DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1)
PART 7: APPLICATION DELIVERY SYSTEM – ARM BINDING

ATSC Standard
Table of Tables

Table 1 Asynchronous Stream Encapsulation ................................................................. 8
Table 2 DASE Trigger Encapsulation .............................................................................. 9
Table 3 Implied Resource References ............................................................................. 9
Table 4 Data Application State Transition Mappings ...................................................... 10
Table 5 Other Application Delivery System Event Mappings ......................................... 10
Table 6 Application Lifecycle State Transition Mappings .............................................. 11
Table 7 Other Application Environment Event Mappings .............................................. 11
Table 8 Application Emission Policy Mappings .............................................................. 12
Table 9 Locator External Forms ..................................................................................... 13
Table 10 Autoload Scenarios .......................................................................................... 37
Table 11 Changes from Candidate Standard ................................................................. 38
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/100-7) dated November 5, 2002.

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

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

1.2 Purpose

This document specifies a binding between a DASE System and the ATSC Data Application Reference Model. This binding specifies a mapping by means of which DASE Applications may be delivered and controlled by the ATSC Data Broadcast Standard and related specifications.¹

1.3 Application

The behavior and facilities of this specification are intended to apply to terrestrial (over-the-air) broadcast systems and receivers. In addition, the same behavior 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 bindings
- Section 5 Specifies binding dependent functionality
- Changes Cumulative changes to 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. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder. (This note was editorially updated on 11 June 2008.).
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/65]
Program and System Information Protocol for Terrestrial Broadcast and Cable, A/65, ATSC

[A/90]
Data Broadcast Standard, A/90, ATSC

[A/94]
Data Application Reference Model (ARM), A/94, ATSC

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

[DASE-SECURITY]
DASE-1 Part 6: Security, A/100-6, ATSC

2.2 Informative References

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

[DASE-PA]
DASE-1 Part 3: Procedural Applications and Environment, A/100-3, ATSC

[JAVATV]

[MPEG-2]
Information technology – Generic coding of moving pictures and associated audio information: Systems, ISO/IEC 13818-1, ISO
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/.

ISO Standards

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

Java Standards

Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA; http://java.sun.com/.
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, which may or may not be present at the option of the application or the application environment implementer.

3.2 Acronyms and Abbreviations

ARM  Application Reference Model
IP    Internet Protocol
NVOD  Near Video On Demand

3.3 Terms

ARM data application: a data application as defined by [A/94].

service: a virtual channel, as defined by [A/65].

Note: Except when referring to a virtual channel table, this specification consistently uses the term service rather than virtual channel in order to remain consistent with general terminology employed by Java TV.
4. **BINDINGS**

The entirety of this section and its subsections is normative.

This section specifies a binding from a DASE System to an application delivery system embodied in mechanisms defined by [A/94].

4.1 **Application Delivery**

A DASE Application shall be delivered from an application emitter to a DASE System as an *ARM data application* as described in the following subsections.

4.1.1 **Application Announcement**

A DASE Application shall be announced in accordance with [A/94], Section 10.1, *Application Announcement*.

A DASE Application shall be announced using a compatibility descriptor in accordance with [A/94], Section 9.2, *DSM-CC Compatibility Descriptor*, and as constrained by the following:

- the *model* field shall be 0x0001, signifying a DASE Application Environment;
- the *version* field shall be 0x0001, signifying a level one (1) DASE application environment.

4.1.2 **Application Signaling**

A DASE Application shall be signaled in accordance with [A/94], Section 11.2, *Application Signaling*.

The resource which represents a DASE Application's root entity shall be signaled by a DST Tap in accordance with [A/94], Section 11.2. If the DASE Application is intended to be automatically launched by the DASE System, then the *action_type* field of this Tap shall be set to 0x01 (Bootstrap data); otherwise, it shall be set to 0x00 (Runtime data). At most one auto-loadable DASE Application signaled by a DST may have its root entity signaled as Bootstrap data. Multiple DASE Applications signaled by a DST may have their root entity signaled as Bootstrap data provided that all but one of the applications specify a *noautoload* application parameter with the value *true*.

*Note:* See [DASE], Section 6.1.1.6.13.5, for information on the *noautoload* parameter.

A DASE Application's root entity referenced by an application intended to be automatically launched by the DASE System shall be signaled with an acquisition directive as described in [A/94], Sections 11.3 and E.3.

The value of the *uuid* attribute of the *identifier* element of an application’s metadata resource shall coincide with the encoded UUID used to identify a DASE Application in the data service table.

A DASE Application shall be signaled using a compatibility descriptor in accordance with [A/94], Section 9.2, *DSM-CC Compatibility Descriptor*, with the same constraints as specified above in Section 4.1.1.

*Note:* See [DASE], Section 6.1.1.6.10, for information on how a DASE Application is identified in an application’s metadata resource.

4.1.3 **Application Resources**

The resources of a DASE Application shall be encapsulated in accordance with [A/94], Section 8, *Data Encapsulation*, as further constrained below.
4.1.3.1 Resource Identifiers

A DASE Application resource identifier shall be restricted to the following URI schemes for the purpose of accessing resources by means of this binding:

- "lid:"
- "tv:"

A relative resource identifier shall be absolutized by the application environment for the purpose of resolving a resource.

A fragment identifier shall be discarded by the application environment for the purpose of resolving a resource.

4.1.3.2 Bounded Resources

A bounded resource of a DASE Application shall be encapsulated either as a file in accordance with [A/94], Section 8.3, Files, or as a module in accordance with [A/94], Section 8.2, Modules. In either case, each bounded resource shall be labeled by a URI and its content type identified by an appropriate content type description.

When a bounded resource is delivered as a file, then each non-terminal component of the pathname from the logical root of the application delivery file system to the file shall correspond to a distinct directory in the local file system extending downwards from the mount point of the application delivery file system. In addition, each non-terminal component shall not contain the pathname separator character ‘/’ and shall not be equal to either "." or "..".

At most one DST Tap of a DASE Application may refer to a root directory of an application delivery file system. The first application delivery file system referenced by a DST Tap signaling a DASE Application shall be automatically mounted upon some directory of the local file system, where the location of the directory is not defined by this specification. This automatic mount process shall occur prior to resolving references to any directory or file within the mounted file system.

If one or more DST Taps of a DASE Application refer to files contained within a mounted application delivery file system, then they shall be accessed as if the locally mounted reflection of the file in the local file system were accessed, excepting access permissions.

Note: See [DASE-SECURITY], Section C.3 for additional information on access permission for application delivery file system elements.

If an application delivery file system makes reference to another application delivery file system, then such references shall be construed as empty.

Note: The above constraint is expected to be removed in a future level of the DASE Standard.

A bounded resource shall be referenced only through the use of a URI of the "lid:" scheme type.

4.1.3.3 Unbounded Resources

An unbounded resource takes one of three forms as described by the following sections: (1) asynchronous data streams, (2) IP packet streams, and (3) trigger streams.

A URI string representation used to reference an unbounded resource shall be no longer than 255 octets in length.
4.1.3.3.1 Asynchronous Data Streams

An asynchronous data stream, which is exposed by means of a built-in data source, shall be encapsulated as a stream in accordance with [A/94], Section 8.4, Streams and Table 1 Asynchronous Stream Encapsulation.

Note: See [DASE-PA], Section 5.1.1.2.2.4.1, for more information on the use of unbounded resources as asynchronous data streams.

