCRC = 16;       // uncompressed file crc-32 value
    static final int CENSIZ = 20;       // compressed size
    static final int CENLEN = 24;       // uncompressed size
    static final int CENNAM = 28;       // filename length
    static final int CENEXT = 30;       // extra field length
    static final int CENCOM = 32;       // comment length
    static final int CENDSK = 34;       // disk number start
    static final int CENATT = 36;       // internal file attributes
    static final int CENATX = 38;       // external file attributes
    static final int CENOFF = 42;       // LOC header offset
    /* End of central directory (END) header field offsets */
    static final int ENDSUB = 8;        // number of entries on this disk
    static final int ENDTOT = 10;       // total number of entries
    static final int ENDSIZ = 12;       // central directory size in bytes
    static final int ENDOFF = 16;       // offset of first CEN header
    static final int ENDCOM = 20;       // zip file comment length
}
```

The fields LOCSIG, EXTSIG, CENSIG, and ENDSIG shall be interpreted as if they have an implicit, but unspecified `final` qualifier; i.e., they shall be interpreted as constants.

5.1.1.2.1.4 Non-Required PJAE Optional Features

The following *PJAE optional features* are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these features:

- `java.math` package, [PJAE], Section 5.11
- `ssl`, `gopher`, `ftp`, `mailto`, `file` protocols, [PJAE], Section 5.12
- `java.rmi` package and subpackages, [PJAE], Section 5.13 through 5.16
- `java.security` code-signing group, [PJAE], Section 5.17
- `java.security.cert` code-signing group, [PJAE], Section 5.19
- `java.security.interfaces` package, [PJAE], Section 5.20
- `java.security.spec` package, [PJAE], Section 5.21
- `java.sql` package, [PJAE], Section 5.22
- `java.util` code-signing group, [PJAE], Section 5.25
- `java.util.jar` package, [PJAE], Section 5.26

The following *PJAE optional* classes in the `java.util.zip` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these classes:

- `Adler32`
- `Deflater`
- `DeflaterOutputStream`
- `GZIPOutputStream`
- `ZipOutputStream`

Note: These classes are not required to be supported even though [PJAE], Section 5.27, Note 2 indicates that they are required when `ZipFile` is supported.

5.1.1.2.1.5 Default Cryptographic Service Provider

A procedural application environment is not required to provide a default cryptographic service provider that implements the `java.security.Provider` abstract class.

5.1.1.2.2 Java Media Framework (JMF) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall support the use of Java Media Framework, Version 1.0, APIs in a manner consistent with [JMF] as extended and restricted below.

A procedural application environment shall use the values of constants defined in ANNEX B in the case that a JMF defined constant field does not specify a value.

Note: See Annexes A.14 and A.15 for all required JMF types.

5.1.1.2.2.1 Clarifications

5.1.1.2.2.1.1 Clarifications to `javax.media.MediaLocator` class

If the external form of an instance of `MediaLocator` takes the form of a URI, then the method `getProtocol()` shall return the *scheme* as defined by [URI]; otherwise, the returned value shall be implementation defined.

If the external form of an instance of `MediaLocator` takes the form of a URI, then the method `getRemainder()` shall return the *scheme-specific-part* as defined by [URI]; otherwise, the returned value shall be implementation defined.

The external form of an instance of `MediaLocator` shall be consistent with the external form of an instance of `javax.tv.media.Locator`, and, in particular, shall satisfy the constraint specified by Section 5.1.1.2.3.1.3.

5.1.1.2.2.1.1.1 Clarifications to `javax.media.protocol.DataSource` class

The method `javax.media.protocol.DataSource.setLocator(MediaLocator)` shall cause an error to be raised in the case that a previous media locator was established.

5.1.1.2.2.2 Constraints

5.1.1.2.2.2.1 Constraints on `javax.media.Clock` interface

The following constraints shall apply to the constant fields specified by `javax.media.Clock`:

- `RESET == new Time ( java.lang.Long.MAX_VALUE )`

5.1.1.2.2.2.2 Constraints on `javax.media.Control` interface

A class which implements `Control` need not implement a user interface component. If a user interface component is not provided for a `Control`, then `getControlComponent()` shall return `null`.

5.1.1.2.2.2.3 Constraints on `javax.media.Controller` interface

The following constraints shall apply to the constant fields specified by `javax.media.-Controller`:

- `LATENCY_UNKNOWN == new Time ( java.lang.Long.MAX_VALUE )`

5.1.1.2.2.2.4 Constraints on `javax.media.Duration` interface

The following constraints shall apply to the constant fields specified by `javax.media.-Duration`:

- `DURATION_UNBOUNDED == new Time ( java.lang.Long.MAX_VALUE )`
- `DURATION_UNKNOWN == new Time ( java.lang.Long.MAX_VALUE - 1 )`

5.1.1.2.2.2.5 Constraints on `javax.media.ControllerListener` interface

A class which implements `ControllerListener` shall implement `java.util.-EventListener`.

5.1.1.2.2.2.6 Constraints on `javax.media.GainChangeListener` interface

A class which implements `GainChangeListener` shall implement `java.util.-EventListener`.

5.1.1.2.2.2.7 Constraints on `javax.media.Manager` class

If the external form of a `MediaLocator` which would be used to construct a `DataSource` takes a form which may map to multiple access methods (protocols) or which, by itself, does not designate an access method, then the implementation of `Manager` shall determine a specific

access method (protocol) to be used in order to construct the class name to be used to instantiate a `DataSource`.

The following constraints shall apply to the constant fields specified by `javax.media.Manager`:

- `UNKNOWN_CONTENT_NAME == ContentDescriptor.CONTENT_UNKNOWN`

The first entry in the `java.util.Vector` instance returned from `javax.media.Manager.getDataSourceList(String)` shall adhere to the following syntax:

`media.protocol.`*`protocol-name`*`.DataSource`

where the value of *protocol-name* corresponds to the argument to `getDataSourceList(String)`.

The second and subsequent entries in the `java.util.Vector` instance returned from `javax.media.Manager.getHandlerClassList(String)` shall adhere to the following syntax:

`content-prefix``.media.content.`*`content-name`*`.Handler`

where the value of *content-prefix* corresponds to an entry of the value returned by `javax.media.PackageManager.getContentPrefixList()` and the value of *content-name* corresponds to the result of applying `javax.media.protocol.ContentDescriptor.mimeTypeToPackageName(String)` to the argument to `getHandlerClassList(String)`.

5.1.1.2.2.2.8 Constraints on `javax.media.MediaEvent` interface

A class which implements `MediaEvent` shall be a subtype of the `java.util.EventObject` class.

5.1.1.2.2.2.9 Constraints on `javax.media.MediaHandler` interface

When the method `MediaHandler.setSource(DataSource)` is invoked on an instance of a built-in `Player`, an `IncompatibleSourceException` shall be raised.

5.1.1.2.2.2.10 Constraints on `javax.media.PackageManager` class

A runtime exception shall be raised upon any invocation of either of the following methods of the `PackageManager` class by an application entity:

- `commitContentPrefixList()`
- `commitProtocolPrefixList()`

5.1.1.2.2.2.11 Constraints on `javax.media.Player` interface

A class which implements `Player` need not implement a visual component. If a visual component is not provided for a `Player`, then `getVisualComponent()` shall return `null`.

If a `Player` does provide a visual component, then the component shall support the semantics of `java.awt.Component` with respect to positioning and scaling.

> *Note*: A `Player` that supports only background video presentation, as opposed to component based video presentation, would not provide a visual component.

5.1.1.2.2.2.12 Constraints on `javax.media.protocol.ContentDescriptor` class

The value returned by the method `getContentType()` and the argument supplied to the constructor `ContentDescriptor(String)` and the method `mimeTypeToPackageName(String)` shall adhere to the *content-type* syntax specified by [DASE], Section 5.1.2.3. If one or more optional content type parameters are present in the argument to `mimeTypeToPackage-`

`Name(String)`, then they shall be ignored for the purpose of computing a corresponding package name.

The value of `ContentDescriptor.CONTENT_UNKNOWN` shall be `"application/octet-stream"`.

5.1.1.2.2.2.13 Constraints on `javax.media.protocol.DataSource` class

The value returned by the method `getContentType()` shall adhere to the *content-type* syntax specified by [DASE], Section 5.1.2.3.

5.1.1.2.2.3 Modifications

5.1.1.2.2.3.1 Modification to `javax.media.ControllerEvent` class

The class `javax.media.ControllerEvent` shall be a subtype of the `java.util.EventObject` class.

5.1.1.2.2.3.2 Modification to `javax.media.GainChangeEvent` class

The class `javax.media.GainChangeEvent` shall be a subtype of the `java.util.EventObject` class.

5.1.1.2.2.3.3 Modifications to `javax.media.protocol` package

The following class in the `java.media.protocol` package is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this class:

- `URLDataSource`

5.1.1.2.2.3.3.1 Modifications to `javax.media.protocol.DataSource` class

The following method in the `javax.media.protocol.DataSource` class is not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for this method:

- `initCheck()`

5.1.1.2.2.3.4 Modifications to `javax.media.protocol.ContentDescriptor` class

The method `ContentDescriptor.mimeTypeToPackageName(String)` shall be defined and accessible as a public static method.

5.1.1.2.2.4 Extensions

5.1.1.2.2.4.1 Built-In Data Sources

A procedural application environment shall implement the following built-in data sources:

- asynchronous data piping data source
- asynchronous data piping, raw packet data source
- asynchronous download, non-flow controlled data source
- asynchronous digital television closed captioning data source

5.1.1.2.2.4.1.1 Asynchronous Data Piping Data Source

A procedural application environment shall implement a built-in data source as follows:

Table 2 Asynchronous Data Piping Data Source Parameters

Parameter	Value	Notes
protocol	`"atsc.async.piping"`	
controls	none required	
seekable	no	
rate configurable	no	
type	push only	
source stream	single	1
stream content type	not required	2
stream content length	not required	3
stream controls	none required	
stream positionable	no	
stream format	decapsulated data	4
minimum transfer size	≥ 1	5

Notes

1. Stream shall be instance of `javax.tv.media.protocol.PushSourceStream2`.
2. If content type of stream is not sniffed or not determined through out-of-band metadata, then `SourceStream.getContentDescriptor()` shall return a value equivalent to `new ContentDescriptor(ContentDescriptor.CONTENT_UNKNOWN)`.
3. If content length cannot be determined by out-of-band metadata, then `SourceStream.getContentLength()` shall return `SourceStream.LENGTH_UNKNOWN`.
4. Decapsulation of data piping shall extract and return payload from transport packets. If a packet is duplicated, then its data shall not be returned. If the continuity counter is discontinuous or an error indicator is present, then `javax.tv.media.protocol.DataLostException` shall be raised upon the next invocation of `javax.tv.media.protocol.PushSourceStream2.readStream` which would cause the lost data to be read.
5. Minimum transfer size may be as few as one byte per read operation. Read operations shall return the remainder of buffered data upon each read up to the requested transfer size. If data arrives more quickly than read operations occur, buffer overflow may occur in which case a `DataLostException` shall be raised upon the next read operation.

5.1.1.2.2.4.1.2 Asynchronous Data Piping, Raw Packet Data Source

A procedural application environment shall implement a built-in data source as follows:

Table 3 Asynchronous Data Piping, Raw Packet Data Source Parameters

Parameter	Value	Notes
protocol	`"atsc.async.piping.raw"`	
controls	none required	
seekable	no	
rate configurable	no	
type	push only	
source stream	single	1
stream content type	not required	2
stream content length	not required	3
stream controls	none required	
stream positionable	no	

stream format	raw transport stream packets	4
minimum transfer size	188	5

Notes

1. Stream shall be instance of `javax.tv.media.protocol.PushSourceStream2`.
2. If content type of stream is not sniffed or not determined through out-of-band metadata, then `SourceStream.getContentDescriptor()` shall return a value equivalent to `new ContentDescriptor(ContentDescriptor.CONTENT_UNKNOWN)`.
3. If content length cannot be determined by out-of-band metadata, then `SourceStream.getContentLength()` shall return `SourceStream.LENGTH_UNKNOWN`.
4. Each transport stream packet associated with the stream shall be returned without modification and in its entirety. Each read from stream shall return exactly zero or one packet.
5. If fewer than 188 bytes are requested in a read, then zero bytes shall be returned irrespective of whether one or more packets are buffered for reading. If greater than this number of bytes are requested and this number is a non-zero integral multiple of 188, then either zero bytes or some integral multiple of 188 bytes shall be returned depending upon how many packets have been buffered for reading.

5.1.1.2.2.4.1.3 Asynchronous Download, Non-Flow Controlled Data Source

A procedural application environment shall implement a built-in data source as follows:

Table 4 Asynchronous Download, Non-Flow Controlled Data Source Parameters

Parameter	Value	Notes
protocol	`"atsc.async.download"`	
controls	none required	
seekable	no	
rate configurable	no	
type	push only	
source stream	single	1
stream content type	not required	2
stream content length	not required	3
stream controls	none required	
stream positionable	no	
stream format	decapsulated data	4
minimum transfer size	≥ 1	5

Notes

1. Stream shall be instance of `javax.tv.media.protocol.PushSourceStream2`.
2. If content type of stream is not sniffed or not determined through out-of-band metadata, then `SourceStream.getContentDescriptor()` shall return a value equivalent to `new ContentDescriptor(ContentDescriptor.CONTENT_UNKNOWN)`.
3. If content length cannot be determined by out-of-band metadata, then `SourceStream.getContentLength()` shall return `SourceStream.LENGTH_UNKNOWN`.
4. Decapsulation of non-flow controlled U-N download shall extract and return payload from DSMCC sections. If data is discontinuous or an error indicator is present, then `javax.tv.media.protocol.DataLostException` shall be raised upon the next invocation of `javax.tv.media.protocol.PushSourceStream2.readStream` which would cause the lost data to be read.
5. Minimum transfer size may be as few as one byte per read operation. Read operations shall return the remainder of buffered data upon each read up to the requested transfer size. If data arrives more quickly than read operations occur, buffer overflow may occur in which case a `DataLostException` shall be raised upon the next read operation.

Note: The use of an asynchronous download, non-flow controlled data source is explicitly restricted for use with asynchronous streaming data and not for use for non-streaming data (e.g., carousel data).

5.1.1.2.2.4.1.4 Asynchronous Digital Television Closed Captioning Data Source

A procedural application environment shall implement a built-in data source as follows:

Table 5 Asynchronous Digital Television Closed Captioning Data Source Parameters

Parameter	Value	Notes
protocol	`"atsc.async.dtvcc"`	
controls	none required	
seekable	no	
rate configurable	no	
type	push only	
source stream	single	1
stream content type	`application/octet-stream`	
stream content length	`LENGTH_UNKNOWN`	2
stream controls	none required	
stream positionable	no	
stream format	decapsulated data	3
minimum transfer size	104	3

Notes

1. Stream shall be instance of `javax.tv.media.protocol.PushSourceStream2`.
2. `SourceStream.getContentLength()` shall return `SourceStream.LENGTH_UNKNOWN`.
3. See Section 5.1.1.2.2.4.1.4.1, *DTVCC Stream Format*.

Note: Access to closed captioning data provided by this mechanism is expressly not intended to satisfy any regulatory requirements regarding the processing of closed captioning data on a digital television receiver or any other terminal device.

5.1.1.2.2.4.1.4.1 DTVCC Stream Format

Each transfer operation performed upon an `atsc.async.dtvcc` data source shall take the form of a data frame which adheres to Table 6 DTVCC Frame Format.

Table 6 DTVCC Frame Format

Syntax	*Number of bits*	*Format*
`dtvcc_frame() {`		
`  stc_valid`	1	bslbf
`  pts_valid`	1	bslbf
`  stc_high`	1	uimsbf
`  pts_high`	1	uimsbf
`  reserved`	4	bslbf
`  stc_low`	32	uimsbf
`  pts_low_`	32	uimsbf
`  reserved`	1	bslbf
`  process_cc_data_flag`	1	bslbf
`  reserved`	1	bslbf
`  cc_count`	5	uimsbf
`  reserved`	8	bslbf
`  for ( i=0; i<cc_count; i++ ) {`		
`    marker_bits`	5	'1111 1'
`    cc_valid`	1	bslbf
`    cc_type`	2	bslbf
`    cc_data_1`	8	bslbf

