DTV APPLICATION SOFTWARE ENVIRONMENT LEVEL 1 (DASE-1)
PART 4: APPLICATION PROGRAMMING INTERFACE

ATSC Standard

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

DASE-1 APPLICATION PROGRAMMING INTERFACE .............................................. 1

1. SCOPE ............................................................................................................. 1
   1.1 Status ........................................................................................................ 1
   1.2 Purpose ..................................................................................................... 1
   1.3 Application ............................................................................................... 1
   1.4 Organization ............................................................................................ 1

2. REFERENCES .................................................................................................... 3
   2.1 Normative References ............................................................................. 3
   2.2 Informative References .......................................................................... 3
   2.3 Reference Acquisition ............................................................................ 3

3. DEFINITIONS ................................................................................................... 5
   3.1 Conformance Keywords .......................................................................... 5
   3.2 Acronyms and Abbreviations .................................................................. 5
   3.3 Terms ......................................................................................................... 5

4. PACKAGES ....................................................................................................... 6
   4.1 org.atsc.application .................................................................................... 6
      4.1.1 ApplicationInformation ..................................................................... 6
   4.2 org.atsc.carousel ......................................................................................... 7
      4.2.1 CarouselException ............................................................................ 7
      4.2.2 DataDeliveryException ...................................................................... 8
      4.2.3 InsufficientResourceException ......................................................... 9
      4.2.4 InvalidFormatException ...................................................................... 9
      4.2.5 NotAuthorizedException .................................................................. 10
      4.2.6 TimeoutException ............................................................................. 11
   4.3 org.atsc.dom ............................................................................................... 11
      4.3.1 DocumentAction ............................................................................... 11
      4.3.2 DocumentFactory ............................................................................. 12
      4.3.3 DOMExceptionExt ............................................................................ 12
      4.3.4 MultipleDocumentsAction ............................................................... 14
   4.4 org.atsc.dom.environment ....................................................................... 14
      4.4.1 History ............................................................................................... 14
      4.4.2 Location ............................................................................................. 15
      4.4.3 Navigator ........................................................................................... 19
      4.4.4 Window .............................................................................................. 20
   4.5 org.atsc.dom.events ............................................................................... 26
      4.5.1 KeyEvent ........................................................................................... 26
      4.5.2 KeyModifiers ..................................................................................... 28
      4.5.3 VirtualKeys ........................................................................................ 29
   4.6 org.atsc.dom.html .................................................................................... 33
      4.6.1 HTMLAnchorElementExt ................................................................. 33
      4.6.2 HTMLDocumentExt .......................................................................... 36
      4.6.3 HTMLFormElementExt ..................................................................... 39
      4.6.4 HTMLImageElementExt .................................................................. 40
      4.6.5 HTMLObjectElementExt .................................................................. 40
   4.7 org.atsc.dom.views ................................................................................... 42
      4.7.1 DocumentViewExt ............................................................................ 42
   4.8 org.atsc.graphics ....................................................................................... 46
      4.8.1 AtscBufferedImage ........................................................................... 46
      4.8.2 FontFactory ....................................................................................... 50
      4.8.3 FontFormatException ......................................................................... 51
   4.9 org.atsc.management ............................................................................... 51
      4.9.1 AdministrativeState ......................................................................... 51
      4.9.2 AlarmStatus ....................................................................................... 52
      4.9.3 AvailabilityStatus ............................................................................. 54
### org.atsc.net
- **4.10** org.atsc.net
- **4.10.1** DatagramSocketBufferControl

### org.atsc.preferences
- **4.11** org.atsc.preferences
- **4.11.1** FavoriteChannelsPreference
- **4.11.2** InvalidPreferenceException
- **4.11.3** LanguagePreference
- **4.11.4** LanguageScope
- **4.11.5** PersonalDataPreference
- **4.11.6** Preference...
- **4.11.7** PreferenceChangeEvent
- **4.11.8** PreferenceChangeListener
- **4.11.9** PreferenceChangeListener
- **4.11.10** PreferenceNames
- **4.11.11** PreferencePermission
- **4.11.12** PreferenceRegistry
- **4.11.13** PreferenceRegistryEvent
- **4.11.14** RatingPreference

### org.atsc.registry
- **4.12** org.atsc.registry
- **4.12.1** Registry
- **4.12.2** RegistryChangeCause
- **4.12.3** RegistryChangeEvent
- **4.12.4** RegistryChangeListener
- **4.12.5** RegistryFactory
- **4.12.6** RegistryType

### org.atsc.security
- **4.13** org.atsc.security
- **4.13.1** AccessDeniedException
- **4.13.2** AtscAllPermission
- **4.13.3** AtscPermission
- **4.13.4** HAVIPermission