Table 1 Asynchronous Stream Encapsulation

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Encapsulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>atsc.async.piping</td>
<td>proprietary data piping</td>
</tr>
<tr>
<td>atsc.async.piping.raw</td>
<td>proprietary data piping</td>
</tr>
<tr>
<td>atsc.async.download</td>
<td>asynchronous non-flow controlled scenario</td>
</tr>
<tr>
<td>atsc.async.dtvcc</td>
<td>picture user data type 0x03</td>
</tr>
</tbody>
</table>

Except for the atsc.async.dtvcc data source, an asynchronous stream shall be referenced only through the use of a URI of the "lid:" scheme type.

An atsc.async.dtvcc data source shall be referenced only through the use of a URI of the "tv:" scheme type. Furthermore, such a reference shall be restricted to the specific URI "tv:?dtvcc", which shall be interpreted as referencing the picture user data carrying the digital television closed captioning data within the active video elementary stream.

Note: See [DASE-PA], Section 5.1.1.2.2.4.1.4, for more information on the asynchronous digital television closed captioning data source.

4.1.3.3.2 IP Packet Streams

An IP (internet protocol) packet stream, which is exposed by means of a datagram socket, shall be encapsulated in accordance with [A/94], Section 8.1, IP Packets.

An IP packet stream shall be referenced only through the use of a URI of the "lid:" scheme type.

Note: See [JAVATV], javax.tv.net.InterfaceMap, and [PJAE], java.net.DatagramSocket and java.net.MulticastSocket, for more information on the use of unbounded resources as IP packet streams.

4.1.3.3.3 Trigger Streams

A trigger stream, which delivers asynchronous DASE trigger resources, shall be encapsulated in accordance with [A/94], Section 8.5, Triggers, as restricted below.

An MPEG-2 Registration Descriptor (MRD) need not accompany the signaling of a DASE Trigger Stream.

A trigger stream shall be referenced only through the use of a URI of the "lid:" scheme type.

Note: A DASE Application typically does not make direct reference to a trigger stream; rather, information in the stream specifies a target for the trigger stream or individual triggers.

Note: In the context of DASE triggers, the term target refers to an application element intended to receive a DASE trigger event; this usage is to be considered distinct from the notion of a data target as a data trigger pre-load resource.
Note: See [DASE], Section 6.9, for more information on the use of unbounded resources as trigger streams.

4.1.3.3.3.1 DASE Trigger Encapsulation

Individual DASE trigger resources shall be encapsulated in the a `daseTrigger()` structure within the `user_data_byte[]` field of an asynchronous data trigger's `event_info` structure:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Number of bits</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>daseTrigger()</code> {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>marker</td>
<td>32</td>
<td>uimsbf</td>
</tr>
<tr>
<td>version</td>
<td>4</td>
<td>bslbf</td>
</tr>
<tr>
<td>reserved</td>
<td>12</td>
<td>bslbf</td>
</tr>
<tr>
<td>triggerLength</td>
<td>16</td>
<td>uimsbf</td>
</tr>
<tr>
<td>for ( i=0; i&lt;triggerLength; i++)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>triggerByte[i]</td>
<td>8</td>
<td>uimsbf</td>
</tr>
<tr>
<td>signatureLength</td>
<td>16</td>
<td>uimsbf</td>
</tr>
<tr>
<td>for ( i=0; i&lt;signatureLength; i++)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>signatureByte[i]</td>
<td>8</td>
<td>uimsbf</td>
</tr>
</tbody>
</table>

The field `marker` shall be set to the constant value 0x44415345. The field `version` shall be set to the constant value zero (0) to signify this version of the structure. The field `reserved` is reserved for definition by future levels of the DASE Standard, and shall be set to all ones for this version of the structure.

The field `triggerLength` shall specify the length of the `triggerByte[]` field in octets. The field `triggerByte[]` shall contain the serialized representation of a DASE Trigger entity as defined by [DASE], Section 6.9.

The field `signatureLength` shall specify the length of the `signatureByte[]` field in octets. The field `signatureByte[]` shall be empty for DASE-1 Applications.

4.1.3.4 Implied Resources

Certain uses of the "tv:" scheme imply a reference to a resource as specified in Table 3 Implied Resource References. In these cases, the resource is not exposed directly to the DASE System; nevertheless, the application delivery system shall be able to determine and report whether such an implied resource exists and, if it does exist, what its content type is.

The use of resource references to these implied resources shall adhere to [A/94], Section 7.3, Content Types and Their URI Dependent Behavior.

<table>
<thead>
<tr>
<th>URI Form</th>
<th>Implied Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>tv: without query identifier</td>
<td>service</td>
</tr>
<tr>
<td>tv: with video query identifier</td>
<td>video elementary stream</td>
</tr>
<tr>
<td>tv: with audio query identifier</td>
<td>audio elementary stream</td>
</tr>
<tr>
<td>tv: with video and audio query identifier</td>
<td>multiplex of video and audio elementary streams</td>
</tr>
</tbody>
</table>

These implied resources shall be referenced only through the use of a URI of the "tv:" scheme type.
4.2 Application State and Events

A DASE Application’s lifecycle is controlled in part by transitions in the application delivery system’s data application state in accordance with [A/94], Section 6, Application State Model. Conversely, a DASE Application’s lifecycle state controls in part the application delivery system’s data application state model. The following subsections describe this control between the application delivery system and application environment layers.

Note: See [DASE], Section 5.1.3, Application Lifecycle, for further information on the DASE Application lifecycle.

4.2.1 Application Delivery System Initiated Events

This section specifies the events which originate in the application delivery system and which are delivered to the DASE System (application environment) for further processing.

4.2.1.1 Data Application State Transition Event Mapping

A DASE System shall map transitions in the data application state model to events which affect the DASE Application lifecycle as specified in Table 4 Data Application State Transition Mappings.

Note: See [A/94], Section 6, Application State Model, for further information on the data application state model events. See [DASE], Section 5.1.3, Application Lifecycle, for further information on the DASE Application lifecycle.

Table 4 Data Application State Transition Mappings

<table>
<thead>
<tr>
<th>Data Application State Transition</th>
<th>DASE Application Lifecycle Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>enter loading state</td>
<td>initialize</td>
</tr>
<tr>
<td>enter running state</td>
<td>resume</td>
</tr>
<tr>
<td>enter suspended state</td>
<td>suspend</td>
</tr>
<tr>
<td>enter unloaded state</td>
<td>terminate</td>
</tr>
</tbody>
</table>

4.2.1.2 Other Application Delivery System Event Mapping

The application delivery system shall report certain other events which occur at the transport and data application model layers to the DASE System for those events which affect the DASE System as specified in Table 5 Other Application Delivery System Event Mappings.

Table 5 Other Application Delivery System Event Mappings

<table>
<thead>
<tr>
<th>Transport or Data Application Layer Event</th>
<th>DASE Application Environment Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAT changed</td>
<td>notify PAT changed</td>
</tr>
<tr>
<td>PMT changed</td>
<td>notify PMT changed</td>
</tr>
<tr>
<td>CAT changed</td>
<td>notify CAT changed</td>
</tr>
<tr>
<td>VCT changed</td>
<td>notify VCT changed</td>
</tr>
<tr>
<td>EIT changed</td>
<td>notify EIT changed</td>
</tr>
<tr>
<td>DET changed</td>
<td>notify DET changed</td>
</tr>
<tr>
<td>LTST changed</td>
<td>notify LTST changed</td>
</tr>
<tr>
<td>DST changed</td>
<td>notify DST changed</td>
</tr>
<tr>
<td>NRT changed</td>
<td>notify NRT changed</td>
</tr>
<tr>
<td>transport stream changed</td>
<td>Note 1</td>
</tr>
<tr>
<td>carousel file changed</td>
<td>Note 2</td>
</tr>
</tbody>
</table>
Notes

1. When service selection causes a change in visibility of a transport stream, then the DASE application environment should be notified of this change.