```
        cc_data_2                                                   8           bslbf
    }
}
```

Note: A `dtvcc_frame()` consists of 11 to 104 octets of data, depending upon the value of `cc_count`.

A `dtvcc_frame()` shall be constructed from picture user data contained in the active video elementary stream of the data source specified by the media locator which was used to instantiate this data source. If no video elementary stream is present in the active video elementary stream or no picture user data is contained therein, then no `dtvcc_frame()` shall be constructed. Only that picture user data whose `user_data_type_code` is the value `0x03` shall be used to construct a `dtvcc_frame()`.

Note: In order to specify a media locator to construct this data source, use the `?dtvcc` query component syntax with a `"tv:"` URI. See [DASE], Section 5.1.2.3.1.4, *Television Scheme*, for further information.

The fields `stc_valid`, `stc_high`, and `stc_low` shall be determined as follows: if the 27MHz *system time clock* (STC) is valid at the time the `dtvcc_frame()` is constructed, then `stc_valid` shall be set to `'1'` and the 33 most significant bits of the 42-bit STC shall be extracted and placed into `stc_high` (STC[41:41]) and `stc_low` (STC[40:9]); otherwise, `stc_valid` shall be set to `'0'`.

Note: The 33 most significant bits of the 42-bit STC represent the STC divided by 300, not divided by 512; that is, the lower 9 bits of the 42-bit STC count up to 300, not to 512. Consequently, the upper 33 bits of the 42-bit STC represent a 90,000 KHz clock.

The determination of the values of fields `stc_valid`, `stc_high`, and `stc_low` shall take place during the read operation performed by the DASE Application; i.e., during the invocation of the method `javax.tv.media.protocol.PushSourceStream2.readStream(…)`. A DASE System should minimize the latency between the determination of the values of these fields and the completion of (return from) this method.

The fields `pts_valid`, `pts_high`, and `pts_low` shall be determined as follows: if a *presentation time stamp* (PTS) is or can be associated with the picture user data from which the `dtvcc_frame()` is constructed, then `pts_valid` shall be set to `'1'` and the 33 significant bits of the associated PTS shall be extracted and placed into `pts_high` (PTS[32:32]) and `pts_low` (PTS[31:0]); otherwise, `pts_valid` shall be set to `'0'`.

The values of the two fields marked as `reserved` are expressly not defined by this specification; a DASE-1 Application shall not rely upon the values of these fields being any specific value.

Note: The values of these reserved fields may be assigned a specific, standardized value in a subsequent level of the DASE Standard.

The remaining fields of `dtvcc_frame()` shall be directly extracted from the identically named fields of the `user_data()` structure as specified by [A/53], Section 5.2.2, User Data Syntax.

A `dtvcc_frame()` instance shall correspond one-to-one with a `user_data()` instance; that is, multiple `user_data()` instances shall not be combined to form a single `dtvcc_frame()`.

A DASE System should minimize any system latency when constructing a `dtvcc_frame()` from picture user data and when performing the subsequent invocation of `SourceTransfer-Handler.transferData(…)`.

At most one `dtvcc_frame()` shall be transferred in a single invocation of `javax.tv.-media.protocol.PushSourceStream2.readStream(…)`. If fewer than 104 bytes are requested in

a read, then zero bytes shall be returned irrespective of whether a `dtvcc_frame()` is available for reading or not.

If data frames arrives more quickly than read operations occur, buffer overflow may occur, in which case a `javax.tv.media.protocol.DataLostException` shall be raised upon the next read operation.

5.1.1.2.2.4.2 Built-In Players

A procedural application environment shall implement the following built-in media players:

- video/mpeg
- video/mpv
- audio/ac3

Each built-in media player shall implement the interface `javax.media.Clock`; however, the implementation need not expose a time base or media time which is derived from content being controlled by the player. A DASE Application shall not rely upon the accuracy or synchronization of time information returned by a built-in player's `javax.media.Clock` interface.

5.1.1.2.2.4.2.1 video/mpeg

A procedural application environment shall implement a built-in data player as follows:

Table 7 Player Parameters: video/mpeg

Parameter	Value	Notes
content type	`"video/mpeg"`	1
controls	none required	2
data source access	not required	3

Notes

1. See [DASE], Section 6.5.1.
2. If applicable, a procedural application environment should support the following controls on this player: `javax.media.GainControl`, `javax.tv.media.MediaSelectControl`, `org.davic.media.AudioLanguageControl`, and `org.davic.media.SubtitlingLanguage-Control`.
3. Access to the data source for this player need not be provided to an application.

5.1.1.2.2.4.2.2 video/mpv

A procedural application environment shall implement a built-in data player as follows:

Table 8 Player Parameters: video/mpv

Parameter	Value	Notes
content type	`"video/mpv"`	1
controls	none required	2
data source access	not required	3

Notes

1. See [DASE], Section 6.5.2.
2. If applicable, a procedural application environment should support the following controls on this player: `org.davic.media.SubtitlingLanguageControl`.
3. Access to the data source for this player need not be provided to an application.

5.1.1.2.2.4.2.3 audio/ac3

A procedural application environment shall implement a built-in data player as follows:

Table 9 Player Parameters: audio/ac3

Parameter	Value	Notes
content type	`"audio/ac3"`	1
controls	none required	2
data source access	not required	3

Notes

1. See [DASE], Section 6.6.1.
2. If applicable, a procedural application environment should support the following controls on this player: `javax.media.GainControl` and `org.davic.media.AudioLanguageControl`.
3. Access to the data source for this player need not be provided to an application.

5.1.1.2.3 Java Television (Java TV) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall support the use of Java TV Specification, Version 1.0, APIs in a manner consistent with [JAVATV] and [JAVATV-INTRO], as extended and restricted below.

Note: See Annexes A.16 through A.28 for all required Java TV types.

5.1.1.2.3.1 Constraints

5.1.1.2.3.1.1 Constraints on `javax.tv.carousel.CarouselFile` class

The following constructors of `javax.tv.carousel.CarouselFile` shall be used only to access file or directory objects of an application delivery file system. If the `CarouselFile` instance would fail to associate with an application delivery file system object, then a `java.io.IOException` shall be raised.

- `CarouselFile(java.lang.String)`
- `CarouselFile(java.lang.String, java.lang.String)`
- `CarouselFile(javax.tv.carousel.CarouselFile, java.lang.String)`

Note: The constructor `CarouselFile(javax.tv.locator.Locator)` may be used to construct a `CarouselFile` instance associated with an unbounded resource which is not associated with an application delivery file system and which has no corresponding path in the local file system; e.g., an unbounded resource encapsulated as a module in the data carousel.

If an application entity invokes the inherited method `canWrite()`, then the value `false` shall be returned; furthermore, a procedural application environment shall not permit any write or modify operation to be performed by an application upon the underlying resource which represents a carousel file or directory object.

In the case that a CarouselFile instance is associated with a file object that has no corresponding path in the local file system, then invocation of the method `CarouselFile.getCanonicalPath()` shall cause a `java.io.IOException` or a subclass thereof to be raised.

When a method of `CarouselFile` would throw an instance of `java.io.IOException`, it shall instead throw an instance of the class `org.atsc.carousel.CarouselException` or a subclass thereof.

5.1.1.2.3.1.2 Constraints on `javax.tv.graphics.TVContainer` class

When returning normally, the method `getRootContainer()` shall return an instance of `org.havi.ui.HScene`.

If an *embedded* Xlet has an initial width or height of zero, then `getRootContainer()` shall return `null`. If an Xlet ever returns `null` from this method, then it shall never subsequently return a value which is not `null`.

Note: See Section 3.3 for the definition of *embedded* Xlet.

5.1.1.2.3.1.3 Constraints on `javax.tv.locator.Locator` interface

For any instance of `Locator` *`loc`*, the value of *`loc`*`.toExternalForm()` shall be a value such that the following method returns true.