### org.atsc.system
- **4.14** org.atsc.system
- **4.14.1** Receiver
- **4.14.2** ReceiverPropertyNames

### org.atsc.trigger
- **4.15** org.atsc.trigger
- **4.15.1** TriggerEvent
- **4.15.2** TriggerListener
- **4.15.3** TriggerSource

### org.atsc.user
- **4.16** org.atsc.user
- **4.16.1** InvalidCapabilityException
- **4.16.2** InvalidUserException
- **4.16.3** UserCapabilities
- **4.16.4** UserChangeCause
- **4.16.5** UserPermission
- **4.16.6** UserProfile
- **4.16.7** UserRegistry
- **4.16.8** UserRegistryEvent

### org.atsc.xlet
- **4.17** org.atsc.xlet
4.17.1 InvalidXletException .......................................................................................... 103
4.17.2 XletAlreadyRegisteredException .................................................................. 104
4.17.3 XletAvailabilityException ............................................................................. 104
4.17.4 XletChangeCause ......................................................................................... 105
4.17.5 XletComponentPresenterProxy ..................................................................... 106
4.17.6 XletContextExt ........................................................................................... 107
4.17.7 XletInformation ........................................................................................... 108
4.17.8 XletNotRegisteredException ........................................................................ 109
4.17.9 XletPermission .......................................................................................... 110
4.17.10 XletProxy ................................................................................................ 111
4.17.11 XletRegistry .............................................................................................. 113
4.17.12 XletRegistryEvent ..................................................................................... 116

CHANGES ....................................................................................................................118
Changes from Candidate Standard to Standard ............................................................. 118

Table of Tables

Table 1 HAViPermission Targets .............................................................................. 87
Table 2 Changes from Candidate Standard ............................................................... 118
Blank Page
DASE-1 Application Programming Interface
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/100-4) 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 a collection of Java packages which defines DASE specific application programming interfaces (APIs) provided by a procedural application environment to DASE Applications.\(^1\)

1.3 Application

The facilities of this specification are intended to apply to terrestrial (over-the-air) broadcast systems and receivers. In addition, the same 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 interface definitions
- 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. 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.
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).

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

[LANG-TAGS]
Tags for the Identification of Languages, RFC3066, IETF

[X.731]

2.2 Informative References

[DASE-DA]
DASE-1 Part 2: Declarative Applications and Environment, A/100-2, ATSC

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

[HAVI-UI-API]

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/.

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/.
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/.

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/.
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

None defined.

3.3 Terms

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.

Note: See [DASE-PA], Section 5.1.1.2.3. Java Television (Java TV) Interfaces, for more information on Java TV related functionality.
4. PACKAGES

This section describes normative definitions of DASE specific Java APIs in the form of a collection of Java packages.

4.1 org.atsc.application

This package includes classes and interfaces related to functionality regarding a DASE Application.

Note: A DASE Application may make use of multiple Xlets. In general, an Xlet needs to correspond one to one with a DASE Application. See the org.atsc.xlet package for functionality that pertains to Xlets as used with DASE Applications.

4.1.1 ApplicationInformation

public interface ApplicationInformation

This interface provides additional information about a DASE Application.

4.1.1.1 Methods

4.1.1.1.1 getDescription()

public java.lang.String getDescription()

Returns a description of the application which can be displayed to the user.

Returns:

A string representing a description of the application. It shall return an empty string if no such information is available.

Note: See [DASE], Section 6.1.1.6.11, for information on how an application’s name is specified.

4.1.1.2 getIdentifier()

public java.lang.String getIdentifier()

Returns a unique identifier for this application. The identifier of an application shall be the value of the uuid attribute of the identifier element of an application’s metadata resource.

Returns:

A string uniquely identifying application.

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

4.1.1.3 getLocator()

public javax.tv.locator.Locator getLocator()

Returns the locator which references the application’s root resource.

Returns:

A locator referencing this application.
4.1.1.1.4  **getName()**

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

Returns a name of the application which can be displayed to the user.

Returns:

A string representing a name of the application. It shall return an empty string if no such information is available.

*Note:* See [DASE], Section 6.1.1.6.11, for information on how an application’s name is specified.

4.1.1.1.5  **getRequiredLevel()**

```java
public int getRequiredLevel()
```

Returns the minimum DASE level that is expected by this application in order to run.

Returns:

A number representing the DASE level required by this application. The value zero indicates that no level was specified.

*Note:* See [DASE], Section 6.1.1.6.4.1.5.1, *level* parameter, for more information on how an application’s required level is specified in an application’s metadata resource.

4.1.1.2  **Fields**

No fields are defined.

4.2  **org.atsc.carousel**

This package defines exceptions raised when working with carousel file functionality as specified by the `javax.tv.carousel` package.

Note that `java.io.FilePermission` is applicable to the `CarouselFile` security protected methods.

See also: `javax.tv.carousel`.

4.2.1  **CarouselException**

```java
public class CarouselException
    extends java.io.IOException
```

`CarouselException` is the base class for the exceptions related to carousel file operations.