   Note: See [JAVATV], javax.tv.service.transport.TransportStreamCollection, for more information on the visibility of transport stream change events.

2. When a carousel file delivered by either a data or object carousel changes, then the DASE application environment should be notified of this change.

   Note: See [JAVATV], javax.tv.carousel.CarouselFile, for more information on the visibility of carousel file change events. See also [DASE-DA], Section 5.3.1.2.8.3, Mutation Event Types, for information on reporting resource change events.

4.2.2 Application Environment Initiated Events

This section specifies the events which originate in the DASE System (application environment) and which are delivered to the application delivery system for further processing.

4.2.2.1 Application Lifecycle State Transition Event Mapping

A DASE System shall report transitions in the DASE Application lifecycle to the application delivery system for those events which affect the data application state model as specified in Table 6 Application Lifecycle State Transition Mappings.

   Note: See [DASE], Section 5.1.3, Application Lifecycle, for further information on the DASE Application lifecycle. See [A/94], Section 6, Application State Model, for further information on the data application state model events.

Table 6 Application Lifecycle State Transition Mappings

<table>
<thead>
<tr>
<th>DASE Application Lifecycle Transition</th>
<th>Data Application State Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>enter initialized state</td>
<td>load complete</td>
</tr>
</tbody>
</table>

4.2.2.2 Other Application Environment Event Mapping

A DASE System shall report certain other events in a DASE application environment to the application delivery system for those events which affect the data application state model as specified in Table 7 Other Application Environment Event Mappings.

Table 7 Other Application Environment Event Mappings

<table>
<thead>
<tr>
<th>DASE Application Environment Event</th>
<th>Data Application State Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>load application</td>
<td>load(appid)</td>
</tr>
<tr>
<td>terminate current application</td>
<td>terminate(appid)</td>
</tr>
<tr>
<td>select service</td>
<td>channel change</td>
</tr>
</tbody>
</table>

Note: The argument appid to the above data application state events represent the internal representation of an application identifier which serves to identify a particular DASE Application. See Section A.1.9.1.2 for additional information on an application identifier.

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

Note: A service selection event is generated internally within a DASE Application Environment as a side effect of certain operations.
4.3 Application Permissions

When determining the effective permissions of a DASE Application as prescribed by [DASE-SECURITY], Section 4, a DASE System shall use an application emission policy in accordance with [A/94], Section 9.4, Broadcaster Policy Descriptor. Application emission policies shall affect the permissions granted to a DASE Application as specified in Table 8 Application Emission Policy Mappings.

Table 8 Application Emission Policy Mappings

<table>
<thead>
<tr>
<th>Emission Policy Denial</th>
<th>DASE System Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>change of receive channel</td>
<td>deny service selection between transport streams</td>
</tr>
<tr>
<td>change of service</td>
<td>deny service selection within transport streams</td>
</tr>
<tr>
<td>executing content</td>
<td>deny execution of active object and script content</td>
</tr>
<tr>
<td>access to external interfaces</td>
<td>Note 1</td>
</tr>
<tr>
<td>overlay A/53 video</td>
<td>deny graphics presentation over video plane</td>
</tr>
<tr>
<td>scale A/53 video</td>
<td>deny video resize requests</td>
</tr>
<tr>
<td>use inactive regions of display</td>
<td>deny graphics presentation outside video plane</td>
</tr>
<tr>
<td>mix or augment audio</td>
<td>deny audio presentation</td>
</tr>
<tr>
<td>replace audio</td>
<td>deny audio presentation</td>
</tr>
<tr>
<td>read viewer preferences</td>
<td>deny access to preferences registry state</td>
</tr>
</tbody>
</table>

Notes

1. The DASE-1 Standard does not specify any mechanism to which this emission policy applies.

   The use of application emission policy information shall only occur upon the initial activation of a DASE Application. Any change that occurs in the application emission policy information during the performance of the application shall not affect its effective permissions.

   If an application emission policy does not explicitly deny the use of some function, then a DASE System shall consider the performance of the function to be granted with respect to application emission policy.

   Note: The use of application emission policy information as described above is not intended to be used to control native applications, but only DASE applications.

   Note: Certain of the broadcaster policies do not map directly to a privileged operation as specified by [DASE-SECURITY], Annex B. In these cases, a DASE System is expected to use implementation specific mechanisms to control application behavior.
5. **BINDING DEPENDENT FUNCTIONALITY**

This section specifies binding dependent functionality or constraints on general functionality which is exposed by means of DASE System facilities.

A DASE System which implements this binding shall indicate this support by means of the Java system property "dase.delivery.system", whose value shall include the token "ARM".

*Note:* See [DASE-PA], Annex C, for more information on Java system properties.

A DASE Application may make use of binding dependent functionality only by properly guarding such usage by an appropriate conditionalization feature.

*Note:* A DASE Application may query the Java system property which specifies the supported application delivery systems as a way of learning if binding dependent functionality is present or not.

5.1 **Active Object Content**

This section specifies binding dependent active object functionality and constraints on general active object functionality.

5.1.1 **application/java**

This section specifies certain application programming interfaces made available to application entities of content type `application/java` only when used with this application delivery system.

*Note:* The definition and constraints regarding `application/java` content type describe here apply also to content types derived from the `application/java` content type; namely, the `application/javatv-xlet` content type.

5.1.1.1 **Java Television (Java TV) Interfaces**

This section specifies constraints on certain Java Television (Java TV) APIs when a DASE System makes use of this application delivery system.

5.1.1.1.1 **Constraints**

5.1.1.1.1.1 **Constraints on `javax.tv.locator.Locator` interface**

The method `getExternalForm()` shall return an external form of the locator in accordance with Table 9 Locator External Forms.

**Table 9 Locator External Forms**

<table>
<thead>
<tr>
<th>Located Element Category</th>
<th>External Form</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>transport stream</td>
<td>implementation defined</td>
<td>1</td>
</tr>
<tr>
<td>service</td>
<td>tv: resource identifier</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>service details</td>
<td>implementation defined</td>
<td>1</td>
</tr>
<tr>
<td>service component</td>
<td>tv: resource identifier</td>
<td>2, 5</td>
</tr>
<tr>
<td>program event</td>
<td>implementation defined</td>
<td>1</td>
</tr>
<tr>
<td>data event</td>
<td>implementation defined</td>
<td>1</td>
</tr>
<tr>
<td>data service</td>
<td>implementation defined</td>
<td>1</td>
</tr>
<tr>
<td>application</td>
<td>lid: resource identifier</td>
<td>2, 6</td>
</tr>
</tbody>
</table>
application resource - bounded | lid: resource identifier | 2, 7
application resource - unbounded | lid: or tv: resource identifier | 2, 8

Notes

1. The external form is not defined by this specification; however, in order to facilitate the determination of which external forms have been generated by an implementation, an implementation should make use of an external form which is easily distinguished from standardized forms, e.g., an implementation may choose to use an external form which adheres to the general syntax of a URI and which uses a scheme component of "internal:". An application shall not rely upon the precise syntax or value of an implementation defined external form.

2. See Section 4.1.3.1 for further information on permissible resource identifiers.

3. The service located element category is represented using an instance of javax.tv.-service.navigation.ServiceDetails or a subtype thereof.

4. If a service is associated with a service name, then the external form shall make use of that service name; however, if no service name is associated with the service, then an implementation should synthesize an internal service name which has a low likelihood of colliding with an externally specified service name.

5. If a service component is associated with a component name descriptor, then the external form shall make use of that component name; however, if no component name is associated with the component, then an implementation should synthesize an internal component name which has a low likelihood of colliding with an externally specified component name.

6. If a data service application is a DASE Application, then the external form of the locator which identifies the application shall be the same as the external form of the locator which identifies the DASE Application's root resource; i.e., the application metadata resource.