```
import javax.media.MediaLocator;
…
boolean testLocator ( Locator loc )
{
  MediaLocator ml      = new MediaLocator ( loc.toExternalForm() );
  Locator      locTest = LocatorFactory.createLocator ( ml.toExternalForm() );
  return locTest.equals ( loc );
}
```

Note: An application delivery system may impose additional constraints on the external form of a `Locator`.

5.1.1.2.3.1.4 Constraints on `javax.tv.media.protocol.PushSourceStream2` interface

An application entity shall not invoke the following method on a class which implements the `PushSourceStream2` interface:

- `read(byte[],int,int)`

An attempt by an application entity to invoke this method may cause a runtime exception to be raised.

Note: The method `read(byte[],int,int)` is inherited from the super-interface `javax.media.protocol.PushSourceStream`. An application entity using the derived interface PushSourceStream2 is expected to use `readStream(byte[],int,int)`.

5.1.1.2.3.1.5 Constraints on `javax.tv.service.SIManager` class

The following methods shall employ a language identifier which adheres to the syntax prescribed by [LANG-TAGS]:

- `getPreferredLanguage()`
- `setPreferredLanguage(java.lang.String)`

5.1.1.2.3.1.6 Constraints on `javax.tv.service.selection.ServiceContext` interface

When selecting a service consisting of one or more Xlets and when returning normally, the method `getServiceContentHandlers()` shall return an array of one or more objects which implement `org.atsc.xlet.XletComponentPresenterProxy`. Furthermore, the Xlet whose proxy is represented by this `XletComponentPresenterProxy` shall be the Xlet which invokes this method.

Note: See [DASE-API] for further information on `XletComponentPresenterProxy`.

The method `getServiceContentHandlers()` should return an object corresponding to the active video and audio service components of the current service, if present.

> *Note:* See Section 5.1.1.2.4.1.5 for further information on `getServiceContent-Handlers()` as apply to a HAVi video device.

If an application makes use of multiple Xlets, then the objects returned by the method `getServiceContentHandlers()` shall be distinct for a distinct Xlet; however, the underlying content handlers which these objects represent shall be identical. For example, if an underlying video content handler H_V and an underlying audio content handler H_A are available within a service context, then for two Xlets X_1 and X_2 in a single application, two distinct sets of objects would be returned by this method, one for X_1: { $O_{V,1}$, $O_{A,1}$ } and another for X_2: { $O_{V,2}$, $O_{A,2}$ }; moreover, both $O_{V,1}$ and $O_{V,2}$ would control H_V, and both $O_{A,1}$ and $O_{A,2}$ would control H_A.

> *Note:* Any synchronization which might be required in order for two Xlets to simultaneously control the same underlying content handler is implementation dependent. Content authors may wish to make use of application-level synchronization techniques to coordinate control of the same handler by multiple Xlets.

If an object returned by `getServiceContentHandlers()` implements the `ServiceMedia-Handler` interface, then the JMF `Player` represented by the `ServiceMediaHandler` shall be in the *started* state, and, if it is presenting video, then that video shall be presenting on the background video device.

> *Note:* See [JMF] `javax.media.Player` interface description for further information on a player's *started* state.

If an application is resumed after service selection occurs, then an application shall not use any previously acquired service content handler returned by `getServiceContentHandlers()`, but shall re-acquire the service content handlers. A procedural application environment need not re-use the same service content handlers for a given application after service selection. If a service content handler is re-used and it supports the `ServiceMediaHandler` interface, then the JMF `Player` represented by the `ServiceMediaHandler` shall be reset to a default condition. In particular, all previously registered listeners shall be silently unregistered and any video scaling or positioning shall be reset.

5.1.1.2.3.1.7 Constraints on `javax.tv.service.selection.ServiceContextFactory` class

An application entity shall not subclass `ServiceContextFactory`; furthermore the protected constructor `ServiceContextFactory()` shall not be used by an application entity.

The method `getServiceContexts()` shall return no more than one instance of a `ServiceContext`; furthermore, that instance shall be the same instance as returned by `getServiceContext(XletContext)` where the `XletContext` argument is associated with the invoking Xlet.

If an application makes use of multiple Xlets, then the objects returned by the method `getServiceContexts()` and `getServiceContext(XletContext)` shall be distinct for a distinct Xlet; however, the underlying service context resources which these objects represent shall be identical.

5.1.1.2.3.1.8 Constraints on `javax.tv.service.selection.ServiceMediaHandler` interface

When returning normally, the method `getVisualComponent()` shall return an instance of `org.havi.ui.HVideoComponent`.

When returning normally and when returning a value other than `null`, the method `getControlPanelComponent()` shall return an instance of `org.havi.ui.HComponent`.

Note: See [HAVI-UI-API] for further information on `HVideoComponent` and `HComponent`.

5.1.1.2.3.1.9 Constraints on use of `javax.tv.xlet.Xlet` interface

The argument to the method `initXlet(XletContext)` shall implement `org.atsc.xlet.XletContextExt`.

5.1.1.2.3.2 Modifications

5.1.1.2.3.2.1 Modifications to `javax.tv.graphics.AlphaColor` class

The following inherited fields in the `javax.tv.graphics.AlphaColor` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these fields:

- `BITMASK`
- `OPAQUE`
- `TRANSLUCENT`

The following inherited methods in the `javax.tv.graphics.AlphaColor` class are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these methods:

- `createContext(…)`
- `getColorComponents(float[])`
- `getColorComponents(ColorSpace,float[])`
- `getColorSpace()`
- `getComponents(float[])`
- `getComponents(ColorSpace,float[])`
- `getRGBColorComponents(float[])`
- `getRGBComponents(float[])`
- `getTransparency()`

Note: The excluded fields and methods described above were erroneously specified in [JAVATV] as inherited from `java.awt.Transparency` and `java.awt.Color` as specified by [JDK1.2.2]; however, neither [JAVATV] nor [PJAE] includes the [JDK1.2.2] definitions of these types.

5.1.1.2.3.2.2 Modifications to `javax.tv.service.transport` package

The following interfaces in the `javax.tv.service.transport` package shall be provided by a procedural application environment; however, no class is required to implement these interfaces:

- `Bouquet`
- `BouquetChangeListener`
- `BouquetCollection`
- `Network`
- `NetworkChangeListener`
- `NetworkCollection`

Note: The ATSC Transport Stream, to which the DASE application delivery system is bound by default, does not support the semantics of these types; therefore, a procedural application environment which supports only the default application delivery system is not expected to implement or otherwise use these interfaces.

5.1.1.2.4 Home Audio Video Interoperability User Interface (HAVi UI) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall implement packages `org.havi.ui` and `org.havi.ui.event` as specified in HAVi Level 2 User Interface APIs, Version 1.1, May 15, 2001, [HAVI-UI-API], according to the semantics specified by The HAVi Specification, Chapter 8, Level 2 User Interface, Version 1.1, [HAVI-UI], as extended and restricted below.

> *Note:* Within [HAVI-UI] and [HAVI-UI-API], the term *application* is to be equated with an individual Xlet, and not with a DASE Application, which may consist of multiple Xlets.
>
> *Note:* See Annexes A.48 and A.49 for all required HAVi types.

5.1.1.2.4.1 Constraints

5.1.1.2.4.1.1 Constraints on `org.havi.ui.HGraphicsConfiguration` class

When invoked by an application, the following methods shall produce no side effect and shall raise a runtime exception:

- `getPunchThroughToBackgroundColor(java.awt.Color,int)`
- `getPunchThroughToBackgroundColor(java.awt.Color,int,HVideoDevice)`
- `getPunchThroughToBackgroundColor(int)`
- `getPunchThroughToBackgroundColor(int,HVideoDevice)`

5.1.1.2.4.1.2 Constraints on `org.havi.ui.HPermissionDeniedException` class

The `HPermissionDeniedException` shall be used only for resource management purposes and not for security purposes; it shall only be used to signal a transient state.

5.1.1.2.4.1.3 Constraints on `org.havi.ui.HScene` class

Within a procedural application, the `HScene` associated with the primary Xlet shall receive event focus by default. Only one instance of `HScene` shall be granted focus at any given time.

> *Note:* The determination of when another Xlet in a procedural application should be granted focus is not defined by this specification. An Xlet may request focus be granted to an `HScene` by means of the method `java.awt.Component.requestFocus()` which is inherited by `HScene`.

When invoked by an Xlet within a declarative application, the inherited method `getBounds()` shall return a rectangle whose origins are `(0,0)`. When invoked by an Xlet within a declarative application, the inherited method `getLocation()` shall return a point whose *x* and *y* coordinates are `(0,0)`.

When invoked by an Xlet within a declarative application, the following methods shall produce no side effect and may raise a runtime exception:

- `setBounds(int,int,int,int)`
- `setBounds(Rectangle)`
- `setLocation(int,int)`
- `setLocation(Point)`
- `setSize(int,int)`
- `setSize(Dimension)`

> *Note:* An Xlet in a declarative application may request the declarative application environment cause its position or size to be changed by making use of functionality provided by the `org.w3c.dom.css` package.

5.1.1.2.4.1.4 Constraints on `org.havi.ui.HSceneFactory` class

The value returned by the `HSceneFactory.getDefaultHScene` methods shall be the same value as returned by `javax.tv.graphics.TVContainer.getRootContainer()`. If this value is not `null`, then it shall be a distinct instance of `HScene` for each Xlet within a single DASE Application.

> *Note:* In the case of an Xlet within a procedural application, the region of a screen which is allocated to an `HScene` is not defined by this specification.

A procedural application environment is not required to satisfy a request to create an instance of `HScene` using the method `HSceneFactory.getBestScene()`.

A procedural application environment is not required to satisfy a request to resize an instance of `HScene` using the method `HSceneFactory.resizeScene()`; furthermore, this request shall not be satisfied when invoked by an Xlet within a declarative application.

When invoked by an Xlet within a declarative application, the method `HSceneFactory.-getFullScreenScene()` shall return `null`.

5.1.1.2.4.1.5 Constraints on `org.havi.ui.HVideoDevice` class

When returning normally, the method `HVideoDevice.getVideoController()` shall return an instance of `javax.tv.service.selection.ServiceMediaHandler` which is in a *prefetched* or *started* state; furthermore, it shall only return a media handler which has already been created in response to the application invoking `javax.media.Manager.createPlayer` or which has been returned or would be returned by `javax.tv.service.selection.getServiceContentHandlers`.

When returning normally, the method `HVideoDevice.getVideoSource()` shall return an instance of `javax.media.protocol.DataSource`. If a video source is not available due to implementation restrictions, then an `HPermissionDeniedException` shall be raised.

5.1.1.2.5 Digital Audio Video Council (DAVIC) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall implement packages `org.davic.media` and `org.davic.-resources` as specified by the following subsections and by [HAVI-UI-API], respectively.

> *Note:* See Annexes A.46 and A.47 for all required DAVIC types.

5.1.1.2.5.1 Definitions

5.1.1.2.5.1.1 Definition of `org.davic.media.AudioLanguageControl` interface

```
public interface AudioLanguageControl
    extends LanguageControl
```

This interface may be implemented by a JMF control in order to provide control over the selection of the language of an audio program element.

5.1.1.2.5.1.1.1 Methods

The following methods are inherited from `LanguageControl`: `getCurrentLanguage`, `listAvailableLanguages`, `selectDefaultLanguage`, and `selectLanguage`.

The following methods are inherited from `javax.media.Control`: `getControlComponent`.

5.1.1.2.5.1.1.2 Fields

No fields are defined.

5.1.1.2.5.1.2 Definition of `org.davic.media.LanguageControl` interface

```
public interface LanguageControl
    extends javax.media.Control
```

This interface is the base interface for both audio and subtitling language controls. This interface should never be implemented in a control alone, but always either as an audio or subtitling language control.

5.1.1.2.5.1.2.1 Methods

The following methods are inherited from `javax.media.Control`: `getControlComponent`.

5.1.1.2.5.1.2.1.1 `getCurrentLanguage()`

```
public java.lang.String getCurrentLanguage()
```

Obtain current language selected.

Returns:

A string which denotes the current language selection. The syntax of this string shall be that prescribed by [LANG-TAGS]. If the language selection is unknown, then the returned string shall be empty.

5.1.1.2.5.1.2.1.2 `listAvailableLanguages()`

```
public java.lang.String[] listAvailableLanguages()
```

Provides a list of available languages. If there are no selectable languages, the returned array is of zero length.

Returns:

A (possibly empty) array of strings, each of which denote a language. The syntax of each string shall be that prescribed by [LANG-TAGS].

5.1.1.2.5.1.2.1.3 `selectDefaultLanguage()`

```
public java.lang.String[] selectDefaultLanguage()
    throws NotAuthorizedException
```

Selects the default language.

Returns:

A string which denotes the default language selection. The syntax of this string shall be that prescribed by [LANG-TAGS]. If the default language selection is unknown, then the returned string shall be empty.

Throws:

`NotAuthorizedException` – if access to the default language is not permitted.

5.1.1.2.5.1.2.1.4 `selectLanguage(java.lang.String)`

```
public void selectLanguage(java.lang.String lang)
    throws LanguageNotAvailableException, NotAuthorizedException
```

Changes the language selection to the indicated language.

Parameters:

lang – a language tag which adheres to the syntax prescribed by [LANG-TAGS]

Throws:

`LanguageNotAvailableException` – if the specified language is not available.

`NotAuthorizedException` – if access to the default language is not permitted.

5.1.1.2.5.1.2.2 Fields

No fields are defined.

5.1.1.2.5.1.3 Definition of `org.davic.media.LanguageNotAvailableException` class

```
public class LanguageNotAvailableException
    extends javax.media.MediaException
```

This exception is thrown when the requested language is not available.

5.1.1.2.5.1.3.1 Constructors

5.1.1.2.5.1.3.1.1 `LanguageNotAvailableException()`

```
public LanguageNotAvailableException()
```

Constructor with no detail message.

5.1.1.2.5.1.3.1.2 `LanguageNotAvailableException(java.lang.String)`

```
public LanguageNotAvailableException(java.lang.String reason)
```

Constructor taking a detail message.

Parameters:

reason – the reason this exception was thrown

5.1.1.2.5.1.3.2 Methods

The following methods are inherited from `java.lang.Throwable`: `fillInStackTrace`, `getLocalizedMessage`, `getMessage`, `printStackTrace()`, `printStackTrace(java.io.PrintStream)`, `printStackTrace(java.io.PrintWriter)`, and `toString`.

The following methods are inherited from `java.lang.Object`: `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait()`, `wait(long)`, and `wait(long,int)`.

5.1.1.2.5.1.3.3 Fields

No fields are defined.

5.1.1.2.5.1.4 Definition of `org.davic.media.MediaPresentedEvent`

```
public class MediaPresentedEvent
    extends javax.media.ControllerEvent
```

Generated as soon as possible after new content is actually being presented to the end-user, regardless of whether a state change has taken place in the player or not.

5.1.1.2.5.1.4.1 Constructors

5.1.1.2.5.1.4.1.1 `MediaPresentedEvent(javax.media.Controller)`

```
public MediaPresentedEvent(javax.media.Controller source)
```

Constructs a `MediaPresentedEvent`.

Parameters:

source – the controller concerned

5.1.1.2.5.1.4.2 Methods

The following methods are inherited from `javax.media.ControllerEvent`: `getSource` and `getSourceController`.

The following methods are inherited from `java.util.EventObject`: `toString`.

The following methods are inherited from `java.lang.Object`: `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait()`, `wait(long)`, and `wait(long,int)`.

5.1.1.2.5.1.4.3 Fields

The following fields are inherited from `java.util.EventObject`: `source`.

5.1.1.2.5.1.5 Definition of `org.davic.media.NotAuthorizedException` class

```
public class NotAuthorizedException
    extends java.security.AccessControlException
```

This exception is thrown when the source cannot be accessed in order to reference the new content or the source has not been accepted.

5.1.1.2.5.1.5.1 Constructors

5.1.1.2.5.1.5.1.1 `NotAuthorizedException()`

```
public NotAuthorizedException()
```

Constructor with no detail message.

5.1.1.2.5.1.5.1.2 `NotAuthorizedException(String)`

```
public NotAuthorizedException(java.lang.String reason)
```

Constructor taking a detail message.

Parameters:

reason – the reason this exception was thrown

5.1.1.2.5.1.5.2 Methods

The following methods are inherited from `java.lang.Throwable`: `fillInStackTrace`, `getLocalizedMessage`, `getMessage`, `printStackTrace()`, `printStackTrace(java.io.PrintStream)`, `printStackTrace(java.io.PrintWriter)`, and `toString`.

The following methods are inherited from `java.lang.Object`: `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait()`, `wait(long)`, and `wait(long,int)`.

5.1.1.2.5.1.5.3 Fields

No fields are defined.

5.1.1.2.5.1.6 Definition of `org.davic.media.SubtitlingLanguageControl` interface

```
public interface SubtitlingLanguageControl
    extends LanguageControl
```

This interface may be implemented by a JMF control in order to provide control over the selection of the language of an available subtitling service.

> *Note:* It is not required that any DASE System class implement this interface. This interface may be used by application defined JMF components to control subtitling available in application defined data.
>
> *Note:* This functionality is not intended to be used to control closed captioning services delivered via [A/53] User Private Data.

5.1.1.2.5.1.6.1 Methods

The following methods are inherited from `LanguageControl`: `getCurrentLanguage`, `listAvailableLanguages`, `selectDefaultLanguage`, and `selectLanguage`.

The following methods are inherited from `javax.media.Control`: `getControlComponent`.

5.1.1.2.5.1.6.1.1 `isSubtitlingOn()`

```
public boolean isSubtitlingOn()
```

Indicates whether subtitling is on or off.

Returns:

If subtitling is on, returns `true`; otherwise, returns `false`.

5.1.1.2.5.1.6.1.2 `setSubtitling(boolean)`