4.2.1.1  **Constructors**

4.2.1.1.1  **CarouselException()**

```java
public CarouselException()
```

Construct a `CarouselException` with no detail message.

4.2.1.1.2  **CarouselException(java.lang.String)**

```java
public CarouselException(java.lang.String s)
```

Construct a `CarouselException` with the specified detail message.
Parameters:

s – the detail message

4.2.1.2 Methods

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

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

4.2.1.3 Fields

No fields are defined.

4.2.2 DataDeliveryException

public class DataDeliveryException
extends CarouselException

A DataDeliveryException exception is thrown when an error occurs while loading data through mechanisms defined by the application delivery system.

This exception is related to the carousel file operation. This exception may be raised by the access methods of the java.io package’s FileInputStream, RandomAccessFile, and FileReader classes.

4.2.2.1 Constructors

4.2.2.1.1 DataDeliveryException()

public DataDeliveryException()

Construct a DataDeliveryException with no detail message.

4.2.2.1.2 DataDeliveryException(java.lang.String)

public DataDeliveryException(java.lang.String s)

Construct a DataDeliveryException with the specified detail message.

Parameters:

s – the detail message

4.2.2.2 Methods

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

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

4.2.2.3 Fields

No fields are defined.
4.2.3  InsufficientResourceException

public class InsufficientResourceException
    extends CarouselException

An InsufficientResourceException exception is thrown when a carousel operation requires resources not available at the time of the call.

Note: This exception may be thrown during the process of constructing an instance of javax.tv.carousel.CarouselFile in the case that some resource which is required to access the carousel file is not available and cannot be obtained at construct time.

4.2.3.1  Constructors

4.2.3.1.1  InsufficientResourceException()

public InsufficientResourceException()
    Construct an InsufficientResourceException with no detail message.

4.2.3.1.2  InsufficientResourceException(java.lang.String)

public InsufficientResourceException(java.lang.String s)
    Construct an InsufficientResourceException with the specified detail message.
    Parameters:
    s  – the detail message

4.2.3.2  Methods

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

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

4.2.3.3  Fields

No fields are defined.

4.2.4  InvalidFormatException

public class InvalidFormatException
    extends CarouselException

An InvalidFormatException is thrown when an inconsistent carousel state condition is detected during a carousel file operation.

4.2.4.1  Constructors

4.2.4.1.1  InvalidFormatException()

public InvalidFormatException()
    Construct an InvalidFormatException with no detail message.
4.2.4.1.2 InvalidFormatException(java.lang.String)

    public InvalidFormatException(java.lang.String s)

    Construct an InvalidFormatException with the specified detail message.

    Parameters:
    s – the detail message

4.2.4.2 Methods

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

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

4.2.4.3 Fields

    No fields are defined.

4.2.5 NotAuthorizedException

    public class NotAuthorizedException
        extends org.atsc.security.AccessDeniedException

    An exception is thrown when the user is not entitled to access carousel file object.

4.2.5.1 Constructors

4.2.5.1.1 NotAuthorizedException()

    public NotAuthorizedException()

    Construct a NotAuthorizedException with no detail message.

4.2.5.1.2 NotAuthorizedException(java.lang.String)

    public NotAuthorizedException(java.lang.String s)

    Construct a NotAuthorizedException with the specified detail message.

    Parameters:
    s – the detail message

4.2.5.2 Methods

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

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

4.2.5.3 Fields

    No fields are defined.
4.2.6 TimeoutException

```java
class TimeoutException
    extends DataDeliveryException
```

A `TimeoutException` exception is thrown when a time-out condition occurs while loading a carousel file object.

4.2.6.1 Constructors

4.2.6.1.1 `TimeoutException()`

```java
public TimeoutException()
```

Construct a `TimeoutException` with no detail message.

4.2.6.1.2 `TimeoutException(java.lang.String)`

```java
public TimeoutException(java.lang.String s)
```

Construct a `TimeoutException` with the specified detail message.

Parameters:
- `s` – the detail message

4.2.6.2 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)`.

4.2.6.3 Fields

No fields are defined.

4.3 `org.atsc.dom`

Specifies DOM bootstrap interfaces and extensions to W3C DOM Level 2 Core interfaces.

4.3.1 DocumentAction

```java
public interface DocumentAction
```

`DocumentAction` is used for an action that modifies a W3C `Document`. When an application wishes to access a `Document`, it creates an instance of a class that implements `DocumentAction`, and passes this instance to the system. The system then calls back to the user code via the `DocumentAction` interface.

See also: `MultipleDocumentsAction`, `DocumentFactory`.

4.3.1.1 Methods

4.3.1.1.1 `run(org.w3c.dom.Document)`

```java
public void run(org.w3c.dom.Document doc)
```
Access and/or modify a W3C DOM Document. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

4.3.1.2 Fields
No fields are defined.

4.3.2 DocumentFactory

public interface DocumentFactory

DocumentFactory contains bootstrap methods for applications to access a W3C Document.