7. A carousel file is to be construed as a bounded application resource.

8. All unbounded application resources except for a closed captioning data source are identified using the lid: resource identifier scheme; whereas, a closed captioning data stream is identified using the tv:?dtvcc resource identifier.

5.1.1.1.2 Constraints on javax.tv.net.InterfaceMap interface

The argument to the getLocalAddress(Locator) method shall reference an application or a service component (program element). The network assigned to the reference shall govern all multicast datagrams associated with the referenced application or service component.

5.1.1.1.3 Constraints on javax.tv.service.Service interface

An object that implements the Service interface shall correspond to a virtual channel entry delivered by a virtual channel table (VCT) as defined by [A/65], Section 6.3.

The value returned by getName() shall correspond to the short_name field of an entry of a virtual channel table (VCT) as defined by [A/65], Section 6.3.

When retrieving the details of a service that contains a data service component using retrieveDetails(), the retrieved SIElement shall implement org.atsc.dataDataServiceDetails; otherwise, the retrieved SIElement shall implement javax.tv.service.nagivation.-ServiceDetails.

Note: See DataServiceDetails.
5.1.1.1.4 **Constraints on javax.tv.service.SIRetrievable interface**

The method `getUpdateTime()` should return a time and date which is as close as possible to the time when the service information was retrieved from the transport stream.

5.1.1.1.5 **Constraints on javax.tv.service.guide.ProgramEventDescription interface**

A program event description shall correspond to an extended text message (ETM) delivered by an extended text table (ETT) as defined by [A/65], Section 6.6.

If multiple language descriptions are available, then the description whose language matches the current locale shall be used; or, if no description’s language matches the current locale, an implementation dependent default language description may be used.

5.1.1.1.6 **Constraints on javax.tv.service.guide.ProgramEvent interface**

A program event shall correspond to an entry of the event information table (EIT) as defined by [A/65], Section 6.5.

If multiple language program event names (titles) are available, then the name whose language matches the current locale shall be used; or, if no name’s language matches the current locale, an implementation dependent default language name may be used.

5.1.1.1.7 **Constraints on javax.tv.service.guide.ProgramSchedule interface**

A program schedule shall correspond to a collection of event information tables (EITs) as defined by [A/65], Section 6.5.

5.1.1.1.8 **Constraints on javax.tv.service.navigation.ServiceDescription interface**

A service description shall correspond to an extended text message (ETM) referenced by a virtual channel table (VCT) and delivered by an extended text table (ETT) in accordance with [A/65], Sections 6.3 and 6.6.

If multiple language descriptions are available, then the description whose language matches the current locale shall be used; or, if no description’s language matches the current locale, an implementation dependent default language description may be used.

5.1.1.1.9 **Constraints on javax.tv.service.navigation.ServiceDetails interface**

An object that implements `ServiceDetails` interface shall correspond to detailed service information provided by or referenced by a virtual channel entry of a virtual channel table (VCT) as defined by [A/65], Section 6.3.

The value returned by `getLongName()` shall correspond to an extended channel name descriptor delivered in an entry of a virtual channel table (VCT) as defined by [A/65], Sections 6.7.5 and 6.3, respectively. If no extended channel name descriptor is available, then the value of the `short_name` field of the VCT may be used as an alternative.

If multiple language descriptions are available, then the description whose language matches the current locale shall be used; or, if no description’s language matches the current locale, an implementation dependent default language description may be used.

5.1.1.2 **DASE Specific (ATSC) Interfaces**

This section specifies constraints on and definitions of certain DASE Specific (ATSC) APIs when a DASE System makes use of this application delivery system.
If a DASE System supports the application delivery system binding defined by this specification, then an application entity may use and a procedural application environment shall implement the packages specified by the following subsections.

5.1.1.2.1 Packages

5.1.1.2.1.1 org.atsc.data

The org.atsc.data package comprises the following types as specified by Section A.1 below.

<table>
<thead>
<tr>
<th>Compatibility</th>
<th>DataScheduleChangeType</th>
<th>DataServiceApplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataEvent</td>
<td>DataScheduleEvent</td>
<td>DataServiceChangeEvent</td>
</tr>
<tr>
<td>DataEventDescription</td>
<td>DataScheduleListener</td>
<td>DataServiceDetails</td>
</tr>
<tr>
<td>DataSchedule</td>
<td>DataService</td>
<td>DataServiceListener</td>
</tr>
</tbody>
</table>

5.1.1.2.1.2 org.atsc.si

The org.atsc.si package comprises the following types as specified by Section A.2 below.

<table>
<thead>
<tr>
<th>AtscLocator</th>
<th>MPEGLocator</th>
<th>TimeShiftedService</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtscService</td>
<td>NVODReference</td>
<td></td>
</tr>
</tbody>
</table>
ANNEX A.  BINDING DEPENDENT PACKAGES

The entirety of this annex and its subsections is normative.

A.1  org.atsc.data

This package provides functionality for accessing features of a data service as defined by [A/94], [A/90], and [A/65].

A.1.1  Compatibility

    public interface Compatibility

    The Compatibility interface provides access to the compatibility descriptor associated with the data service application.

A.1.1.1  Methods

A.1.1.1.1  getContent()

    public byte[] getContent()

    This method provides access to the raw descriptor information of the data service application compatibility information.

    The array returned shall consist of the contents of the compatibilityDescriptorByte[] array found in the compatibilityDescriptorWrapper as defined by [A/94], Section 9.1.

    Returns:

    A byte array containing the compatibility descriptor bytes.

A.1.2  DataEvent

    extends javax.tv.service.SIElement

    The DataEvent interface provides information about a data event, including name (title), start time, and duration.

    A data event shall correspond to a distinct entry of the data event table (DET) or long-term service table (LTST) as defined by [A/90], Sections 11.3 and 11.7.

A.1.2.1  Methods

    The following methods are inherited from javax.tv.service.SIElement: equals,
    getLocator, getServiceInformationType, hashCode.

    The following methods are inherited from javax.tv.service.SIRetrievable:
    getUpdateTime.
A.1.2.1.1 getDuration()

public long getDuration()

Returns the duration of this data event in seconds. If the event is unbounded, then the value -1 is returned.

Returns:

This data event’s duration in seconds or -1 in the case of an unbounded event.

A.1.2.1.2 getEndTime()

public java.util.Date getEndTime()

Returns the end time of this data event. The end time is in UTC time. If the event is unbounded, then the value null is returned.

Returns:

This data event’s end time (UTC) or null if the event is unbounded.

A.1.2.1.3 getName()

public java.lang.String getName()

Returns the data event title.

If multiple language data event names (titles) are available, then the name whose language matches the current locale shall be used; or, if no name’s language matches the current locale, an implementation dependent default language name may be used.

Returns:

A string representing this data event’s title.

A.1.2.1.4 getRating()

public javax.tv.service.guide.ContentRatingAdvisory getRating()

Returns content advisory information associated with this data event for the local rating region.

Returns:

A ContentRatingAdvisory object describing the rating of this data event or null if no rating information is available.

A.1.2.1.5 getService()

public javax.tv.service.Service getService()

Returns the service with which this data event is associated.

Returns:

The service on which this data event will be delivered.

A.1.2.1.6 getStartTime()

public java.util.Date getStartTime()

Returns the start time of this data event. The start time is in UTC time.
Returns:
This data event's start time (UTC).

A.1.2.1.7 retrieveDescription(javax.tv.service.SIRequestor)

public javax.tv.service.SIRequest
retrieveDescriptor(javax.tv.service.SIRequestor requestor)

Returns a description of the event. This method delivers its results asynchronously.
Parameters:
requestor – The SIRequestor to be notified when this retrieval operation completes.
Returns:
An SIRequest object identifying this asynchronous request.
See also: DataEventDescription.