```
public boolean setSubtitling(boolean newValue)
```

Changes subtitling to on or off.

Parameters:

newValue – if `true`, then turns subtitling on; otherwise, turns it off.

Returns:

The previous value.

5.1.1.2.5.1.6.2 Fields

No fields are defined.

5.1.1.2.5.2 Modifications

5.1.1.2.5.2.1 Modification to `org.davic.resource.ResourceStatusEvent` class

The class of `ResourceStatusEvent` shall be a subtype of the `java.util.EventObject` class.

5.1.1.2.6 W3C Document Object Model (DOM) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall implement packages `org.w3c.dom` , `org.w3c.dom.css`, `org.w3c.dom.events`, `org.w3c.dom.html2`, `org.w3c.dom.stylesheets`, and

`org.w3c.dom.views` as specified in [DOM2], [DOM2-EVENTS], [DOM2-HTML], [DOM2-STYLE], and [DOM2-VIEWS] as extended and restricted below.

Note: See Annexes A.50 through A.55 for all required W3C DOM types.

5.1.1.2.6.1 Modifications

5.1.1.2.6.1.1 Modifications to `org.w3c.dom.css` package

The following interfaces in the `org.w3c.dom.css` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these interfaces:

- `CSSPageRule`
- `CSS2Properties`

Note: The functionality of the `CSS2Properties` interface may be obtained by using other interfaces defined in package `org.w3c.dom.css`, particularly the `ElementCSSInlineStyle` and `CSSStyleDeclaration` interfaces.

5.1.1.2.6.1.2 Modifications to `org.w3c.dom.html2` package

The following interfaces in the `org.w3c.dom.html2` package are not required to be implemented by a procedural application environment; furthermore, an application entity shall not rely on the presence of support for these interfaces:

- `HTMLAppletElement`
- `HTMLAreaElement`
- `HTMLBaseElement`
- `HTMLBaseFontElement`
- `HTMLBRElement`
- `HTMLButtonElement`
- `HTMLDirectoryElement`
- `HTMLDivElement`
- `HTMLDListElement`
- `HTMLFieldSetElement`
- `HTMLFontElement`
- `HTMLFrameElement`
- `HTMLFrameSetElement`
- `HTMLHeadElement`
- `HTMLHeadingElement`
- `HTMLHRElement`
- `HTMLHtmlElement`
- `HTMLIFrameElement`
- `HTMLIsIndexElement`
- `HTMLLabelElement`
- `HTMLLegendElement`
- `HTMLLIElement`
- `HTMLLinkElement`
- `HTMLMapElement`
- `HTMLMenuElement`
- `HTMLMetaElement`
- `HTMLModElement`
- `HTMLOListElement`

- HTMLOptGroupElement
- HTMLParagraphElement
- HTMLParamElement
- HTMLPreElement
- HTMLQuoteElement
- HTMLScriptElement
- HTMLStyleElement
- HTMLTableCaptionElement
- HTMLTableCellElement
- HTMLTableColElement
- HTMLTableElement
- HTMLTableRowElement
- HTMLTableSectionElement
- HTMLTitleElement
- HTMLUListElement

Note: The functionality of the preceding interfaces may be obtained by using interfaces defined in package org.w3c.dom, particularly the Node, Element, and Attr interfaces.

5.1.1.2.7 DASE Specific (ATSC) Interfaces

An application entity which adheres to this content type may use and a procedural application environment shall implement all facilities specified in DASE Application Programming Interface [DASE-API].

All package, class, and interface names that have the prefix org.atsc are reserved for the sole use of the ATSC. An implementation of a procedural application environment shall not superset the org.atsc package namespace unless required or permitted by a future level or edition of the DASE Standard.

If a procedural application environment makes use of a local file system to persist DASE specific registry state, including ApplicationRegistry, PreferencesRegistry, and UserRegistry, and access to this file system is provided to application entities by means of java.io functionality, then no access to this registry state shall be granted to application entities by means of java.io functionality.

Note: See Annexes A.29 through A.45 for all required ATSC types.

5.1.1.2.7.1 Constraints

5.1.1.2.7.1.1 Constraints on org.atsc.preferencees.PreferenceRegistry interface

A procedural application environment shall maintain any state changes due to the addition or removal of preferences through the PreferenceRegistry interface for no less time than the duration of the current application.

5.1.1.2.7.1.2 Constraints on org.atsc.user.UserRegistry interface

A procedural application environment shall maintain any state changes due to the creation or deletion of users through the UserRegistry interface for no less time than the duration of the current application.

5.1.1.2.7.1.3 Constraints on `org.atsc.xlet.XletRegistry` interface

A procedural application environment shall maintain any state changes due to registration of Xlets through the `XletRegistry` interface for no less time than the duration of the current application.

> *Note:* An application's duration is the interval from when an application is placed in the initialized state until the time when it is placed in the uninitialized state.

5.1.1.3 Interface Implementation Constraints

This section describes additional constraints on active object content and on a procedural application environment's implementation of the interfaces specified in Section 5.1.1.2, Java Application Programming Interfaces.

5.1.1.3.1 Instance Sharing

A procedural application environment shall not permit direct instance sharing of application-defined classes between two or more Xlets.

> *Note:* Two Xlets may reference the same system object in certain cases. For example, if two Xlets request the same system property, the same string instance may be returned. This kind of sharing is harmless, because String instances are immutable.

5.1.1.3.2 Finalizers

An Xlet shall not rely upon an application-defined finalizer being invoked by the procedural application environment.

5.1.1.3.3 Class Loaders

A procedural application environment shall associate a distinct `java.lang.ClassLoader` instance with each Xlet instance. This `java.lang.ClassLoader` instance shall be used to load all of the classes that comprise a distinct Xlet.

5.1.2 application/javatv-xlet

This active object content type shall adhere to the specification of `application/java` content as specified above, and, in addition, shall implement the `javax.tv.xlet.Xlet` interface as specified by [JAVATV]. This content shall be identified as content type `application/javatv-xlet`.

5.2 Application Defined Content

This facility consists of a generic content type which may be used by an application to implement its own content types.

5.2.1 application/octet-stream

This content type shall consist of an arbitrary length sequence of arbitrary octets (8-bit bytes).

5.3 Text Content

This facility consists of a generic content type which is used in specific cases to represent text oriented data.

5.3.1 text/plain

This content type shall consist of an arbitrary length sequence of arbitrary encoded characters, where such a sequence is referred to generically as *plain text*.

An application entity which employs this content type shall specify one of the following character encoding systems in a `charset` parameter in accordance with [MIME-MEDIA], Section 4.1.2, and, furthermore, the entity's representation shall employ the specified encoding system as its actual character encoding system:

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

If no character encoding system is specified or if the actual character encoding system does not correspond to the specified encoding system, then the application entity shall be considered to be not well-formed, and shall cause an exception to be raised if an attempt is made to read the entity.

> *Note:* In DASE-1, the use of this content type is limited to two cases: (1) to represent a property list resource (see Section 5.1.1.2.1.1.10.4), and (2) to represent textual oriented application defined content.

5.4 Java Archive Content

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

Table 10 Java Archive Content Types

`application/jar`	Java Archive

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 Java archive content are not presented as such; in contrast, application resources embodied within Java archive content may be presented according to their specific content types.

5.4.1 application/jar

An application entity identified as content type `application/jar` shall consist of a Java Archive. A Java Archive shall adhere to the `application/zip` content type as defined by [DASE], Section 6.8.1, and additionally shall contain a manifest entry which adheres to [JAR].

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 by the `application/zip` content type and those specified by [JAR].

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

If a `Content-Type` per-entry attribute is present in a manifest entry, then its value shall be identical to the value of the `Content-Type` MIME header extension which applies to the entry in accordance with [DASE], Section 6.8.1. In case of a conflict, the MIME header extension shall be given priority.

A procedural application environment is not required to check or otherwise verify the validity of any digital signature specified in a manifest entry; furthermore, an application entity shall not rely on the presence of support for validating a digital signature present in a manifest entry.

> *Note:* Limited programmatic access to an application entity that uses content type `application/jar` may be obtained by use of functionality defined in the `java.util.zip` package.

ANNEX A. REQUIRED JAVA TYPES

The entirety of this annex and its subsections is normative.

This annex specifies, on a package by package basis, all system-defined Java types which may be referenced by active object content and which shall be supported by a procedural application environment.

In addition to the Java types specified here, additional binding dependent types may be required according the applicable application delivery system.

If active object content requires resolution of a reference to a Java type other than (1) the types specified by this annex, (2) binding dependent types specified by the application delivery system, or (3) an application-defined type, and the active object content is processed, then a procedural application environment may destroy the Xlet in whose context the reference occurs.

> *Note:* Each of the following subsections specifies all required Java types for a specific package. No significance is to be attributed to the tabular representation of these types; they are specified in alphabetic order from top to bottom in multiple columns.

A.1 java.awt

Adjustable	Cursor	Image
AWTError	Dimension	Insets
AWTEvent	EventQueue	ItemSelectable
AWTEventMulticaster	FlowLayout	LayoutManager
AWTException	Font	LayoutManager2
AWTPermission	FontMetrics	MediaTracker
BorderLayout	Graphics	Point
CardLayout	GridBagConstraints	Polygon
Color	GridBagLayout	Rectangle
Component	GridLayout	Shape
Container	IllegalComponentStateException	Toolkit

A.2 java.awt.event

ActionEvent	FocusAdapter	MouseEvent
ActionListener	FocusEvent	MouseListener
AdjustmentEvent	FocusListener	MouseMotionAdapter
AdjustmentListener	InputEvent	MouseMotionListener
ComponentAdapter	ItemEvent	PaintEvent
ComponentEvent	ItemListener	TextEvent
ComponentListener	KeyAdapter	TextListener
ContainerAdapter	KeyEvent	WindowAdapter
ContainerEvent	KeyListener	WindowEvent
ContainerListener	MouseAdapter	WindowListener

A.3 java.awt.image

AreaAveragingScaleFilter	ImageConsumer	MemoryImageSource
ColorModel	ImageFilter	PixelGrabber
CropImageFilter	ImageObserver	ReplicateScaleFilter
DirectColorModel	ImageProducer	RGBImageFilter
FilteredImageSource	IndexColorModel	

A.4 java.beans

Beans	PropertyChangeSupport	VetoableChangeSupport
PropertyChangeEvent	PropertyVetoException	Visibility
PropertyChangeListener	VetoableChangeListener	

A.5 java.io

BufferedInputStream	FileWriter	OutputStreamWriter
BufferedOutputStream	FilterInputStream	PipedInputStream
BufferedReader	FilterOutputStream	PipedOutputStream
BufferedWriter	FilterReader	PipedReader
ByteArrayInputStream	FilterWriter	PipedWriter
ByteArrayOutputStream	InputStream	PrintStream
CharArrayReader	InputStreamReader	PrintWriter
CharArrayWriter	InterruptedIOException	PushbackInputStream
CharConversionException	InvalidClassException	PushbackReader
DataInput	InvalidObjectException	RandomAccessFile
DataInputStream	IOException	Reader
DataOutput	LineNumberReader	SequenceInputStream
DataOutputStream	NotActiveException	Serializable
EOFException	NotSerializableException	SerializablePermission
Externalizable	ObjectInput	StreamCorruptedException
File	ObjectInputStream	StreamTokenizer
FileDescriptor	ObjectInputValidation	StringReader
FileInputStream	ObjectOutput	StringWriter
FilenameFilter	ObjectOutputStream	SyncFailedException
FileNotFoundException	ObjectStreamClass	UnsupportedEncodingException
FileOutputStream	ObjectStreamException	UTFDataFormatException