See also: DocumentAction, MultipleDocumentsAction.

4.3.2.1 Methods

4.3.2.1.1 performAction(DocumentAction)

public void performAction(DocumentAction act)

Perform an action on the document that contains the Xlet controlled by the XletContext used to access this DocumentFactory.

Parameters:

act – The action to perform. It will be called by the system either synchronously, or on a system thread.

4.3.2.1.2 performActionOnFrames(java.lang.String[], MultipleDocumentsAction)

public void

performActionOnFrames(java.lang.String[] names, MultipleDocumentsAction act)

Perform an action on a set of documents, each contained in a frame that is a part of the same application as the Xlet controlled by the XletContext used to access this DocumentFactory.

Parameters:

names – The names of the desired frames.

act – The action to perform. It will be called by the system either synchronously, or on a system thread.

4.3.2.2 Fields

4.3.2.2.1 DOM

public static final java.lang.String DOM = "DOM"

The property string to use with XletContext.getXletProperty in order to obtain the DocumentFactory for this Xlet (if one exists).

4.3.3 DOMExceptionExt

Provides functionality for creating and receiving DOM exceptions.
Note: See [DASE-DA], Section 5.3.1.2.1.3, for further information on the semantics of this type.

4.3.3.1 Constructors

4.3.3.1.1 DOMExceptionExt(short, java.lang.String)

public DOMExceptionExt(short codeExt, java.lang.String s)

Construct a DOMExceptionExt with a detail message.

Parameters:

codeExt – the extended DOM exception code.

s – the detail message.

4.3.3.2 Methods

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

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

4.3.3.2.1 getCodeExtension()

public int getCodeExtension()

Retrieves the DOM exception code extension value.

Note: See [DASE-DA], Section 5.3.1.2.1.3, for further information on the semantics of this method.

Returns:

An integer denoting the DOM exception code extension.

4.3.3.3 Fields

The following fields are inherited from org.w3c.dom.DOMException: code, DOMSTRING_SIZE_ERR, HIERARCHY_REQUEST_ERR, INDEX_SIZE_ERR, INUSE_ATTRIBUTE_ERR, INVALID_ACCESS_ERR, INVALID_CHARACTER_ERR, INVALID_MODIFICATION_ERR, INVALID_STATE_ERR, NAMESPACE_ERR, NO_DATA_ALLOWED_ERR, NO_MODIFICATION_ALLOWED_ERR, NOT_FOUND_ERR, NOT_SUPPORTED_ERR, SYNTAX_ERR, WRONG_DOCUMENT_ERR.

4.3.3.3.1 NO_CLOSE_ALLOWED_ERR

public static final short NO_CLOSE_ALLOWED_ERR = 2

Indicates a window close operation is not allowed.

4.3.3.3.2 VALIDATION_ERR

public static final short VALIDATION_ERR = 1

Indicates a validation error occurred.
### 4.3.4 MultipleDocumentsAction

*MultipleDocumentAction* is used for an action that modifies zero or more W3C Document instances. When an application wishes to access a number of Documents simultaneously, it creates an instance of a class that implements *MultipleDocumentAction*, and passes this instance to the system. The system then calls back to the user code via the *MultipleDocumentAction* interface.

See also: DocumentAction, DocumentFactory.

#### 4.3.4.1 Methods

##### 4.3.4.1.1 run(org.w3c.dom.Document[])  

```java
public void run(org.w3c.dom.Document[] docs)
```

Access and/or modify a set of W3C DOM Documents. All modifications must take place during this callback. The results of retaining and using DOM references outside the context of this callback are undefined.

Parameters:

- **docs** – The array of documents on which to operate. If a requested document is not found, the corresponding entry in this array will be null.

#### 4.3.4.2 Fields

No fields are defined.

### 4.4 org.atsc.dom.environment

Specifies interfaces to environment DOM functionality.

#### 4.4.1 History

```java
public interface History
```

Provides functionality for accessing the browsing history of the window.

*Note:* See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this type.

#### 4.4.1.1 Methods

##### 4.4.1.1.1 back()  

```java
public void back()
```

Navigate to the document referenced by the previous history entry.

*Note:* See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this method.

##### 4.4.1.1.2 forward()  

```java
public void forward()
```

Navigate to the document referenced by the next history entry.
Note: See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this method.

4.4.1.3 getLength()

public long getLength()

Retrieve the number of history entries.

Note: See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this method.

Returns:

The number of history entries.

4.4.1.4 go(long)

public void go(long index)

Navigate to the document referenced by the specified history entry.

Note: See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this method.

Parameters:

index – history entry index.

4.4.1.5 go(java.lang.String)

public void go(java.lang.String uri)

Navigate to the document referenced by the specified URI.