A.1.2.2 Fields
No fields are defined.

A.1.3 DataEventDescription

public interface DataEventDescription
extends javax.tv.service.SIRetrievable

This interface provides a textual description of a DataEvent.

A data event description shall correspond to an extended text message (ETM) referenced by a data event table (DET) or long term service table (LTST) and delivered by an extended text table (ETT) in accordance with [A/90], Sections 11.3 and 11.7, and [A/65], Section 6.6.

If multiple language descriptions are available, then the description whose language matches the current locale shall be used; or, if no description’s language matches the current locale, an implementation dependent default language description may be used.

A.1.3.1 Methods

The following methods are inherited from javax.tv.service.SIRetrievable:
getUpdateTime.

A.1.3.1.1 getDataEventDescription()

public java.lang.String getDataEventDescription()

Provides a textual description of the DataEvent.
Returns:
A textual description of the DataEvent or an empty string if no description is available.

A.1.3.2 Fields
No fields are defined.
A.1.4 DataSchedule

public interface DataSchedule

This interface represents a collection of data events for a given service ordered by time. It provides the current, next and future data events.

A data schedule shall correspond to a collection of data event tables (DETs) and long-term service tables (LTSTs) as defined by [A/90], Sections 11.3 and 11.7.

Note that all time values are in UTC time.

A.1.4.1 Methods

A.1.4.1.1 addListener(DataScheduleListener)

public void addListener(DataScheduleListener listener)

Registers a DataScheduleListener to be notified of changes to data events on this DataSchedule. Subsequent changes will be indicated through instances of DataScheduleEvent, with this DataSchedule as the event source and a DataScheduleChangeType of ADD, REMOVE, MODIFY, or CURRENT_DATA_EVENT. Only changes to DataEvent instances of which the caller has javax.tv.service.ReadPermission(d.getLocator()) will be reported.

This method provides a means to request notification only. No guarantee is provided that the service information database will detect all, or even any, changes to the DataSchedule, or whether such changes will be detected in a timely fashion.

If the specified DataScheduleListener is already registered, no action is performed.

Parameters:

listener – A DataScheduleListener to be notified of changes related to data events on this DataSchedule.

See also: DataEvent, DataScheduleEvent, DataScheduleChangeType, and javax.tv.-service.ReadPermission.

A.1.4.1.2 getServiceLocator()

public javax.tv.locator.Locator getServiceLocator()

Reports the transport-dependent locator referencing the service to which this DataSchedule belongs. Note that applications may use this method to establish the identity of a DataSchedule after it has changed.

Returns:

The transport-dependent locator referencing the service to which this DataSchedule belongs.

A.1.4.1.3 removeListener(DataScheduleListener)

public void removeListener(DataScheduleListener listener)

Unregister a DataScheduleListener. If the DataScheduleListener is not registered, no action is performed.

Parameters:

listener – A previously registered listener.
A.1.4.1.4  retrieveCurrentDataEvent(javax.tv.service.SIRequestor)

```java
public javax.tv.service.SIRequest
retrieveCurrentDataEvent(javax.tv.service.SIRequestor requestor)
```

Returns the current DataEvent.

This method delivers its results asynchronously. If the caller does not have `javax.tv.service.ReadPermission(d.getLocator())` (where `d` is the current data event), this method will result in an `SIRequestFailureType` of `DATA_UNAVAILABLE`.

Parameters:

- `requestor` – The SIRequestor to be notified when this retrieval operation completes.

Returns:

- An SIRequest object identifying this asynchronous retrieval request.

A.1.4.1.5  retrieveDataEvent(javax.tv.locator.Locator, javax.tv.service.SIRequestor)

```java
public javax.tv.service.SIRequest
retrieveDataEvent(javax.tv.locator.Locator locator, javax.tv.service.SIRequestor requestor)
```

Retrieves a data event matching the locator. Note that the event must be part of this schedule. This method returns data asynchronously.

Parameters:

- `locator` – A locator referencing the DataEvent of interest.
- `requestor` – The SIRequestor to be notified when the retrieval operation completes.

Returns:

- An SIRequest object identifying this asynchronous retrieval request.

Throws:

- `javax.tv.locator.InvalidLocatorException` – If the locator does not reference a valid DataEvent in this DataSchedule.
- `java.lang.SecurityException` – If the caller does not have `javax.tv.service.-ReadPermission(locator)`.

A.1.4.1.6  retrieveFutureDataEvent(java.util.Date, javax.tv.service.SIRequestor)

```java
public javax.tv.service.SIRequest
retrieveFutureDataEvent(java.util.Date time, javax.tv.service.SIRequestor requestor)
```

Retrieves the data event for the specified time. The specified time will fall between the resulting data event’s start time (inclusive) and end time (exclusive).

This method delivers its results asynchronously. If the caller does not have `javax.tv.service.ReadPermission(d.getLocator())` (where `d` is the data event at the specified time), this method will result in an `SIRequestFailureType` of `DATA_UNAVAILABLE`.

Parameters:

- `time` – The time of the DataEvent to be retrieved.
requestor – The SIRequestor to be notified when the retrieval operation completes.

Returns:

An SIRequest object identifying this asynchronous retrieval request.

Throws:

javax.tv.service.SIException – If time does not represent a future time value.

A.1.4.1.7 retrieveFutureDataEvents(java.util.Date, java.util.Date, ...)

public javax.tv.service.SIRequest
retrieveFutureDataEvents
(java.util.Date begin,
java.util.Date end,
javax.tv.service.SIRequestor requestor)
throws javax.tv.service.SIException

Retrieves all known data events on this service for the specified time interval. A data event \( d \) is retrieved by this method if the time interval from \( d \).getStartTime() (inclusive) to \( d \).getEndTime() (exclusive) intersects the time interval from \( \text{begin} \) (inclusive) to \( \text{end} \) (exclusive) specified by the input parameters.

This method delivers its results asynchronously. If the caller does not have javax.tv.service.ReadPermission\( (d \).getLocator()) (where \( d \) is the data event at the specified time), this method will result in an SIRequestFailureType of DATA_UNAVAILABLE.

Parameters:

\( \text{begin} \) – Time identifying the beginning of the interval.

\( \text{end} \) – Time identifying the end of the interval.

\( \text{requestor} \) – The SIRequestor to be notified when the retrieval operation completes.

Returns:

An SIRequest object identifying this asynchronous retrieval request.

Throws:

javax.tv.service.SIException – If \( \text{end} \) represents a time value before \( \text{begin} \), or if \( \text{end} \) does not represent a future time value.

A.1.4.1.8 retrieveNextDataEvent(DataEvent, javax.tv.service.SIRequestor)

public javax.tv.service.SIRequest
retrieveNextDataEvent(DataEvent event, javax.tv.service.SIRequestor requestor)
throws javax.tv.service.SIException

Retrieves an event which follows the specified event.

This method delivers its results asynchronously. If the caller does not have javax.tv.service.ReadPermission\( (d \).getLocator()) (where \( d \) is the next data event), this method will result in an SIRequestFailureType of DATA_UNAVAILABLE.

Parameters:

\( \text{event} \) – A reference event.

\( \text{requestor} \) – The SIRequestor to be notified when this retrieval operation completes.
Returns:
   An SIRequest object identifying this asynchronous retrieval request.

Throws:
   javax.tv.service.SIException – If event does not belong to this DataSchedule.

A.1.4.2 Fields
   No fields are defined.

A.1.5 DataScheduleChangeEvent
   public class DataScheduleChangeEvent
      extends javax.tv.service.SIChangeEvent

   This class represents types of changes to data schedules. It is used by DataScheduleEvent
   to indicate the nature of the change which produced the event.
   