FilePermission	OptionalDataException	WriteAbortedException
FileReader	OutputStream	Writer

A.6 java.lang

AbstractMethodError	IllegalMonitorStateException	OutOfMemoryError
ArithmeticException	IllegalStateException	Runnable
ArrayIndexOutOfBoundsException	IllegalThreadStateException	Runtime
ArrayStoreException	IncompatibleClassChangeError	RuntimeException
Boolean	IndexOutOfBoundsException	RuntimePermission
Byte	InstantiationError	SecurityException
Character	InstantiationException	SecurityManager
Class	Integer	Short
ClassCastException	InternalError	StackOverflowError
ClassCircularityError	InterruptedException	String
ClassFormatError	LinkageError	StringBuffer
ClassLoader	Long	StringIndexOutOfBoundsException
ClassNotFoundException	Math	System
Cloneable	NegativeArraySizeException	Thread
CloneNotSupportedException	NoClassDefFoundError	ThreadDeath
Double	NoSuchFieldError	ThreadGroup
Error	NoSuchFieldException	Throwable
Exception	NoSuchMethodError	UnknownError
ExceptionInInitializerError	NoSuchMethodException	UnsatisfiedLinkError
Float	NullPointerException	UnsupportedOperationException
IllegalAccessError	Number	VerifyError
IllegalAccessException	NumberFormatException	VirtualMachineError
IllegalArgumentException	Object	Void

A.7 java.lang.reflect

AccessibleObject	Field	Method
Array	InvocationTargetException	Modifier
Constructor	Member	ReflectPermission

A.8 java.net

BindException	MalformedURLException	SocketPermission
DatagramPacket	MulticastSocket	UnknownHostException
DatagramSocket	ProtocolException	URL
InetAddress	SocketException	URLEncoder

A.9 java.security

AccessControlContext	KeyException	ProtectionDomain
AccessControlException	MessageDigest	Provider
AccessController	MessageDigestSpi	ProviderException
AllPermission	NoSuchAlgorithmException	PublicKey
BasicPermission	NoSuchProviderException	SecureClassLoader
CodeSource	Permission	SecureRandom
DigestException	PermissionCollection	SecureRandomSpi
DigestOutputStream	Permissions	Security
GeneralSecurityException	Policy	SecurityPermission
Guard	Principal	SignatureException
GuardedObject	PrivilegedAction	UnresolvedPermission
InvalidKeyException	PrivilegedActionException	
Key	PrivilegedExceptionAction	

A.10 java.security.cert

Certificate	CertificateEncodingException	CertificateException

A.11 java.text

ChoiceFormat	DecimalFormatSymbols	NumberFormat
DateFormat	FieldPosition	ParseException
DateFormatSymbols	Format	ParsePosition
DecimalFormat	MessageFormat	SimpleDateFormat

A.12 java.util

BitSet	Hashtable	PropertyResourceBundle
Calendar	ListResourceBundle	Random
Date	Locale	ResourceBundle
Dictionary	MissingResourceException	SimpleTimeZone
EmptyStackException	NoSuchElementException	Stack
Enumeration	Observable	StringTokenizer
EventListener	Observer	TimeZone
EventObject	Properties	TooManyListenersException
GregorianCalendar	PropertyPermission	Vector

A.13 java.util.zip

CheckedInputStream	DataFormatException	ZipEntry
CheckedOutputStream	Inflater	ZipException
Checksum	InflaterInputStream	ZipFile
CRC32	ZipConstants	ZipInputStream

A.14 javax.media

CachingControl	GainChangeEvent	NotRealizedError
CachingControlEvent	GainChangeListener	PackageManager
Clock	GainControl	Player
ClockStartedError	IncompatibleSourceException	PrefetchCompleteEvent
ClockStoppedException	IncompatibleTimeBaseException	RateChangeEvent
ConnectionErrorEvent	InternalErrorEvent	RealizeCompleteEvent
Control	Manager	ResourceUnavailableEvent
Controller	MediaError	RestartingEvent
ControllerClosedEvent	MediaEvent	StartEvent
ControllerErrorEvent	MediaException	StopAtTimeEvent
ControllerEvent	MediaHandler	StopByRequestEvent
ControllerListener	MediaLocator	StopEvent
DataStarvedEvent	MediaProxy	StopTimeChangeEvent
DeallocateEvent	MediaTimeSetEvent	StopTimeSetError
Duration	NoDataSourceException	Time
DurationUpdateEvent	NoPlayerException	TimeBase
EndOfMediaEvent	NotPrefetchedError	TransitionEvent

A.15 javax.media.protocol

ContentDescriptor	PullSourceStream	RateRange
Controls	PushDataSource	Seekable
DataSource	PushSourceStream	SourceStream
Positionable	RateConfigurable	SourceTransferHandler
PullDataSource	RateConfiguration	

A.16 javax.tv.carousel

CarouselFile	CarouselFileChangeEvent	CarouselFileListener

A.17 javax.tv.graphics

AlphaColor	TVContainer	

A.18 javax.tv.locator

InvalidLocatorException	LocatorFactory	
Locator	MalformedLocatorException	

A.19 javax.tv.media

AWTVideoSize	MediaSelectControl	MediaSelectListener
AWTVideoSizeControl	MediaSelectEvent	MediaSelectPermission
MediaSelectCARefusedEvent	MediaSelectFailedEvent	MediaSelectSucceededEvent

A.20 javax.tv.media.protocol

DataLostException	PushSourceStream2	

A.21 javax.tv.net

InterfaceMap		

A.22 javax.tv.service

RatingDimension	ServiceType	SIManager
ReadPermission	SIChangeEvent	SIRequest

Service	SIChangeListener	SIRequestFailureType
ServiceInformationType	SIChangeType	SIRequestor
ServiceMinorNumber	SIElement	SIRetrievable
ServiceNumber	SIException	

A.23 javax.tv.service.guide

ContentRatingAdvisory	ProgramSchedule	ProgramScheduleListener
ProgramEvent	ProgramScheduleChangeType	
ProgramEventDescription	ProgramScheduleEvent	

A.24 javax.tv.service.navigation

CAIdentification	ServiceComponentChangeEvent	ServiceList
DeliverySystemType	ServiceComponentChangeListener	ServiceProviderInformation
FavoriteServicesName	ServiceDescription	ServiceTypeFilter
FilterNotSupportedException	ServiceDetails	SIElementFilter
LocatorFilter	ServiceDetailsSIChangeEvent	SortNotAvailableException
PreferenceFilter	ServiceFilter	StreamType
ServiceComponent	ServiceIterator	

A.25 javax.tv.service.selection

AlternativeContentEvent	SelectionFailedEvent	ServiceContextException
InsufficientResourcesException	SelectPermission	ServiceContextFactory
InvalidServiceComponentException	ServiceContentHandler	ServiceContextListener
NormalContentEvent	ServiceContext	ServiceContextPermission
PresentationChangedEvent	ServiceContextDestroyedEvent	ServiceMediaHandler
PresentationTerminatedEvent	ServiceContextEvent	

A.26 javax.tv.service.transport

Bouquet	NetworkChangeListener	TransportStream
BouquetChangeEvent	NetworkCollection	TransportStreamChangeEvent
BouqueChangeListener	ServiceDetailsChangeEvent	TransportStreamChangeListener
BouquetCollection	ServiceDetailsChangeListener	TransportStreamCollection
Network	Transport	
NetworkChangeEvent	TransportSIChangeEvent	

A.27 javax.tv.util

TVTimer	TVTimerSpec	TVTimerWentOffListener
TVTimerScheduleFailedException	TVTimerWentOffEvent	

A.28 javax.tv.xlet

Xlet	XletContext	XletStateChangeException

A.29 org.atsc.application

ApplicationInformation		

A.30 org.atsc.carousel

CarouselException	InsufficientResourceException	NotAuthorizedException
DataDeliveryException	InvalidFormatException	TimeoutException

A.31 org.atsc.dom

DocumentAction	DOMExceptionExt	
DocumentFactory	MultipleDocumentsAction	

A.32 org.atsc.dom.environment

History	Navigator	
Location	Window	

A.33 org.atsc.dom.events

KeyEvent	KeyModifiers	VirtualKeys

A.34 org.atsc.dom.html

HTMLAnchorElementExt	HTMLFormElementExt	HTMLObjectElementExt
HTMLDocumentExt	HTMLImageElementExt	

A.35 org.atsc.dom.views

DocumentViewExt		

A.36 org.atsc.graphics

AtscBufferedImage	FontFactory	FontFormatException

A.37 org.atsc.management

AdministrativeState	OperationalState	StateChangeListener
AlarmStatus	ProceduralStatus	StatusChangeEvent
AvailabilityStatus	SourceIndicator	UsageState
ManagementPermission	StateChangeEvent	
ObjectStates	StateChangeException	

A.38 org.atsc.net

DatagramSocketBufferControl		

A.39 org.atsc.preferences

FavoriteChannelsPreference	Preference	PreferencePermission
InvalidPreferenceException	PreferenceChangeCause	PreferenceRegistry
LanguagePreference	PreferenceChangeEvent	PreferenceRegistryEvent
LanguageScope	PreferenceChangeListener	RatingPreference
PersonalDataPreference	PreferenceNames	

A.40 org.atsc.registry

Registry	RegistryChangeEvent	RegistryFactory
RegistryChangeCause	RegistryChangeListener	RegistryType

A.41 org.atsc.security

AccessDeniedException	AtscPermission	
AtscAllPermission	HAViPermission	

A.42 org.atsc.system

Receiver	ReceiverPropertyNames	

A.43 org.atsc.trigger

TriggerEvent	TriggerListener	TriggerSource

A.44 org.atsc.user

InvalidCapabilityException	UserChangeCause	UserRegistry
InvalidUserException	UserPermission	UserRegistryEvent
UserCapabilities	UserProfile	

A.45 org.atsc.xlet

InvalidXletException	XletComponentPresenterProxy	XletPermission
XletAlreadyRegisteredException	XletContextExt	XletProxy
XletAvailabilityException	XletInformation	XletRegistry
XletChangeCause	XletNotRegisteredException	XletRegistryEvent

A.46 org.davic.media

AudioLanguageControl	LanguageNotAvailableException	NotAuthorizedException
LanguageControl	MediaPresentedEvent	SubtitlingLanguageControl

A.47 org.davic.resources

ResourceClient	ResourceServer	ResourceStatusListener
ResourceProxy	ResourceStatusEvent	

A.48 org.havi.ui

HActionable	HImageEffectMatte	HScreenDimension
HActionInputPreferred	HImageHints	HScreenPoint
HAdjustableLook	HImageMatte	HScreenRectangle
HAdjustmentInputPreferred	HInvalidLookException	HSelectionInputPreferred
HAdjustmentValue	HItemValue	HSinglelineEntry
HAnimateEffect	HKeyboardInputPreferred	HSinglelineEntryLook
HAnimateLook	HListElement	HSound
HAnimation	HListGroup	HState
HBackgroundConfigTemplate	HListGroupLook	HStaticAnimation
HBackgroundConfiguration	HLook	HStaticIcon
HBackgroundDevice	HMatte	HStaticRange
HBackgroundImage	HMatteException	HStaticText
HChangeData	HMatteLayer	HStillImageBackgroundConfiguration
HComponent	HMultilineEntry	HSwitchable
HComponentOrdering	HMultilineEntryLook	HText
HConfigurationException	HNavigable	HTextButton
HContainer	HNavigationInputPreferred	HTextLayoutManager
HDefaultTextLayoutManager	HNoInputPreferred	HTextLook
HEmulatedGraphicsConfiguration	HOrientable	HTextValue
HEmulatedGraphicsDevice	HPermissionDeniedException	HToggleButton
HEventMulticaster	HRange	HToggleGroup
HFlatEffectMatte	HRangeLook	HUIException
HFlatMatte	HRangeValue	HVersion
HFontCapabilities	HScene	HVideoComponent
HGraphicButton	HSceneFactory	HVideoConfigTemplate

HGraphicLook	HSceneTemplate	HVideoConfiguration
HGraphicsConfigTemplate	HScreen	HVideoDevice
HGraphicsConfiguration	HScreenConfigTemplate	HVisible
HGraphicsDevice	HScreenConfiguration	
HIcon	HScreenDevice	

A.49 org.havi.ui.event

HActionEvent	HFocusListener	HScreenConfigurationEvent
HActionListener	HItemEvent	HScreenConfigurationListener
HAdjustmentEvent	HItemListener	HScreenDeviceReleasedEvent
HAdjustmentListener	HKeyCapabilities	HScreenDeviceReservedEvent
HBackgroundImageEvent	HKeyEvent	HScreenLocationModifiedEvent
HBackgroundImageListener	HKeyListener	HScreenLocationModififedListener
HEventGroup	HMouseCapabilities	HTextEvent
HEventRepresentation	HRcCapabilities	HTextListener
HFocusEvent	HRcEvent	

A.50 org.w3c.dom

Attr	DocumentType	NamedNodeMap
CDATASection	DOMException	Node
CharacterData	DOMImplementation	NodeList
Comment	Element	Notation
Document	Entity	ProcessingInstruction
DocumentFragment	EntityReference	Text

A.51 org.w3c.dom.css

Counter	CSSRuleList	DocumentCSS
CSSCharsetRule	CSSStyleDeclaration	DOMImplementationCSS
CSSFontFaceRule	CSSStyleRule	ElementCSSInlineStyle
CSSImportRule	CSSStyleSheet	Rect
CSSMediaRule	CSSUnknownRule	RGBColor
CSSPrimitiveValue	CSSValue	ViewCSS
CSSRule	CSSValueList	

A.52 org.w3c.dom.events

DocumentEvent	EventListener	MutationEvent
Event	EventTarget	UIEvent
EventException	MouseEvent	

A.53 org.w3c.dom.html2

HTMLAnchorElement	HTMLFormElement	HTMLOptionsCollection
HTMLBodyElement	HTMLImageElement	HTMLSelectElement
HTMLCollection	HTMLInputElement	HTMLTextAreaElement
HTMLDocument	HTMLObjectElement	
HTMLElement	HTMLOptionElement	

A.54 org.w3c.dom.stylesheets

DocumentStyle	MediaList	StyleSheetList
LinkStyle	StyleSheet	

A.55 org.w3c.dom.views

AbstractView	DocumentView	

ANNEX B. JAVA CONSTANTS

The entirety of this annex is normative.

This annex specifies the values of certain public static final (constant) fields of Java types for which a value is not otherwise specified in the referenced document in which the type is defined and not specified above. A procedural application environment shall use the values defined below.

Table 11 Java Constants

Package	Class or Interface	Field Name	Value
java.awt	Adjustable	HORIZONTAL	0
java.awt	Adjustable	VERTICAL	1
java.awt	AWTEvent	ACTION_EVENT_MASK	128
java.awt	AWTEvent	ADJUSTMENT_EVENT_MASK	256
java.awt	AWTEvent	COMPONENT_EVENT_MASK	1
java.awt	AWTEvent	CONTAINER_EVENT_MASK	2
java.awt	AWTEvent	FOCUS_EVENT_MASK	4
java.awt	AWTEvent	ITEM_EVENT_MASK	512
java.awt	AWTEvent	KEY_EVENT_MASK	8
java.awt	AWTEvent	MOUSE_EVENT_MASK	16
java.awt	AWTEvent	MOUSE_MOTION_EVENT_MASK	32
java.awt	AWTEvent	RESERVED_ID_MAX	1999
java.awt	AWTEvent	TEXT_EVENT_MASK	1024
java.awt	AWTEvent	WINDOW_EVENT_MASK	64
java.awt	BorderLayout	CENTER	"Center"