Note: See [DASE-DA], Section 5.3.1.2.9.1, for further information on the semantics of this method.

Parameters:

uri – URI of document to load.

4.4.1.2 Fields

No fields are defined.

4.4.2 Location

public interface Location

Provides functionality for accessing the location of the document presently loaded into the window.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this type.

4.4.2.1 Methods

4.4.2.1.1 getHash()

public java.lang.String getHash()

This method retrieves the fragment component of the URI referenced by this location.
Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:
A string denoting the fragment component.

4.4.2.1.2 getHost()

public java.lang.String getHost()

This method retrieves the hostport component of the URI referenced by this location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:
A string denoting the hostport component.

4.4.2.1.3 getHostname()

public java.lang.String getHostname()

This method retrieves the host component of the URI referenced by this location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:
A string denoting the host component.

4.4.2.1.4 getHref()

public java.lang.String getHref()

This method retrieves a string representation of the URI referenced by this location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:
A string denoting the location’s URI.

4.4.2.1.5 getPathname()

public java.lang.String getPathname()

This method retrieves the abs_path or rel_path component of the URI referenced by this location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:
A string denoting the abs_path or rel_path component.

4.4.2.1.6 getPort()

public java.lang.String getPort()

This method retrieves the port component of the URI referenced by this location.
**Note:** See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:

A string denoting the *port* component.

### 4.4.2.1.7 `getProtocol()`

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

This method retrieves the *scheme* component of the URI referenced by this location.

**Note:** See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:

A string denoting the *scheme* component.

### 4.4.2.1.8 `getSearch()`

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

This method retrieves the *query* component of the URI referenced by this location.

**Note:** See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Returns:

A string denoting the *query* component.

### 4.4.2.1.9 `setHash(java.lang.String)`

```java
public void setHash(java.lang.String hash)
```

This method sets the *fragment* component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

**Note:** See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

`hash` — string denoting the new *fragment* component value.

### 4.4.2.1.10 `setHost(java.lang.String)`

```java
public void setHost(java.lang.String host)
```

This method sets the *hostport* component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

**Note:** See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

`host` — string denoting the new *hostport* component value.
4.4.2.1.11  setHostname(java.lang.String)

public void setHostname(java.lang.String hostname)

This method sets the host component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

hostname – string denoting the new host component value.

4.4.2.1.12  setHref(java.lang.String)

public void setHref(java.lang.String href)

This method sets the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

href – string denoting the new URI.

4.4.2.1.13  setPathname(java.lang.String)

public void setPathname(java.lang.String pathname)

This method sets the abs_path or rel_path component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

pathname – string denoting the new abs_path or rel_path component value.

4.4.2.1.14  setPort(java.lang.String)

public void setPort(java.lang.String port)

This method sets the port component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

port – string denoting the new port component value.
4.4.2.1.15 setProtocol(java.lang.String)

public void setProtocol(java.lang.String protocol)

This method sets the scheme component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

scheme – string denoting the new scheme component value.

4.4.2.1.16 setSearch(java.lang.String)

public void setSearch(java.lang.String search)

This method sets the query component of the URI referenced by this location.

Invocation of this method will generally produce a side effect of causing navigation to the resource referenced by the URI representing the new value of the location.

Note: See [DASE-DA], Section 5.3.1.2.9.2, for further information on the semantics of this method.

Parameters:

query – string denoting the new query component value.

4.4.2.2 Fields

No fields are defined.

4.4.3 Navigator

public interface Navigator

Provides functionality for accessing the document navigator.

Note: See [DASE-DA], Section 5.3.1.2.9.3, for further information on the semantics of this type.

4.4.3.1 Methods

4.4.3.1.1 getAppCodeName()

public java.lang.String getAppCodeName()

Retrieves a vendor specific code name associated with the navigator implementation.

Note: See [DASE-DA], Section 5.3.1.2.9.3, for further information on the semantics of this method.

Returns:

A string denoting the navigator’s code name.

4.4.3.1.2 getAppName()

public java.lang.String getAppName()
Retrieves a vendor specific name associated with the navigator implementation.

*Note:* See [DASE-DA], Section 5.3.1.2.9.3, for further information on the semantics of this method.

**Returns:**

A string denoting the navigator’s name.

### 4.4.3.1.3 `getAppVersion()`

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

Retrieves a vendor specific version associated with the navigator implementation.

*Note:* See [DASE-DA], Section 5.3.1.2.9.3, for further information on the semantics of this method.

**Returns:**

A string denoting the navigator’s version.

### 4.4.3.1.4 `getUserAgent()`

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

Retrieves the vendor specific user agent tokens associated with the navigator implementation.

*Note:* See [DASE-DA], Section 5.3.1.2.9.3, for further information on the semantics of this method.

**Returns:**

A string denoting the navigator’s user agent tokens.

### 4.4.3.2 Fields

No fields are defined.