   See also: DataScheduleEvent, DataSchedule.

A.1.5.1 Constructors

A.1.5.1.1 DataScheduleChangeEvent(java.lang.String)
   protected DataScheduleChangeEvent(java.lang.String name)
      Protected constructor.
      Parameters:
      name – The string name of this type (e.g. "CURRENT_DATA_EVENT").

A.1.5.2 Methods
   The following methods are inherited from java.lang.Object: clone, equals, finalize,
   getClass, hashCode, notify, notifyAll, wait(), wait(long), wait(long, int).

A.1.5.2.1 toString()
   public java.lang.String toString()
      Provides the string name of the type. For the type objects defined in this class, the string
   name will be identical to the class variable name.
      Overrides:
      javax.tv.service.SIChangeEvent.toString()
      Returns:
      The string name of the type.

A.1.5.3 Fields
   The following fields are inherited from javax.tv.service.SIChangeEvent: ADD, MODIFY,
   REMOVE.
A.1.5.3.1 CURRENT_DATA_EVENT

public static final DataScheduleChangeType CURRENT_DATA_EVENT

DataScheduleChangeType indicating that the current data event has changed.

A.1.6 DataScheduleEvent

public class DataScheduleEvent
extends javax.tv.service.SIChangeEvent

A DataScheduleEvent notifies a DataScheduleListener of changes to data events detected in a DataSchedule. Specifically, this event signals the addition, removal, or modification of a DataEvent in a DataSchedule, or a change to the DataEvent that is current.

The class DataScheduleChangeType defines the kinds of changes reported by DataScheduleEvent. A DataScheduleChangeType of CURRENT_DATA_EVENT indicates that the current DataEvent of a DataSchedule has changed its identity.

See also: DataScheduleListener, DataScheduleChangeType.

A.1.6.1 Constructors

A.1.6.1.1 DataScheduleEvent(DataSchedule, ...)

public DataScheduleEvent
(DataSchedule schedule,
javax.tv.service.SIChangeEvent type,
DataEvent e)

Constructs a DataScheduleEvent.

Parameters:

schedule – The schedule in which the change occurred.

type – The type of change that occurred.

e – The DataEvent that changed.

A.1.6.2 Methods

The following methods are inherited from javax.tv.service.SIChangeEvent: getChangeType, getSIElement.

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

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

A.1.6.2.1 getDataEvent()

public DataEvent getDataEvent()

Reports the DataEvent that changed. If the DataScheduleChangeType is CURRENT_DATA_EVENT, the DataEvent that became current will be returned. The object returned will be identical to the object returned by the inherited SIChangeEvent.getSIElement method.

Returns:

The DataEvent that changed.
See also: `javax.tv.service.SIChangeEvent.getSIElement()`.

A.1.6.2.2  getDataSchedule()

```java
public DataSchedule getDataSchedule()
```

Reports the DataSchedule that generated the event. The object returned will be identical to the object returned by the inherited `EventObject.getSource()` method.

Returns:

The DataSchedule that generated the event.

See also: `java.util.EventObject.getSource()`.

A.1.6.3  Fields

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

A.1.7  DataScheduleListener

```java
public interface DataScheduleListener
    extends javax.tv.service.SIChangeListener
```

This interface is implemented by applications wishing to receive notification of changes to DataSchedule data.

A.1.7.1  Methods

A.1.7.1.1  notifyChange(DataScheduleEvent)

```java
public void notifyChange(DataScheduleEvent event)
```

Notifies the DataScheduleListener of a change to a DataSchedule.

Parameters:

- `event` — A DataScheduleEvent describing what changed and how.

A.1.7.2  Fields

No fields are defined.

A.1.8  DataService

```java
public interface DataService
    extends javax.tv.service.SIElement
```

The DataService interface provides access to information associated with a data service.

When a DASE Application is delivered by means of the application delivery system binding described in this specification, then the method `org.atsc.xlet.XletContextExt.getDataService()` shall return an object which implements this interface.

Note: See [DASE-API], Section 4.17.6.1.1, for more information on `XletContextExt.getDataService()`.
A.1.8.1 Methods

The following methods are inherited from `javax.tv.service.SIElement`:
- `equals`,
- `getLocator`,
- `getServiceInformationType`,
- `hashCode`.

The following methods are inherited from `javax.tv.service.SIRetrievable`:
- `getUpdateTime`.

A.1.8.1.1 addListener(DataServiceListener)

```java
public void addListener(DataServiceListener listener)
```

 Registers a `DataServiceListener` to be notified of changes to this data service. Subsequent changes will be indicated through instances of `DataServiceChangeEvent`, with this `DataService` as the event source and a `javax.tv.service.SIChangeType` of `ADD`, `REMOVE`, or `MODIFY`.

This method provides a means to request notification only. No guarantee is provided that the DASE System will detect all, or even any, changes to the `DataService`, or whether such changes will be detected in a timely fashion.

If the specified `DataServiceListener` is already registered, no action is performed.

Parameters:

- `listener` – A `DataServiceListener` to be notified of changes related to data events on this `DataService`.

A.1.8.1.2 getApplication(javax.tv.locator.Locator)

```java
public DataServiceApplication getApplication(javax.tv.locator.Locator locator)
```

Retrieves the application identified by the specified locator.

Parameters:

- `locator` – A locator identifying the application.

Returns:

- A `DataServiceApplication` identified by `locator`.

Throws:

- `javax.tv.locator.InvalidLocatorException` – If the specified locator does not reference an application or does not reference an application in this data service.

A.1.8.1.3 getApplication(java.lang.String)

```java
public DataServiceApplication getApplication(java.lang.String appId)
```

Retrieves the application identified by the specified application identifier.

*Note:* See Section A.1.9.1.2 for information on an application identifier.

*Note:* This identifier does not take the form of a URI, but is the string representation of the UUID used to uniquely identify the application.

Parameters:

- `appId` – A unique string identifying the application.
Returns:
A DataServiceApplication identified by application identifier or null if the application cannot be found in this data service.

A.1.8.1.4 getApplications()

public DataServiceApplication[] getApplications()

Retrieves a list of DASE Applications associated with this data service. Any application in the data service which is not a DASE Application shall not be retrieved.

Returns:
An array of DataServiceApplication instances.

A.1.8.1.5 getPrivateData()

public byte[] getPrivateData()

Retrieves the private data associated with this data service.

The array returned shall consist of the contents of the service_private_data_byte[] array found in the data_service_table_bytes as defined by [A/90], Section 12.2, Data Service Table.

Returns:
A byte array of private data for this data service.

A.1.8.1.6 removeListener(DataServiceListener)

public void removeListener(DataServiceListener listener)

Unregister a DataServiceListener. If the DataServiceListener is not registered, no action is performed.

Parameters:
listener – A previously registered listener.

A.1.8.2 Fields

No fields are defined.

A.1.9 DataServiceApplication

public interface DataServiceApplication
extends javax.tv.service.SIElement

The DataServiceApplication interface represents a data service application associated with a data service as defined by [A/90]. This interface may be used to discover information about a DASE Application.

A.1.9.1 Methods

The following methods are inherited from javax.tv.service.SIElement: equals, getLocator, getServiceInformationType, hashCode.

The following methods are inherited from javax.tv.service.SIRetrievable: getUpdateTime.
A.1.9.1.1 getCompatibilityInformation()

public Compatibility[] getCompatibilityInformation()

Retrieves the Compatibility instances associated with this DataServiceApplication.

Returns:
An array of Compatibility instances which apply to this DataServiceApplication.

See also: Compatibility.