java.awt	BorderLayout	EAST	"East"
java.awt	BorderLayout	NORTH	"North"
java.awt	BorderLayout	SOUTH	"South"
java.awt	BorderLayout	WEST	"West"
java.awt	Component	BOTTOM_ALIGNMENT	1.0
java.awt	Component	CENTER_ALIGNMENT	0.5
java.awt	Component	LEFT_ALIGNMENT	0.0
java.awt	Component	RIGHT_ALIGNMENT	1.0
java.awt	Component	TOP_ALIGNMENT	0.0
java.awt	Cursor	CROSSHAIR_CURSOR	1
java.awt	Cursor	DEFAULT_CURSOR	0
java.awt	Cursor	E_RESIZE_CURSOR	11
java.awt	Cursor	HAND_CURSOR	12
java.awt	Cursor	MOVE_CURSOR	13
java.awt	Cursor	N_RESIZE_CURSOR	8
java.awt	Cursor	NE_RESIZE_CURSOR	7
java.awt	Cursor	NW_RESIZE_CURSOR	6
java.awt	Cursor	S_RESIZE_CURSOR	9
java.awt	Cursor	SE_RESIZE_CURSOR	5
java.awt	Cursor	SW_RESIZE_CURSOR	4
java.awt	Cursor	TEXT_CURSOR	2
java.awt	Cursor	W_RESIZE_CURSOR	10
java.awt	Cursor	WAIT_CURSOR	3
java.awt	FlowLayout	CENTER	1
java.awt	FlowLayout	LEFT	0
java.awt	FlowLayout	RIGHT	2
java.awt	Font	BOLD	1
java.awt	Font	ITALIC	2
java.awt	Font	PLAIN	0
java.awt	GridBagConstraints	BOTH	1
java.awt	GridBagConstraints	CENTER	10
java.awt	GridBagConstraints	EAST	13

Package	Class or Interface	Field Name	Value
java.awt	GridBagConstraints	HORIZONTAL	2
java.awt	GridBagConstraints	NONE	0
java.awt	GridBagConstraints	NORTH	11
java.awt	GridBagConstraints	NORTHEAST	12
java.awt	GridBagConstraints	NORTHWEST	18
java.awt	GridBagConstraints	RELATIVE	-1
java.awt	GridBagConstraints	REMAINDER	0
java.awt	GridBagConstraints	SOUTH	15
java.awt	GridBagConstraints	SOUTHEAST	14
java.awt	GridBagConstraints	SOUTHWEST	16
java.awt	GridBagConstraints	VERTICAL	3
java.awt	GridBagConstraints	WEST	17
java.awt	Image	SCALE_AREA_AVERAGING	16
java.awt	Image	SCALE_DEFAULT	1
java.awt	Image	SCALE_FAST	2
java.awt	Image	SCALE_REPLICATE	8
java.awt	Image	SCALE_SMOOTH	4
java.awt	MediaTracker	ABORTED	2
java.awt	MediaTracker	COMPLETE	8
java.awt	MediaTracker	ERRORED	4
java.awt	MediaTracker	LOADING	1
java.awt.event	ActionEvent	ACTION_FIRST	1001
java.awt.event	ActionEvent	ACTION_LAST	1001
java.awt.event	ActionEvent	ACTION_PERFORMED	1001
java.awt.event	ActionEvent	ALT_MASK	8
java.awt.event	ActionEvent	CTRL_MASK	2
java.awt.event	ActionEvent	META_MASK	4
java.awt.event	ActionEvent	SHIFT_MASK	1
java.awt.event	AdjustmentEvent	ADJUSTMENT_FIRST	601
java.awt.event	AdjustmentEvent	ADJUSTMENT_LAST	601
java.awt.event	AdjustmentEvent	ADJUSTMENT_VALUE_CHANGED	601
java.awt.event	AdjustmentEvent	BLOCK_DECREMENT	3
java.awt.event	AdjustmentEvent	BLOCK_INCREMENT	4
java.awt.event	AdjustmentEvent	TRACK	5
java.awt.event	AdjustmentEvent	UNIT_DECREMENT	2
java.awt.event	AdjustmentEvent	UNIT_INCREMENT	1
java.awt.event	ComponentEvent	COMPONENT_FIRST	100
java.awt.event	ComponentEvent	COMPONENT_HIDDEN	103
java.awt.event	ComponentEvent	COMPONENT_LAST	103
java.awt.event	ComponentEvent	COMPONENT_MOVED	100
java.awt.event	ComponentEvent	COMPONENT_RESIZED	101
java.awt.event	ComponentEvent	COMPONENT_SHOWN	102
java.awt.event	ContainerEvent	COMPONENT_ADDED	300
java.awt.event	ContainerEvent	COMPONENT_REMOVED	301
java.awt.event	ContainerEvent	CONTAINER_FIRST	300
java.awt.event	ContainerEvent	CONTAINER_LAST	301
java.awt.event	FocusEvent	FOCUS_FIRST	1004
java.awt.event	FocusEvent	FOCUS_GAINED	1004
java.awt.event	FocusEvent	FOCUS_LAST	1005
java.awt.event	FocusEvent	FOCUS_LOST	1005
java.awt.event	InputEvent	ALT_MASK	8
java.awt.event	InputEvent	BUTTON1_MASK	16
java.awt.event	InputEvent	BUTTON2_MASK	8
java.awt.event	InputEvent	BUTTON3_MASK	4
java.awt.event	InputEvent	CTRL_MASK	2
java.awt.event	InputEvent	META_MASK	4
java.awt.event	InputEvent	SHIFT_MASK	1
java.awt.event	ItemEvent	DESELECTED	2
java.awt.event	ItemEvent	ITEM_FIRST	701
java.awt.event	ItemEvent	ITEM_LAST	701

Package	Class or Interface	Field Name	Value
java.awt.event	ItemEvent	ITEM_STATE_CHANGED	701
java.awt.event	ItemEvent	SELECTED	1
java.awt.event	KeyEvent	CHAR_UNDEFINED	0
java.awt.event	KeyEvent	KEY_FIRST	400
java.awt.event	KeyEvent	KEY_LAST	402
java.awt.event	KeyEvent	KEY_PRESSED	401
java.awt.event	KeyEvent	KEY_RELEASED	402
java.awt.event	KeyEvent	KEY_TYPED	400
java.awt.event	KeyEvent	VK_0	48
java.awt.event	KeyEvent	VK_1	49
java.awt.event	KeyEvent	VK_2	50
java.awt.event	KeyEvent	VK_3	51
java.awt.event	KeyEvent	VK_4	52
java.awt.event	KeyEvent	VK_5	53
java.awt.event	KeyEvent	VK_6	54
java.awt.event	KeyEvent	VK_7	55
java.awt.event	KeyEvent	VK_8	56
java.awt.event	KeyEvent	VK_9	57
java.awt.event	KeyEvent	VK_A	65
java.awt.event	KeyEvent	VK_ACCEPT	30
java.awt.event	KeyEvent	VK_ADD	107
java.awt.event	KeyEvent	VK_ALT	18
java.awt.event	KeyEvent	VK_B	66
java.awt.event	KeyEvent	VK_BACK_QUOTE	192
java.awt.event	KeyEvent	VK_BACK_SLASH	92
java.awt.event	KeyEvent	VK_BACK_SPACE	8
java.awt.event	KeyEvent	VK_C	67
java.awt.event	KeyEvent	VK_CANCEL	3
java.awt.event	KeyEvent	VK_CAPS_LOCK	20
java.awt.event	KeyEvent	VK_CLEAR	12
java.awt.event	KeyEvent	VK_CLOSE_BRACKET	93
java.awt.event	KeyEvent	VK_COMMA	44
java.awt.event	KeyEvent	VK_CONTROL	17
java.awt.event	KeyEvent	VK_CONVERT	28
java.awt.event	KeyEvent	VK_D	68
java.awt.event	KeyEvent	VK_DECIMAL	110
java.awt.event	KeyEvent	VK_DELETE	127
java.awt.event	KeyEvent	VK_DIVIDE	111
java.awt.event	KeyEvent	VK_DOWN	40
java.awt.event	KeyEvent	VK_E	69
java.awt.event	KeyEvent	VK_END	35
java.awt.event	KeyEvent	VK_ENTER	10
java.awt.event	KeyEvent	VK_EQUALS	61
java.awt.event	KeyEvent	VK_ESCAPE	27
java.awt.event	KeyEvent	VK_F	70
java.awt.event	KeyEvent	VK_F1	112
java.awt.event	KeyEvent	VK_F10	121
java.awt.event	KeyEvent	VK_F11	122
java.awt.event	KeyEvent	VK_F12	123
java.awt.event	KeyEvent	VK_F2	113
java.awt.event	KeyEvent	VK_F3	114
java.awt.event	KeyEvent	VK_F4	115
java.awt.event	KeyEvent	VK_F5	116
java.awt.event	KeyEvent	VK_F6	117
java.awt.event	KeyEvent	VK_F7	118
java.awt.event	KeyEvent	VK_F8	119
java.awt.event	KeyEvent	VK_F9	120
java.awt.event	KeyEvent	VK_FINAL	24
java.awt.event	KeyEvent	VK_G	71
java.awt.event	KeyEvent	VK_H	72

Package	Class or Interface	Field Name	Value
java.awt.event	KeyEvent	VK_HELP	156
java.awt.event	KeyEvent	VK_HOME	36
java.awt.event	KeyEvent	VK_I	73
java.awt.event	KeyEvent	VK_INSERT	155
java.awt.event	KeyEvent	VK_J	74
java.awt.event	KeyEvent	VK_K	75
java.awt.event	KeyEvent	VK_KANA	21
java.awt.event	KeyEvent	VK_KANJI	25
java.awt.event	KeyEvent	VK_L	76
java.awt.event	KeyEvent	VK_LEFT	37
java.awt.event	KeyEvent	VK_M	77
java.awt.event	KeyEvent	VK_META	157
java.awt.event	KeyEvent	VK_MODECHANGE	31
java.awt.event	KeyEvent	VK_MULTIPLY	106
java.awt.event	KeyEvent	VK_N	78
java.awt.event	KeyEvent	VK_NONCONVERT	29
java.awt.event	KeyEvent	VK_NUM_LOCK	144
java.awt.event	KeyEvent	VK_NUMPAD0	96
java.awt.event	KeyEvent	VK_NUMPAD1	97
java.awt.event	KeyEvent	VK_NUMPAD2	98
java.awt.event	KeyEvent	VK_NUMPAD3	99
java.awt.event	KeyEvent	VK_NUMPAD4	100
java.awt.event	KeyEvent	VK_NUMPAD5	101
java.awt.event	KeyEvent	VK_NUMPAD6	102
java.awt.event	KeyEvent	VK_NUMPAD7	103
java.awt.event	KeyEvent	VK_NUMPAD8	104
java.awt.event	KeyEvent	VK_NUMPAD9	105
java.awt.event	KeyEvent	VK_O	79
java.awt.event	KeyEvent	VK_OPEN_BRACKET	91
java.awt.event	KeyEvent	VK_P	80
java.awt.event	KeyEvent	VK_PAGE_DOWN	34
java.awt.event	KeyEvent	VK_PAGE_UP	33
java.awt.event	KeyEvent	VK_PAUSE	19
java.awt.event	KeyEvent	VK_PERIOD	46
java.awt.event	KeyEvent	VK_PRINTSCREEN	154
java.awt.event	KeyEvent	VK_Q	81
java.awt.event	KeyEvent	VK_QUOTE	222
java.awt.event	KeyEvent	VK_R	82
java.awt.event	KeyEvent	VK_RIGHT	39
java.awt.event	KeyEvent	VK_S	83
java.awt.event	KeyEvent	VK_SCROLL_LOCK	145
java.awt.event	KeyEvent	VK_SEMICOLON	59
java.awt.event	KeyEvent	VK_SEPARATER	108
java.awt.event	KeyEvent	VK_SHIFT	16
java.awt.event	KeyEvent	VK_SLASH	47
java.awt.event	KeyEvent	VK_SPACE	32
java.awt.event	KeyEvent	VK_SUBTRACT	109
java.awt.event	KeyEvent	VK_T	84
java.awt.event	KeyEvent	VK_TAB	9
java.awt.event	KeyEvent	VK_U	85
java.awt.event	KeyEvent	VK_UNDEFINED	0
java.awt.event	KeyEvent	VK_UP	38
java.awt.event	KeyEvent	VK_V	86
java.awt.event	KeyEvent	VK_W	87
java.awt.event	KeyEvent	VK_X	88
java.awt.event	KeyEvent	VK_Y	89
java.awt.event	KeyEvent	VK_Z	90
java.awt.event	MouseEvent	MOUSE_CLICKED	500
java.awt.event	MouseEvent	MOUSE_DRAGGED	506
java.awt.event	MouseEvent	MOUSE_ENTERED	504

Package	Class or Interface	Field Name	Value
java.awt.event	MouseEvent	MOUSE_EXITED	505
java.awt.event	MouseEvent	MOUSE_FIRST	500
java.awt.event	MouseEvent	MOUSE_LAST	506
java.awt.event	MouseEvent	MOUSE_MOVED	503
java.awt.event	MouseEvent	MOUSE_PRESSED	501
java.awt.event	MouseEvent	MOUSE_RELEASED	502
java.awt.event	PaintEvent	PAINT	800
java.awt.event	PaintEvent	PAINT_FIRST	800
java.awt.event	PaintEvent	PAINT_LAST	801
java.awt.event	PaintEvent	UPDATE	801
java.awt.event	TextEvent	TEXT_FIRST	900
java.awt.event	TextEvent	TEXT_LAST	900
java.awt.event	TextEvent	TEXT_VALUE_CHANGED	900
java.awt.event	WindowEvent	WINDOW_ACTIVATED	205
java.awt.event	WindowEvent	WINDOW_CLOSED	202
java.awt.event	WindowEvent	WINDOW_CLOSING	201
java.awt.event	WindowEvent	WINDOW_DEACTIVATED	206
java.awt.event	WindowEvent	WINDOW_DEICONIFIED	204
java.awt.event	WindowEvent	WINDOW_FIRST	200
java.awt.event	WindowEvent	WINDOW_ICONIFIED	203
java.awt.event	WindowEvent	WINDOW_LAST	206
java.awt.event	WindowEvent	WINDOW_OPENED	200
java.awt.image	ImageConsumer	COMPLETESCANLINES	4
java.awt.image	ImageConsumer	IMAGEABORTED	4
java.awt.image	ImageConsumer	IMAGEERROR	1
java.awt.image	ImageConsumer	RANDOMPIXELORDER	1
java.awt.image	ImageConsumer	SINGLEFRAME	16
java.awt.image	ImageConsumer	SINGLEFRAMEDONE	2
java.awt.image	ImageConsumer	SINGLEPASS	8
java.awt.image	ImageConsumer	STATICIMAGEDONE	3
java.awt.image	ImageConsumer	TOPDOWNLEFTRIGHT	2
java.awt.image	ImageObserver	ABORT	128
java.awt.image	ImageObserver	ALLBITS	32
java.awt.image	ImageObserver	ERROR	64
java.awt.image	ImageObserver	FRAMEBITS	16
java.awt.image	ImageObserver	HEIGHT	2
java.awt.image	ImageObserver	PROPERTIES	4
java.awt.image	ImageObserver	SOMEBITS	8
java.awt.image	ImageObserver	WIDTH	1
java.io	PipedInputStream	PIPE_SIZE	1024
java.io	StreamTokenizer	TT_EOF	-1
java.io	StreamTokenizer	TT_EOL	10
java.io	StreamTokenizer	TT_NUMBER	-2
java.io	StreamTokenizer	TT_WORD	-3
java.lang	Byte	MAX_VALUE	127
java.lang	Byte	MIN_VALUE	-128
java.lang	Character	COMBINING_SPACING_MARK	8
java.lang	Character	CONNECTOR_PUNCTUATION	23
java.lang	Character	CONTROL	15
java.lang	Character	CURRENCY_SYMBOL	26
java.lang	Character	DASH_PUNCTUATION	20
java.lang	Character	DECIMAL_DIGIT_NUMBER	9
java.lang	Character	ENCLOSING_MARK	7
java.lang	Character	END_PUNCTUATION	22
java.lang	Character	FORMAT	16
java.lang	Character	LETTER_NUMBER	10
java.lang	Character	LINE_SEPARATOR	13
java.lang	Character	LOWERCASE_LETTER	2
java.lang	Character	MATH_SYMBOL	25
java.lang	Character	MAX_RADIX	36

Package	Class or Interface	Field Name	Value
java.lang	Character	MAX_VALUE	65535
java.lang	Character	MIN_RADIX	2
java.lang	Character	MIN_VALUE	0
java.lang	Character	MODIFIER_LETTER	4
java.lang	Character	MODIFIER_SYMBOL	27
java.lang	Character	NON_SPACING_MARK	6
java.lang	Character	OTHER_LETTER	5
java.lang	Character	OTHER_NUMBER	11
java.lang	Character	OTHER_PUNCTUATION	24
java.lang	Character	OTHER_SYMBOL	28
java.lang	Character	PARAGRAPH_SEPARATOR	14
java.lang	Character	PRIVATE_USE	18
java.lang	Character	SPACE_SEPARATOR	12
java.lang	Character	START_PUNCTUATION	21
java.lang	Character	SURROGATE	19
java.lang	Character	TITLECASE_LETTER	3
java.lang	Character	UNASSIGNED	0
java.lang	Character	UPPERCASE_LETTER	1
java.lang	Double	MAX_VALUE	1.7976931348623157E308
java.lang	Double	MIN_VALUE	4.9406564584124654E-324
java.lang	Double	NaN	NaN
java.lang	Double	NEGATIVE_INFINITY	-Infinity
java.lang	Double	POSITIVE_INFINITY	Infinity
java.lang	Float	MAX_VALUE	3.4028235E38
java.lang	Float	MIN_VALUE	1.40129846E-45
java.lang	Float	NaN	NaN
java.lang	Float	NEGATIVE_INFINITY	-Infinity
java.lang	Float	POSITIVE_INFINITY	Infinity
java.lang	Integer	MAX_VALUE	2147483647
java.lang	Integer	MIN_VALUE	-2147483648
java.lang	Long	MAX_VALUE	9223372036854775807
java.lang	Long	MIN_VALUE	-9223372036854775808
java.lang	Math	E	2.718281828459045
java.lang	Math	PI	3.141592653589793
java.lang	Short	MAX_VALUE	32767
java.lang	Short	MIN_VALUE	-32768
java.lang	Thread	MAX_PRIORITY	10
java.lang	Thread	MIN_PRIORITY	1
java.lang	Thread	NORM_PRIORITY	5
java.lang.reflect	Member	DECLARED	1
java.lang.reflect	Member	PUBLIC	0
java.lang.reflect	Modifier	ABSTRACT	1024
java.lang.reflect	Modifier	FINAL	16
java.lang.reflect	Modifier	INTERFACE	512
java.lang.reflect	Modifier	NATIVE	256
java.lang.reflect	Modifier	PRIVATE	2
java.lang.reflect	Modifier	PROTECTED	4
java.lang.reflect	Modifier	PUBLIC	1
java.lang.reflect	Modifier	STATIC	8
java.lang.reflect	Modifier	SYNCHRONIZED	32
java.lang.reflect	Modifier	TRANSIENT	128
java.lang.reflect	Modifier	VOLATILE	64
java.text	DateFormat	AM_PM_FIELD	14
java.text	DateFormat	DATE_FIELD	3
java.text	DateFormat	DAY_OF_WEEK_FIELD	9
java.text	DateFormat	DAY_OF_WEEK_IN_MONTH_FIELD	11
java.text	DateFormat	DAY_OF_YEAR_FIELD	10