### 4.4.4 Window

```java
public interface Window
```

Provides functionality for accessing a window in which declarative application markup content is being presented.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this type.

### 4.4.4.1 Methods

#### 4.4.4.1.1 `alert(java.lang.String)`

```java
public void alert(java.lang.String message)
```

Present an alert dialog to the end-user.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.
Parameters:

`message` – message to display in dialog.

### 4.4.4.1.2 clearTimeout(long timerId)

```java
public void clearTimeout(long timerId)
```

Clear a scheduled statement timer.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

`timerId` – identifier of timer to be cleared.

### 4.4.4.1.3 close()

```java
public void close()
```

Close window if it is a top-level window, possibly terminating the application.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

### 4.4.4.1.4 confirm(java.lang.String)

```java
public boolean confirm(java.lang.String message)
```

Present a confirmation dialog to the end-user.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

`message` – message to display in dialog.

Returns:

A boolean denoting if the end-user selected OK or CANCEL.

### 4.4.4.1.5 getDefaultStatus()

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

Retrieves the window’s default status string.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:

A string denoting the window’s default status.

### 4.4.4.1.6 getDocument()

```java
public org.w3c.dom.html2.HTMLDocument getDocument()
```

Retrieves the document instance associated with the window.

*Note:* See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.
Returns:
A document instance.

4.4.4.1.7 getFrames()
public org.w3c.dom.html2.HTMLCollection getFrames()
Retrieves the collection of children windows for a frameset window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A collection of children windows (frames).

4.4.4.1.8 getHistory()
public History getHistory()
Retrieves the history instance associated with the window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A history instance.

4.4.4.1.9 getLength()
public long getLength()
Retrieves the number of children windows for a frameset window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
The number of children windows (frames).

4.4.4.1.10 getLocation()
public Location getLocation()
Retrieves the location instance associated with the window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A location instance.

4.4.4.1.11 getName()
public java.lang.String getName()
Retrieves the window’s name.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.
Returns:
A string denoting the window’s name.

4.4.4.1.12 getNavigator()
public Navigator getNavigator()
Retrieves the navigator instance.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A navigator instance.

4.4.4.1.13 getOpener()
public Window getOpener()
Retrieves the window instance responsible for opening this window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A window instance or null.

4.4.4.1.14 getParent()
public Window getParent()
Retrieves the parent window instance or the current window instance in the case that this window is a top-level window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A window instance.

4.4.4.1.15 getSelf()
public Window getSelf()
Retrieves the current window instance.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A window instance.

4.4.4.1.16 getStatus()
public java.lang.String getStatus()
Retrieves the window’s status string.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.
Returns:
A string denoting the window’s status.

4.4.4.1.17 getTop()

public Window getTop()

Retrieves the top-level window instance or the current window instance in the case that this window is a top-level window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A window instance.

4.4.4.1.18 getWindow()

public Window getWindow()

Retrieves the current window instance.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Returns:
A window instance.

4.4.4.1.19 open(java.lang.String, java.lang.String, java.lang.String)

public Window
open(java.lang.String uri, java.lang.String name, java.lang.String features)

Opens a new window or cause navigation in existing window.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

uri – URI to load into window.
name – name of new or existing window.
features – window features.

Returns:
The new or existing window instance.

4.4.4.1.20 prompt(java.lang.String, java.lang.String)

public java.lang.String
prompt(java.lang.String message, java.lang.String defaultResponse)

Present a prompt dialog to the end-user.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

message – message to display in dialog.
defaultResponse – default response.

Returns:

A string representing the end-user response or the default response.

4.4.4.1.21 setDefaultStatus(java.lang.String)

defaultResponse – default response.

Returns:

A string representing the end-user response or the default response.

4.4.4.1.22 setLocation(java.lang.String)

Sets the window’s default status string.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

defaultStatus – default status string.

4.4.4.1.23 setOpener(Window)

Sets the window’s opener.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

window – window to treat as opener or null.

4.4.4.1.24 setStatus(java.lang.String)

Sets the window’s status string.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.

Parameters:

status – status string.

4.4.4.1.25 setTimeout(java.lang.String, long)

Schedule a statement timer.

Note: See [DASE-DA], Section 5.3.1.2.9.4, for further information on the semantics of this method.
Parameters:

statement – statement to be scheduled.
delay – number of milliseconds to delay before statement evaluation.

Returns:

An identifier for the scheduled statement timer.

4.4.4.2 Fields

No fields are defined.

4.5 org.atsc.dom.events

Specifies extensions to W3C DOM Level 2 Events functionality.

4.5.1 KeyEvent

public interface KeyEvent extends org.w3c.dom.events.UIEvent

Provides functionality for being notified of key events.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this type.

4.5.1.1 Methods

4.5.1.1.1 checkModifier(int)

public void initModifier(int modifier, boolean value)

This method retrieves the value of a key event modifier.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method.