A.1.9.1.2 getIdentifier()

public java.lang.String getIdentifier()

Retrieves a unique identifier associated with this DataServiceApplication, where the identifier is derived from the app_id_byte[] array according to [A/90], Section 12.2. The value of this identifier shall be identical to the value of the uuid attribute of the identifier element of an application’s metadata resource.

Note: See [DASE], Section 6.1.1.6.10, for information on how a DASE Application is identified in an application’s metadata resource.

Returns:
A string identifier for this DataServiceApplication.

A.1.9.1.3 getPrivateData()

public byte[] getPrivateData()

Retrieves the private data associated with this DataServiceApplication.

The array returned shall consist of the contents of the app_data_byte[] array found in the data_service_table_bytes[] array as defined by [A/90], Section 12.2, Data Service Table.

Returns:
A byte array of private data for this data service application.

A.1.9.1.4 getResourceContentType(javax.tv.locator.Locator)

public java.lang.String getResourceContentType(javax.tv.locator.Locator locator) throws javax.tv.locator.InvalidLocatorException

Returns the content type of the application resource identified by the specified locator. Content type information shall be determined using the content type descriptor or application resource metadata information as defined by [A/94].

Parameters:
locator – A javax.tv.locator.Locator instance identifying an application resource.

Returns:
A string representing the content type. The form of the returned value shall adhere to the syntax prescribed by [DASE], Section 5.1.2.3. An empty string is returned if the content type is not available.

Throws:
javax.tv.locator.InvalidLocatorException – If the specified locator does not reference a valid application resource.
A.1.9.1.5 getResourceLocator(java.lang.String uri)

public javax.tv.locator.Locator getResourceLocator(java.lang.String uri)
    throws javax.tv.locator.MalformedLocatorException

Retrieves the javax.tv.locator.Locator of the application resource identified by the specified resource identifier.

Parameters:
uri – A resource identifier as permitted by Section 4.1.3.1.

Returns:
A javax.tv.locator.Locator which identifies the application resource or null if no such resource is present in this data application.

Throws:
javax.tv.locator.MalformedLocatorException – if the specified identifier could not be parsed.

See also: javax.tv.locator.Locator.

A.1.9.1.6 getResourceLocators()

public Locator[] getResourceLocators()

Retrieves a Locator instance for each resource present in this DataServiceApplication.

Returns:
An array of javax.tv.locator.Locator instances.

Note: The set of Locator instances returned by this method may contain multiple locators which reference the same underlying bounded resource; consequently, the arity of this set should not be used as an indication of the number of distinct bounded resources.

A.1.9.2 Fields

No fields are defined.

A.1.10 DataServiceChangeEvent

public class DataServiceChangeEvent
    extends javax.tv.service.SIChangeEvent

A DataServiceChangeEvent notifies a DataServiceListener of certain changes to a data service application, such as changes in compatibility or private data, or changes to application resources.

A.1.10.1 Constructors

A.1.10.1.1 DataServiceChangeEvent(DataService, ...)

public DataServiceChangeEvent
    (DataService source,
    javax.tv.service.SIChangeEvent type,
    DataServiceApplication app,
    javax.tv.locator.Locator locator)
Constructs a `DataServiceChangeEvent` instance.

Parameters:

- `source` – The `DataService` that generated this event.
- `type` – The type of data service change.
- `app` – The `DataServiceApplication` that has incurred some change.
- `locator` – The locator which identifies the changed data resource.

### A.1.10.2 Methods

The following methods are inherited from `javax.tv.service.SIChangeEvent`:

- `getChangeType`, `getSIElement`.

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

- `getSource`, `toString`.

The following methods are inherited from `java.lang.Object`:

- `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait()`, `wait(long)`, `wait(long, int)`.

#### A.1.10.2.1 `getApplication()`

```java
public DataServiceApplication getApplication()
```

Retrieves the `DataServiceApplication` which has incurred some change. The object returned will be identical to the object returned by inherited `SIChangeEvent.getSIElement()` method.

Returns:

- The `DataServiceApplication` which changed.

See also: `javax.tv.service.SIChangeEvent.getSIElement()`.

#### A.1.10.2.2 `getChangedResourceLocator()`

```java
public javax.tv.locator.Locator getChangedResourceLocator()
```

Retrieves the locator which identifies a changed data resource if a changed resource was responsible for the application change.

Returns:

- A locator of the changed resource or `null` if no resource changed or the implementation is not able to detect which resource changed.

#### A.1.10.2.3 `getDataService()`

```java
public DataService getDataService()
```

Retrieves the data service that generated the event. The object returned will be identical to the object returned by the inherited `EventObject.getSource()` method.

Returns:

- The `DataService` that generated the event.

### A.1.10.3 Fields

The following fields are inherited from `java.util.EventObject`: source.
A.1.11 DataServiceDetails

public interface DataServiceDetails
    extends javax.tv.service.navigation.ServiceDetails

The DataServiceDetails interface provides access to DataService objects associated with a service which contains a data service component.

If a service contains a data service component as defined by [A/90], then a request to retrieve the service details of that service shall return an instance of this DataServiceDetails subinterface.

See also: DataService, javax.tv.service.navigation.ServiceDetails.

A.1.11.1 Methods

The following methods are inherited from javax.tv.service.navigation.ServiceDetails: addServiceComponentChangeListener, getDeliverySystemType, getLongName, getProgramSchedule, getService, getServiceType, removeServiceComponentChangeListener, retrieveComponents, retrieveServiceDescription.

The following methods are inherited from javax.tv.service.SIElement: equals, getLocator, getServiceInformationType, hashCode.

The following methods are inherited from javax.tv.service.SIRetrievable: getUpdateTime.

The following methods are inherited from javax.tv.service.navigation.CAIdentification: getCASystemIDs, isFree.

A.1.11.1.1 getDataSchedule()

public DataSchedule getDataSchedule()

Returns a schedule of data events associated with this service.

Returns:

The data schedule for this service.

A.1.11.1.2 retrieveDataService(javax.tv.service.SIRequestor)

public javax.tv.service.SIRequest retrieveDataService(javax.tv.service.SIRequestor requestor)

This method retrieves one or more DataService objects with which this DataServiceDetails is associated.

Parameters:

requestor – The SIRequestor to be notified when this retrieval operation completes.

Returns:

An SIRequest object identifying this asynchronous retrieval request.

A.1.11.2 Fields

No fields are defined.
A.1.12 DataServiceListener

public interface DataServiceListener
    extends javax.tv.service.SIChangeListener

This interface is implemented by applications wishing to receive notification of changes to a DataService.

A.1.12.1 Methods

A.1.12.1.1 notifyChange(DataServiceChangeEvent)

public void notifyChange(DataServiceChangeEvent event)

Notifies the DataServiceListener of a change to a DataService.

Parameters:
- event – A DataServiceChangeEvent describing what changed and how.

A.1.12.2 Fields

No fields are defined.

A.2 org.atsc.si

This package provides ATSC specific extensions to the Java TV service information (SI) facilities.

A.2.1 AtscLocator

public interface AtscLocator
    extends MPEGLocator

Extends MPEGLocator to add ATSC specific features.

Any MPEGLocator which references a program event, data event, or service shall implement this AtscLocator interface.

A.2.1.1 Methods

The following methods are inherited MPEGLocator: getProgramNumber, getTransportStreamID.

The following methods are inherited from javax.tv.locator.Locator: equals, hashCode, hasMultipleTransformations, toExternalForm, toString.

A.2.1.1.1 getEventIdentifier()

public int getEventIdentifier()
    throws javax.tv.service.SIException

Retrieve the event identifier of the program or data event referenced by this locator.