java.text	DateFormat	DEFAULT	2
java.text	DateFormat	ERA_FIELD	0
java.text	DateFormat	FULL	0

Package	Class or Interface	Field Name	Value
java.text	DateFormat	HOUR_OF_DAY0_FIELD	5
java.text	DateFormat	HOUR_OF_DAY1_FIELD	4
java.text	DateFormat	HOUR0_FIELD	16
java.text	DateFormat	HOUR1_FIELD	15
java.text	DateFormat	LONG	1
java.text	DateFormat	MEDIUM	2
java.text	DateFormat	MILLISECOND_FIELD	8
java.text	DateFormat	MINUTE_FIELD	6
java.text	DateFormat	MONTH_FIELD	2
java.text	DateFormat	SECOND_FIELD	7
java.text	DateFormat	SHORT	3
java.text	DateFormat	TIMEZONE_FIELD	17
java.text	DateFormat	WEEK_OF_MONTH_FIELD	13
java.text	DateFormat	WEEK_OF_YEAR_FIELD	12
java.text	DateFormat	YEAR_FIELD	1
java.text	NumberFormat	FRACTION_FIELD	1
java.text	NumberFormat	INTEGER_FIELD	0
java.util	Calendar	AM	0
java.util	Calendar	AM_PM	9
java.util	Calendar	APRIL	3
java.util	Calendar	AUGUST	7
java.util	Calendar	DATE	5
java.util	Calendar	DAY_OF_MONTH	5
java.util	Calendar	DAY_OF_WEEK	7
java.util	Calendar	DAY_OF_WEEK_IN_MONTH	8
java.util	Calendar	DAY_OF_YEAR	6
java.util	Calendar	DECEMBER	11
java.util	Calendar	DST_OFFSET	16
java.util	Calendar	ERA	0
java.util	Calendar	FEBRUARY	1
java.util	Calendar	FIELD_COUNT	17
java.util	Calendar	FRIDAY	6
java.util	Calendar	HOUR	10
java.util	Calendar	HOUR_OF_DAY	11
java.util	Calendar	JANUARY	0
java.util	Calendar	JULY	6
java.util	Calendar	JUNE	5
java.util	Calendar	MARCH	2
java.util	Calendar	MAY	4
java.util	Calendar	MILLISECOND	14
java.util	Calendar	MINUTE	12
java.util	Calendar	MONDAY	2
java.util	Calendar	MONTH	2
java.util	Calendar	NOVEMBER	10
java.util	Calendar	OCTOBER	9
java.util	Calendar	PM	1
java.util	Calendar	SATURDAY	7
java.util	Calendar	SECOND	13
java.util	Calendar	SEPTEMBER	8
java.util	Calendar	SUNDAY	1
java.util	Calendar	THURSDAY	5
java.util	Calendar	TUESDAY	3
java.util	Calendar	UNDECIMBER	12
java.util	Calendar	WEDNESDAY	4
java.util	Calendar	WEEK_OF_MONTH	4
java.util	Calendar	WEEK_OF_YEAR	3
java.util	Calendar	YEAR	1
java.util	Calendar	ZONE_OFFSET	15
java.util	GregorianCalendar	AD	1
java.util	GregorianCalendar	BC	0

Package	Class or Interface	Field Name	Value
java.util.zip	ZipEntry	DEFLATED	8
java.util.zip	ZipEntry	STORED	0
javax.media	CachingControl	LENGTH_UNKNOWN	java.lang.Long.MAX_VALUE
javax.media	Controller	Prefetched	500
javax.media	Controller	Prefetching	400
javax.media	Controller	Realized	300
javax.media	Controller	Realizing	200
javax.media	Controller	Started	600
javax.media	Controller	Unrealized	100
javax.media	Time	ONE_SECOND	1000000000
javax.media.protocol	Positionable	RoundDown	2
javax.media.protocol	Positionable	RoundNearest	3
javax.media.protocol	Positionable	RoundUp	1
javax.media.protocol	SourceStream	LENGTH_UNKNOWN	-1

ANNEX C. JAVA SYSTEM PROPERTIES

The entirety of this annex is normative.

This annex specifies the values of certain Java system properties which are accessible by means of `System.getProperty(java.lang.String)` and related methods. A procedural application environment shall use the values defined below except where marked *implementation specific* in which case an implementation specified value shall be used. Access to read these properties shall not be denied to any application.

Other than the properties listed below, system properties that begin with `"dase"` or `"org.atsc"` are reserved for future use and shall not be used by an application entity or procedural application environment.

Table 12 Java System Properties

Property Name	Value
"dase.delivery.system"	"ARM"
"dase.implementation.name"	*implementation specific*
"dase.implementation.vendor"	*implementation specific*
"dase.implementation.version"	*implementation specific*
"dase.implementation.level"	"1"
"dase.specification.name"	"DASE-1"
"dase.specification.vendor"	"ATSC"
"dase.specification.version"	"1.0"
"file.separator"	*implementation specific*
"line.separator"	*implementation specific*
"path.separator"	*implementation specific*
"user.language"	*user specified language identifier*
"user.region"	*user specified region identifier*
"user.timezone"	*user specified timezone identifier*

Notes

1. It is recommended that the value of the `"file.separator"` system property be `"/"`.
2. It is recommended that the value of the `"line.separator"` system property be `"\n"`.
3. It is recommended that the value of the `"path.separator"` system property be `";"`.
4. The value of the `"user.language"` system property shall either be an empty string or be consistent with the syntax of the language identifier as returned by `java.util.Locale.getLanguage()`. The actual value should be specified or specifiable by the end-user in an implementation specific fashion.
5. The value of the `"user.region"` system property shall either be an empty string or be consistent with the syntax of a string composed of a country identifier as returned by `java.util.Locale.getCountry()` followed optionally by `'_'` (underscore) and a variant identifier as returned by `java.util.Locale.getVariant()`. The actual value should be specified or specifiable by the end-user in an implementation specific fashion.
6. The value of the `"user.timezone"` system property shall be consistent with the syntax of the timezone identifier as returned by `java.util.TimeZone.getID()`. The actual value should be specified or specifiable by the end-user in an implementation specific fashion.

ANNEX D. XLET CONTEXT PROPERTIES

The entirety of this annex is normative.

This annex specifies all standard, pre-defined properties which are accessible by means of `javax.tv.xlet.XletContext.getXletProperty()`.

Other than the properties listed below, Xlet context properties which begin with `"dase"` or `"org.atsc"` are reserved for future use and shall not be used by an application entity or procedural application environment.

Table 13 Xlet Context Properties

Property Key	Value Type
"javax.tv.xlet.args"	java.lang.String[]
"org.atsc.trigger.source.default"	org.atsc.trigger.TriggerSource
"org.atsc.util.locales"	java.util.Locale[]
"org.atsc.xlet.obj.codebase"	java.lang.String or null
"org.atsc.xlet.obj.data"	java.lang.String or null
"org.atsc.xlet.obj.type"	java.lang.String or null

The determination of the values of these properties is described in the following sub-sections.

D.1 javax.tv.xlet.args

The value of the Xlet context property `javax.tv.xlet.args` shall be a `String` array whose entries are determined as follows:

(1) for a *primary* Xlet, use the values of the `arg.`*`n`* application parameters specified by the application's metadata resource in accordance with [DASE], Section 6.1.1.13.1;

(2) for a *secondary* Xlet, use the value of the *args* parameter provided to the `org.atsc.xlet.XletRegistry.startXlet(…)` method;

(3) for an *embedded* Xlet, use the values of the `arg.`*`n`* parameters specified by *param* element children of the *object* element in accordance with *DASE-1 Part 2: Declarative Applications and Environment*, Section 5.1.1.6.8.

Note: See Section 3.3 for definitions of *primary*, *secondary*, and *embedded* Xlets.

D.2 org.atsc.trigger.source.default

The value of the Xlet context property `org.atsc.trigger.source.default` shall be determined as follows:

(1) for each Xlet, create and use an instance of `org.atsc.trigger.TriggerSource`.

Note: The use of distinct instances of `TriggerSource` for distinct Xlets is not intended to imply the presence of multiple broadcast trigger sources; that is, a single broadcast trigger source whose target is a single DASE application results in dispatching each `TriggerEvent` to each `TriggerListener` registered with each `TriggerSource` instance associated with the single broadcast trigger source.

D.3 org.atsc.util.locales

The value of the Xlet context property `org.atsc.util.locales` shall be an array of `java.util.Locale` objects where each object is a cloned copy of the `java.util.Locale` object representing a supported locale.

Note: The value of this Xlet context property is equivalent to that returned by the `java.util.Locale.getAvailableLocales()` method defined by [JDK1.2.2] but not supported by [PJAE].

D.4 org.atsc.xlet.obj.codebase

The value of the Xlet context property `org.atsc.xlet.obj.codebase` shall be determined as follows:

(1) for an embedded Xlet, use the absolutized form of the URI value of the *object* element's *codebase* attribute, if specified, or an empty string, if not specified;
(2) otherwise, use the value *null*.

D.5 org.atsc.xlet.obj.data

The value of the Xlet context property `org.atsc.xlet.obj.data` shall be determined as follows:

(1) for an embedded Xlet, use the absolutized form of the URI value of the *object* element's *data* attribute, if specified, or an empty string, if not specified;
(2) otherwise, use the value *null*.

D.6 org.atsc.xlet.obj.type

The value of the Xlet context property `org.atsc.xlet.obj.type` shall be determined as follows:

(1) for an embedded Xlet, use the value of the *object* element's *type* attribute, if specified, or an empty string, if not specified;
(2) otherwise, use the value *null*;

ANNEX E. INTERNATIONAL RESOURCES

The entirety of this annex and its subsections is normative.

This annex specifies certain resources required to support Java localization and internationalization features.

E.1 External Character Encodings

A procedural application environment shall support the following external character encoding systems. These systems shall be supported by all Java methods which take a character encoding system name as an argument; e.g., `String(byte[],int,int,String)` and `String.getBytes(String)`. A procedural application environment should use [UTF-8] as the default platform character encoding system.

Table 14 External Character Encodings

Character Encoding Name	Description
"UTF-8"	UCS Transformation Format, 8-bit Form
"ISO-8859-1"	ISO Latin 1

E.2 Built-In Locales

A procedural application environment shall support all necessary resources required to implement the semantics of the following locales:

Table 15 Built-In Locales

Locale Name	Predefined Locale Field	Description
"en"	Locale.ENGLISH	English, Generic
"en_US"	Locale.US	English, United States
"en_CA"	Locale.CANADA	English, Canada
"fr"	Locale.FRENCH	French, Generic
"fr_CA"	Locale.CANADA_FRENCH	Fench, Canada
"es"	*none*	Spanish, Generic
"es_MX"	*none*	Spanish, Mexico

CHANGES

The entirety of 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 16 Changes from Candidate Standard

Section	Description
1	Change status to standard.
2	Add [DASE-ZIP] normative reference.
2	Change [JAR-MANIFEST] to [JAR]; update name and hyperlink.
2	Add [JDK1.1.8] and [JDK1.2.2] normative references.
2	Add [MIME-MEDIA] normative reference.
2	Add [UTF-8] normative reference; update [UNICODE] to reference Unicode Version 3.2.
3	Add JDK acronym.
4.5	Clarify relative file name resolution for application delivery file systems.
4.5	Clarify role and value of class file name extension.
5.1	Clarify requirements for default constructors.
5.1	Correct reference to ClassNotFoundException to read NoClassDefFoundError.
5.1	Correct references to No{Method,Field}FoundException to read No{Method,Field}FoundError.
5.1	Recommend use of java.lang.UnsupportedOperationException.
5.1.1.2.1	Don't require java.awt.AWTEventMulticaster.save(ObjectOutputStream,String)
5.1.1.2.1	Don't require java.awt.AWTEventMulticaster.saveInternal(ObjectOutputStream,String)
5.1.1.2.1	Specify values of java.awt.Color constants.
5.1.1.2.1	Don't require java.awt.GridBagLayoutInfo.
5.1.1.2.1	Don't require java.awt.MenuContainer.
5.1.1.2.1	Don't require java.awt.Component to implement MenuContainer, but do require getFont().
5.1.1.2.1	Don't require java.awt.Component.list().
5.1.1.2.1	Don't require java.awt.Component.list(PrintStream).
5.1.1.2.1	Don't require java.awt.Component.list(PrintStream,int).
5.1.1.2.1	Don't require java.awt.Component.list(PrintWriter).
5.1.1.2.1	Don't require java.awt.Component.list(PrintWriter,int).
5.1.1.2.1	Don't require java.awt.Component.paramString().
5.1.1.2.1	Recommend UnsupportedOperationException if setCursor() semantics aren't supported.
5.1.1.2.1	Don't require java.awt.Container.add(String,Component).
5.1.1.2.1	Don't require java.awt.Container.addImpl(Component,Object,int).
5.1.1.2.1	Don't require java.awt.Container.list(PrintStream,int).
5.1.1.2.1	Don't require java.awt.Container.list(PrintWriter,int).
5.1.1.2.1	Don't require java.awt.Container.printComponents(Graphics).
5.1.1.2.1	Recommend java.awt.Dimension.hashCode().
5.1.1.2.1	Clarify semantics of java.awt.FontMetrics.bytesWidth(byte[],int,int).
5.1.1.2.1	Clarify semantics of java.awt.Graphics.drawBytes(byte[],int,int,int,int).
5.1.1.2.1	Don't require java.awt.GridBagLayout.comptable.
5.1.1.2.1	Don't require java.awt.GridBagLayout.defaultConstraints.