Parameters:

modifier – a key modifier.

Returns:

The value of the specified modifier.

4.5.1.1.2 getKeyVal()

public int getKeyVal()

This method retrieves zero or a Unicode character associated with the depressed key.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method.

Returns:

Either zero or an integer denoting a Unicode character.

4.5.1.1.3 getNumPad()

public boolean getNumPad()
This method retrieves an indication of whether the depressed key was on the number pad.

| Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method. |

| Returns: |
| An boolean true value if key is on number pad. |

4.5.1.1.4 getOutputString()

public java.lang.String getOutputString()

This method retrieves the output string associated with the key event.

| Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method. |

| Returns: |
| An string denoting the key event text output. |

4.5.1.1.5 getVirtKeyVal()

public int getVirtKeyVal()

This method retrieves VirtualKeys.VK_UNDEFINED or a virtual key value associated with the depressed key.

| Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method. |

| Returns: |
| An integer representing one of the fields defined by VirtualKeys. |

4.5.1.1.6 getVisibleOutputGenerated()

public boolean getVisibleOutputGenerated()

This method retrieves an indication of whether the depressed key resulted in visible output.

| Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method. |

| Returns: |
| An boolean true value if visible output was generated. |

4.5.1.1.7 initKeyEvent(java.lang.String,boolean,boolean,org.w3c.dom.views.AbstractView,-int,java.lang.String,int,int,boolean,boolean)

public void initKeyEvent (|
  java.lang.String type, |
  boolean canBubble, |
  boolean cancelable, |
  org.w3c.dom.views.AbstractView view, |
  int detail, |
  java.lang.String outputString, |
  int keyVal, |
  int virtKeyVal, |
  boolean visibleOutputGenerated, |
boolean numPad

This method initializes a key event.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method.

Parameters:

type – event type.

canBubble – indication of whether event can bubble.

cancelable – indication of whether event is cancelable.

view – view from which event is generated.

detail – number of key presses, if available.

outputString – generated text output.

keyVal – key value as Unicode character.

virtKeyVal – virtual key value..

visibleOutputGenerated – indication of whether visible output was generated.

numPad – indication of whether event was generated by number pad.

4.5.1.1.8 initModifier(int,boolean)

public void initModifier(int modifier, boolean value)

This method initializes a key event modifier.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this method.

Parameters:

modifier – a key modifier.

value – value to initialize modifier.

4.5.1.2 Fields

No fields are defined.

4.5.2 KeyModifiers

public interface KeyModifiers

Provides key event modifier constants.

Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this type.

4.5.2.1 Methods

No methods are defined.
4.5.2.2 Fields

4.5.2.2.1 MOD_NONE

    public static final int MOD_NONE = 0

    No modifier applies.

4.5.2.2.2 MOD_SHIFT

    public static final int MOD_NONE = 1

    Shift modifier active.

4.5.2.2.3 MOD_CONTROL

    public static final int MOD_CONTROL = 2

    Control modifier active.

4.5.2.2.4 MOD_META

    public static final int MOD_META = 4

    Meta modifier active.

4.5.2.2.5 MOD_ALT

    public static final int MOD_ALT = 8

    Alt modifier active.

4.5.3 VirtualKeys

    public interface VirtualKeys

    Provides virtual key event constants.

    Note: See [DASE-DA], Section 5.3.1.2.8.1, for further information on the semantics of this type.

4.5.3.1 Methods

    No methods are defined.

4.5.3.2 Fields

4.5.3.2.1 VK_UNDEFINED

    public static final int VK_UNDEFINED = 0

    Virtual key is not defined.