If this locator references a program event, then the identifier returned shall be derived from the event_id field of an event information table (EIT); if it references a data event, then the identifier returned shall be derived from the data_id field of the data event table (DET) or long term service table (LTST).
Note: See [A/65], Section 6.5, and [A/90], Sections 11.3 and 11.7, for further information on event and data identifiers.

Returns:
A program or data event identifier.

Throws:
javax.tv.service.SIException – when the object referenced by this locator does not refer to an ATSC program or data event.

A.2.1.1.2 getSourceIdentifier()

public int getSourceIdentifier()
throws javax.tv.service.SIException

Retrieve the source identifier of the service referenced by this locator.

Note: See [A/65], Section 6.3, Virtual Channel Table (VCT), for further information on source identifiers.

Returns:
A source identifier.

Throws:
javax.tv.service.SIException – when the object referenced by this locator does not refer to a service.

A.2.1.2 Fields
No fields are defined.

A.2.2 AtscService

public interface AtscService
extends javax.tv.service.ServiceMinorNumber

Extends javax.tv.service.ServiceMinorNumber to provide ATSC specific service information.

A.2.2.1 Methods
The following methods are inherited from javax.tv.service.ServiceMinorNumber:
getMinorNumber.

The following methods are inherited from javax.tv.service.ServiceNumber:
getServiceNumber.

A.2.2.1.1 isHidden()

public boolean isHidden()

Determines whether this service is directly visible to the user or whether it is hidden. The determination of a service’s hidden characteristic is derived from the hidden field of the service’s entry in the virtual channel table.

Note: See [A/65], Section 6.3, Virtual Channel Table (VCT), for further information on the hidden field.
Returns:

The value \texttt{true} if this is a hidden service; otherwise, returns \texttt{false}.

\textbf{A.2.2.1.2 isVisible()}

\begin{verbatim}
public boolean isVisible()

Determines whether this service is intended to be displayed in a program guide. The
determination of a service’s visibility characteristic is derived from the hide_guide field of the
service’s entry in the virtual channel table.

Note: See [A/65], Section 6.3, \textit{Virtual Channel Table (VCT)}, for further information on the
hide_guide flag.

Returns:

The value \texttt{true} if this is a visible service; otherwise, returns \texttt{false}.
\end{verbatim}

\textbf{A.2.2 Fields}

No fields are defined.

\textbf{A.2.3 MPEGLocator}

\begin{verbatim}
public interface MPEGLocator
extends javax.tv.locator.Locator

The MPEGLocator interface provides an extension to \texttt{javax.tv.locator.Locator} in order
to specify functionality which pertains to MPEG-2 transport streams.
\end{verbatim}

\textbf{A.2.3.1 Methods}

The following methods are inherited from \texttt{javax.tv.locator.Locator}: equals, hashCode,
hasMultipleTransformations, toExternalForm, toString.

\textbf{A.2.3.1.1 getProgramNumber()}

\begin{verbatim}
public int getProgramNumber()
throws javax.tv.service.SIException

Retrieve the MPEG-2 program number of the service which this locator references.

Note: See [MPEG-2], Clause 2.4.4.8, \textit{Program Map Table}, for further information on an
MPEG-2 program number (\texttt{program_number}).

Returns:

An MPEG-2 program number.

Throws:

\texttt{javax.tv.service.SIException} – when the object referenced by this locator does not
refer to an MPEG-2 program.
\end{verbatim}

\textbf{A.2.3.1.2 getTransportStreamID()}

\begin{verbatim}
public int getTransportStreamID()
throws javax.tv.service.SIException
\end{verbatim}
Retrieve the MPEG-2 transport stream identifier of the service which this locator references.

Note: See [MPEG-2], Clause 2.4.4.3, Program Association Table, for further information on an MPEG-2 transport stream identifier (transport_stream_id).

Returns:
An MPEG-2 transport stream identifier.

Throws:
javax.tv.service.SIException – when the object referenced by this locator does not refer to an MPEG-2 transport stream.

A.2.3.2 Fields
No fields are defined.

A.2.4 NVODReference

public interface NVODReference

This interface provides additional information about a service which represents a near video on demand (NVOD) service.

If an object which implements javax.tv.service.navigation.ServiceDetails is an NVOD service, then it shall implement the NVODReference interface.

See also: javax.tv.service.navigation.ServiceDetails.

A.2.4.1 Methods

A.2.4.1.1 getShiftedServices()

public TimeShiftedService[] getShiftedServices()

Provides a list of time-shifted services which are referenced by this service. This list shall be ordered by time-shift value.

Returns:
An array of time shift channel descriptions.

A.2.4.2 Fields
No fields are defined.

A.2.5 TimeShiftedService

public interface TimeShiftedService

This interface provides information about a time-shifted service.

A.2.5.1 Methods

A.2.5.1.1 getService()

public javax.tv.service.Service getService()

Provides a reference to a time-shifted service.
Returns:

A time-shifted service object.

See also: NVODReference.

A.2.5.1.2 getTimeShift()

public long getTimeShift()

Specifies the time shift of this service with respect to the reference service.

Returns:

An integer representing number of seconds of the time shift.

A.2.5.2 Fields

No fields are defined.
ANNEX B. AUTOLOAD SCENARIOS

This annex is informative.

The following table indicates the scenarios in which a signaled DASE Application is or is not autoloaded by a DASE System. In this context, autoloading means automatic application activation without explicit end-user action. The first column specifies the value of the action_type field of the Tap which signals the DASE Application's root resource, where the value 0x00 denotes run-time data and the value 0x01 denotes bootstrap data. The second column specifies the computed value of the DASE Application's noautoload parameter. The third column specifies whether automatic activation (autoloading) occurs based on the values in the first two columns.

Table 10 Autoload Scenarios

<table>
<thead>
<tr>
<th>action_type field</th>
<th>noautoload parameter</th>
<th>automatic activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>0x00</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>0x01</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>0x01</td>
<td>true</td>
<td>no</td>
</tr>
</tbody>
</table>

Note: See [A/90], Table 12.6, for information on the action_type field. See [DASE], Section 6.1.1.6.13.5, for information on the noautoload parameter. See [DASE], Section 5.1.3.1.1, for information on application activation.
CHANGES

This section is informative.

Changes from Candidate Standard to Standard

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

Table 11 Changes from Candidate Standard

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Change status to standard.</td>
</tr>
<tr>
<td>3</td>
<td>Add definition of ARM data application.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Specify use of noautoload parameter with multiple signaled applications in single data service.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Specify use of acquisition directive with root entity.</td>
</tr>
<tr>
<td>4.1.3.2</td>
<td>Add use of modules for encapsulating bounded resources.</td>
</tr>
<tr>
<td>4.1.3.2</td>
<td>Constrain use of application delivery file system.</td>
</tr>
<tr>
<td>4.1.3.3</td>
<td>Specify auto mount semantics for application delivery file system.</td>
</tr>
<tr>
<td>4.1.3.3</td>
<td>Don't require MRD for DASE trigger streams.</td>
</tr>
<tr>
<td>4.1.3.3</td>
<td>Note difference in terminology regarding target.</td>
</tr>
<tr>
<td>4.1.3.3</td>
<td>Specify encapsulation structure for daseTrigger() in event_info::user_data_byte[].</td>
</tr>
<tr>
<td>4.2.1.2</td>
<td>Note carousel file change relationship with DASE-DA mutation events.</td>
</tr>
<tr>
<td>5.1.1.1</td>
<td>Constrain javax.tv.net.InterfaceMap.getLocalAddress(Locator) argument semantics.</td>
</tr>
<tr>
<td>A.1.9.1</td>
<td>Clarify semantics of DataServiceApplication.getResourceLocators.</td>
</tr>
</tbody>
</table>