5.1.1.2.1	Don't require java.awt.GridBagLayout.layoutInfo.
5.1.1.2.1	Don't require java.awt.GridBagLayout.MAXGRIDSIZE.
5.1.1.2.1	Don't require java.awt.GridBagLayout.MINSIZE.
5.1.1.2.1	Don't require java.awt.GridBagLayout.PREFERREDSIZE.
5.1.1.2.1	Don't require java.awt.GridBagLayout.AdjustForGravity(GridBagConstraints,Rectangle).
5.1.1.2.1	Don't require java.awt.GridBagLayout.ArrangeGrid(Container).
5.1.1.2.1	Don't require java.awt.GridBagLayout.GetLayoutInfo(Container,int).
5.1.1.2.1	Don't require java.awt.GridBagLayout.GetMinSize(Container,GridBagLayoutInfo).
5.1.1.2.1	Clarify semantics of java.awt.Image.getProperty(String); specify pre-defined image properties.
5.1.1.2.1	Recommend java.awt.Insets.hashCode().
5.1.1.2.1	Don't require java.awt.Toolkit.getNativeContainer(Component).

Section	Description
5.1.1.2.1	Don't require java.awt.Toolkit.getScreenResolution().
5.1.1.2.1	Don't require java.awt.Toolkit.getSystemEventQueue().
5.1.1.2.1	Don't require java.awt.Toolkit.getSystemEventQueueImpl().
5.1.1.2.1	Clarify semantics of java.awt.Toolkit.createImage(byte[]).
5.1.1.2.1	Clarify semantics of java.awt.Toolkit.getProperty(String,String).
5.1.1.2.1	Don't require java.awt.event.KeyEvent(Component,int,long,int,int).
5.1.1.2.1	Clarify semantics of java.awt.event.KeyEvent.isAction().
5.1.1.2.1	Don't require java.awt.image.ColorModel.finalize().
5.1.1.2.1	Clarify semantics of java.awt.image.CropImageFilter.setProperties(Hashtable).
5.1.1.2.1	Clarify override semantics of java.awt.image.DirectColorModel.getRGB().
5.1.1.2.1	Clarify semantics of java.awt.image. ImageFilter methods reserved for use by image producer.
5.1.1.2.1	Clarify semantics of java.awt.image. ImageFilter.setProperties(Hashtable).
5.1.1.2.1	Don't require java.awt.image.PixelGrabber.status().
5.1.1.2.1	Clarify semantics of java.awt.image. ReplicateScaleFilter.setProperties(Hashtable).
5.1.1.2.1	Don't require java.beans.BeanDescriptor.
5.1.1.2.1	Don't require java.beans.BeanInfo.
5.1.1.2.1	Don't require java.beans.Customizer.
5.1.1.2.1	Don't require java.beans.EventSetDescriptor.
5.1.1.2.1	Don't require java.beans.FeatureDescriptor.
5.1.1.2.1	Don't require java.beans.IndexedPropertyDescriptor.
5.1.1.2.1	Don't require java.beans.IntrospectionException.
5.1.1.2.1	Don't require java.beans.Introspector.
5.1.1.2.1	Don't require java.beans.MethodDescriptor.
5.1.1.2.1	Don't require java.beans.ParameterDescriptor.
5.1.1.2.1	Don't require java.beans.PropertyDescriptor.
5.1.1.2.1	Don't require java.beans.PropertyEditor.
5.1.1.2.1	Don't require java.beans.PropertyEditorManager.
5.1.1.2.1	Don't require java.beans.PropertyEditorSupport.
5.1.1.2.1	Don't require java.beans.SimpleBeanInfo
5.1.1.2.1	Don't require java.beans.Beans.getInstanceOf(Object,Class).
5.1.1.2.1	Don't require java.beans.Beans.isInstanceOf(Object,Class).
5.1.1.2.1	Don't require java.beans.Beans.setDesignTime(boolean).
5.1.1.2.1	Don't require java.beans.Beans.setGuiAvailable(boolean).
5.1.1.2.1	Constrain semantics of java.beans.Beans.isDesignTime().
5.1.1.2.1	Constrain semantics of java.beans.Beans.isGuiAvailable().
5.1.1.2.1	Require semantic support for java.io.* types related to object serialization.
5.1.1.2.1	Constrain semantics of java.io.BufferedInputStream.BufferedInputStream(InputStream).
5.1.1.2.1	Clarify semantics of java.io.DataInput with respect to JDK1.2.2.
5.1.1.2.1	Don't require java.io.DataInput.readLine().
5.1.1.2.1	Clarify semantics of java.io.DataOutput with respect to JDK1.2.2.
5.1.1.2.1	Clarify semantics of java.io.File with respect to object carousel directories.
5.1.1.2.1	Clarify semantics of java.io.FileInputStream with respect to JDK1.2.2.
5.1.1.2.1	Clarify semantics of java.io.FileOutputStream with respect to JDK1.2.2.
5.1.1.2.1	Require semantic support for java.io.ObjectInputStream.
5.1.1.2.1	Require semantic support for java.io.ObjectOutputStream.
5.1.1.2.1	Require semantic support for java.io.ObjectStreamClass.
5.1.1.2.1	Clarify semantics of java.io.ObjectInputStream with respect to JDK1.2.2.
5.1.1.2.1	Clarify semantics of java.io.ObjectOutputStream with respect to JDK1.2.2.
5.1.1.2.1	Clarify semantics of java.io.PipedInputStream constants.
5.1.1.2.1	Don't require java.io.RandomAccessFile.readLine().
5.1.1.2.1	Don't require java.lang.Compiler.
5.1.1.2.1	Don't require java.lang.Process.
5.1.1.2.1	Don't require java.lang.Class.getClasses().
5.1.1.2.1	Don't require java.lang.Class.getDeclaredClasses().
5.1.1.2.1	Clarify semantics of java.lang.Class with respect to JDK1.2.2.
5.1.1.2.1	Permit java.lang.ClassLoader.loadClass(String,boolean) to be concrete.
5.1.1.2.1	Clarify semantics of java.lang.ClassLoader.findLoadedClass(String) with respect to JDK1.2.2.
5.1.1.2.1	Constrain semantics of java.lang.Float constants.
5.1.1.2.1	Constrain semantics of java.lang.Double constants.
5.1.1.2.1	Clarify semantics of java.lang.Math with regard to non-strict floating point.

Section	Description
5.1.1.2.1	Clarify semantics of java.lang.Object.toString() with regard to form of return value.
5.1.1.2.1	Don't require java.lang.Runtime.traceInstructions().
5.1.1.2.1	Don't require java.lang.Runtime.traceMethodCalls().
5.1.1.2.1	Don't require java.lang.Runtime.exec(…).
5.1.1.2.1	Don't require java.lang.Runtime.exit(int).
5.1.1.2.1	Don't require java.lang.Runtime.load(String).
5.1.1.2.1	Don't require java.lang.Runtime.loadLibrary(String).
5.1.1.2.1	Don't require java.lang.SecurityManager.classDepth(String).
5.1.1.2.1	Don't require java.lang.SecurityManager.classLoaderDepth().
5.1.1.2.1	Don't require java.lang.SecurityManager.currentClassLoader().
5.1.1.2.1	Don't require java.lang.SecurityManager.currentLoadedClass().
5.1.1.2.1	Don't require java.lang.SecurityManager.getInCheck().
5.1.1.2.1	Don't require java.lang.SecurityManager.inClass(String).
5.1.1.2.1	Don't require java.lang.SecurityManager.inClassLoader().
5.1.1.2.1	Don't require java.lang.SecurityManager.inCheck.
5.1.1.2.1	Clarify semantics of java.lang.String.String(byte[],int,int).
5.1.1.2.1	Don't require java.lang.System.exit(int).
5.1.1.2.1	Don't require java.lang.System.load(String).
5.1.1.2.1	Don't require java.lang.System.loadLibrary(String).
5.1.1.2.1	Don't require java.lang.System.setIn(InputStream).
5.1.1.2.1	Don't require java.lang.System.setErr(PrintStream).
5.1.1.2.1	Don't require java.lang.System.setOut(PrintStream).
5.1.1.2.1	Require java.lang.UnsupportedOperationException.
5.1.1.2.1	Don't require java.net.MulticastSocket.getTTL().
5.1.1.2.1	Don't require java.net.MulticastSocket.send(DatagramPacket,byte).
5.1.1.2.1	Don't require java.net.MulticastSocket.setTTL(byte).
5.1.1.2.1	Don't require java.security.Key.serialVersionUID.
5.1.1.2.1	Clarify semantics of java.security.Provider with respect to JDK1.2.2.
5.1.1.2.1	Don't require java.security.PublicKey.serialVersionUID.
5.1.1.2.1	Don't require java.security.Security.getAlgorithmProperty(String,String).
5.1.1.2.1	Don't require java.text.CharacterIterator.
5.1.1.2.1	Don't require java.text.BreakIterator.
5.1.1.2.1	Don't require java.text.CollationElementIterator.
5.1.1.2.1	Don't require java.text.CollationKey.
5.1.1.2.1	Don't require java.text.Collator.
5.1.1.2.1	Don't require java.text.RuleBasedCollator.
5.1.1.2.1	Don't require java.text.StringCharacterIterator.
5.1.1.2.1	Clarify semantics of java.util.Calendar.after(Object) with regard to argument type.
5.1.1.2.1	Clarify semantics of java.util.Calendar.before(Object) with regard to argument type.
5.1.1.2.1	Recommend override of java.util.Calendar.hashCode().
5.1.1.2.1	Don't require java.util.Date.Date(String).
5.1.1.2.1	Recommend override of java.util.Date.clone().
5.1.1.2.1	Clarify semantics of java.util.Locale.getDefault().
5.1.1.2.1	Clarify semantics of java.util.Locale.setDefault(Locale) with regard to priviliged operation access.
5.1.1.2.1	Specify use of Xlet context property to obtain supported java.util.Locale instances.
5.1.1.2.1	Don't require java.util.Properties.save(OutputStream,String).
5.1.1.2.1	Don't require java.util.Properties.list(PrintStream).
5.1.1.2.1	Don't require java.util.Properties.list(PrintWriter).
5.1.1.2.1	Specify syntax of property list resource read by java.util.Properties.load(InputStream).
5.1.1.2.1	Constrain semantics of java.util.Properties.put(Object,Object).
5.1.1.2.1	Clarify semantics of java.util.PropertyResourceBundle.PropertyResourceBundle(InputStream).
5.1.1.2.1	Specify convention for determining locale represented by a ResourceBundle.
5.1.1.2.1	Specify syntax of time zone identifiers used by java.util.TimeZone.
5.1.1.2.1	Clarify semantics of java.util.TimeZone.getDefault().
5.1.1.2.1	Clarify semantics of java.util.TimeZone.setDefault(TimeZone) with regard to priviliged operation access.
5.1.1.2.1	Constrain semantics of java.util.zip.CRC32 with regard to algorithm.
5.1.1.2.1	Don't require java.util.zip.Inflater.Inflater().
5.1.1.2.1	Don't require java.util.zip.Inflater.getAdler().
5.1.1.2.1	Don't require java.util.zip.Inflater.setDictionary(byte[]).
5.1.1.2.1	Don't require java.util.zip.Inflater.setDictionary(byte[],int,int).

Section	Description
5.1.1.2.1	Clarify semantics of java.util.zip.Inflater with regard to ZLIB support.
5.1.1.2.1	Recommend override of java.util.zip.InflaterInputStream.available().
5.1.1.2.1	Recommend override of java.util.zip.InflaterInputStream.close().
5.1.1.2.1	Don't require com.sun.awt package.
5.1.1.2.1	Don't require com.sun.lang package.
5.1.1.2.1	Don't require java.util code signing optional group.
5.1.1.2.1	Don't require java.util.jar code signing optional group.
5.1.1.2.2	Clarify semantics of javax.media.protocol.DataSource.setMediaLocator(MediaLocator).
5.1.1.2.2	Constrain value of javax.media.Manager.UNKNOWN_CONTENT_NAME.
5.1.1.2.2	Clarify semantics of javax.media.Manager.getDataSourceList(String).
5.1.1.2.2	Clarify semantics of javax.media.Manager.getHandlerClassList(String).
5.1.1.2.2	Constrain semantics of javax.media.protocol.ContentDescriptor with regard to content type name.
5.1.1.2.2	Constrain value of javax.media.protcol.ContentDescriptor.CONTENT_UNKNOWN.
5.1.1.2.2	Constrain value of javax.media.Clock.RESET.
5.1.1.2.2	Constrain value of javax.media.Controller.LATENCY_UNKNOWN.
5.1.1.2.2	Constrain value of javax.media.Duration.DURATION_UNBOUNDED.
5.1.1.2.2	Constrain value of javax.media.Duration.DURATION_UNKNOWN.
5.1.1.2.2	Constrain return value of javax.media.protocol.DataSource.getContentType().
5.1.1.2.2	Don't require javax.media.protocol.DataSource.initCheck().
5.1.1.2.2	Require javax.media.protocol.ContentDescriptor.mimeTypeToPackageName(String) to be public.
5.1.1.2.2	Change process_em_data_flag field in DTVCC Frame Format to reserved.
5.1.1.2.2	Change em_data field in DTVCC Frame Format to reserved.
5.1.1.2.3	Unconstrain javax.tv.carousel.CarouselFile to support object carousel directories.
5.1.1.2.3	Unconstrain javax.tv.locator.LocatorFactory to support object carousel directories.
5.1.1.2.3	Don't require a runtime exception from javax.tv.media.protocol.PushSourceStream2.read(byte[],int,int).
5.1.1.2.3	Remove constraint on javax.tv.service.selection.ServiceContextFactory.createServiceContext().
5.1.1.2.4	Remove dependence on com.sun.awt package.
5.1.1.2.6	Change org.w3c.dom.html to org.w3c.dom.html2.
5.1.1.2.7	Reserve sole use of org.atsc namespace for ATSC standardization purposes.
5.3	Add text content facility.
5.3.1	Add text/plain content type.
5.4.1	Constrain use of content-type per-entry attribute.
5.4.1	Don't require semantic support for digital signature of archive.
A	Remove com.sun.awt package.
A	Remove com.sun.lang package.
A	Remove non-required types from java.beans.
A	Remove non-required Compiler and Process types from java.lang.
A	Add UnsupportedOperationException type to java.lang.
A	Remove non-required types from java.text.
A	Remove java.util code-signing optional group types.
A	Remove java.util.jar package.
A	Add org.atsc.dom.events package.
A	Add org.atsc.dom.events.KeyEvent.
A	Add org.atsc.dom.events.KeyModifiers.
A	Add org.atsc.dom.events.VirtualKeys.
A	Add org.atsc.dom.html.HTMLImageElementExt.
A	Add missing types to org.havi.ui.event: H{Item,Key,Text}{Event,Listener}.
A	Change org.w3c.dom.html to org.w3c.dom.html2.
A	Add org.w3c.dom.html2.{HTMLImageElement,HTMLOptionsCollection}.
A	Remove org.w3c.dom.html2.HTMLDOMImplementation due to removal from DOM2-HTML CR.
B	Add missing java.awt.GridBagConstraints constants.
B	Add missing java.lang.Double.MIN_VALUE constant.
B	Correct value of java.lang.Float.MIN_VALUE constant.
C	Remove javatv.* system properties.
C	Specify user.language system property.
C	Specify user.region system property.
C	Specify user.timezone system property.
D	Add org.atsc.util.locales Xlet context property.
E	Recommend use of UTF-8 as default character encoding system.
E	Correct case of locale names.