4.5.3.2.2 VK_* (Defined Virtual Keys)

    public static final int VK_CANCEL = 3
    public static final int VK_BACK_SPACE = 8
    public static final int VK_TAB = 9
    public static final int VK_ENTER = 10
    public static final int VK_CLEAR = 12
    public static final int VK_SHIFT = 16
public static final int VK_CONTROL = 17
public static final int VK_ALT = 18
public static final int VK_PAUSE = 19
public static final int VK_CAPS_LOCK = 20
public static final int VK_KANA = 21
public static final int VK_FINAL = 24
public static final int VK_KANJI = 25
public static final int VK_ESCAPE = 27
public static final int VK_CONVERT = 28
public static final int VK_NONCONVERT = 29
public static final int VK_ACCEPT = 30
public static final int VK_MODECHANGE = 31
public static final int VK_SPACE = 32
public static final int VK_PAGE_UP = 33
public static final int VK_PAGE_DOWN = 34
public static final int VK_END = 35
public static final int VK_HOME = 36
public static final int VK_LEFT = 37
public static final int VK_UP = 38
public static final int VK_RIGHT = 39
public static final int VK_DOWN = 40
public static final int VK_COMMA = 44
public static final int VK_PERIOD = 46
public static final int VK_SLASH = 47
public static final int VK_0 = 48
public static final int VK_1 = 49
public static final int VK_2 = 50
public static final int VK_3 = 51
public static final int VK_4 = 52
public static final int VK_5 = 53
public static final int VK_6 = 54
public static final int VK_7 = 55
public static final int VK_8 = 56
public static final int VK_9 = 57
public static final int VK_SEMICOLON = 59
public static final int VK_EQUALS = 61
public static final int VK_A = 65
public static final int VK_B = 66
public static final int VK_C = 67
public static final int VK_D = 68
public static final int VK_E = 69
public static final int VK_F = 70
public static final int VK_G = 71
public static final int VK_H = 72
public static final int VK_I = 73
public static final int VK_J = 74
public static final int VK_K = 75
public static final int VK_L = 76
public static final int VK_M = 77
public static final int VK_N = 78
public static final int VK_O = 79
public static final int VK_P = 80
public static final int VK_Q = 81
public static final int VK_R = 82
public static final int VK_S = 83
public static final int VK_T = 84
public static final int VK_U = 85
public static final int VK_V = 86
public static final int VK_W = 87
public static final int VK_X = 88
public static final int VK_Y = 89
public static final int VK_Z = 90
public static final int VK_OPEN_BRACKET = 91
public static final int VK_BACK_SLASH = 92
public static final int VK_CLOSE_BRACKET = 93
public static final int VK_NUMPAD0 = 96
public static final int VK_NUMPAD1 = 97
public static final int VK_NUMPAD2 = 98
public static final int VK_NUMPAD3 = 99
public static final int VK_NUMPAD4 = 100
public static final int VK_NUMPAD5 = 101
public static final int VK_NUMPAD6 = 102
public static final int VK_NUMPAD7 = 103
public static final int VK_NUMPAD8 = 104
public static final int VK_NUMPAD9 = 105
public static final int VK_MULTIPLY = 106
public static final int VK_ADD = 107
public static final int VK_SEPARATER = 108
public static final int VK_SUBTRACT = 109
public static final int VK_DECIMAL = 110
public static final int VK_DIVIDE = 111
public static final int VK_F1 = 112
public static final int VK_F2 = 113
public static final int VK_F3 = 114
public static final int VK_F4 = 115
public static final int VK_F5 = 116
public static final int VK_F6 = 117
public static final int VK_F7 = 118
public static final int VK_F8 = 119
public static final int VK_F9 = 120
public static final int VK_F10 = 121
public static final int VK_F11 = 122
public static final int VK_F12 = 123
public static final int VK_DELETE = 127
public static final int VK_NUM_LOCK = 144
public static final int VK_SCROLL_LOCK = 145
public static final int VK_PRINTSCREEN = 154
public static final int VK_INSERT = 155
public static final int VK_HELP = 156
public static final int VK_META = 157
public static final int VK_BACK_QUOTE = 192
public static final int VK_QUOTE = 222
public static final int VK_COLORED_KEY_0 = 403
public static final int VK_COLORED_KEY_1 = 404
public static final int VK_COLORED_KEY_2 = 405
public static final int VK_COLORED_KEY_3 = 406
public static final int VK_COLORED_KEY_4 = 407
public static final int VK_COLORED_KEY_5 = 408
public static final int VK_POWER = 409
public static final int VK_DIMMER = 410
public static final int VK_WINK = 411
public static final int VK_REWIND = 412
public static final int VK_STOP = 413
public static final int VK_EJECT_TOGGLE = 414
public static final int VK_PLAY = 415
public static final int VK_RECORD = 416
public static final int VK_FAST_FWD = 417
public static final int VK_PLAY_SPEED_UP = 418
public static final int VK_PLAY_SPEED_DOWN = 419
public static final int VK_PLAY_SPEED_RESET = 420
public static final int VK_RECORD_SPEED_NEXT = 421
public static final int VK_GO_TO_START = 422
public static final int VK_GO_TO_END = 423
public static final int VK_TRACK_PREV = 424
public static final int VK_TRACK_NEXT = 425
public static final int VK_RANDOM_TOGGLE = 426
public static final int VK_CHANNEL_UP = 427
public static final int VK_CHANNEL_DOWN = 428
public static final int VK_STORE_FAVORITE_0 = 429
public static final int VK_STORE_FAVORITE_1 = 430
public static final int VK_STORE_FAVORITE_2 = 431
public static final int VK_STORE_FAVORITE_3 = 432
public static final int VK_RECALL_FAVORITE_0 = 433
public static final int VK_RECALL_FAVORITE_1 = 434
public static final int VK_RECALL_FAVORITE_2 = 435
public static final int VK_RECALL_FAVORITE_3 = 436
public static final int VK_CLEAR_FAVORITE_0 = 437
public static final int VK_CLEAR_FAVORITE_1 = 438
public static final int VK_CLEAR_FAVORITE_2 = 439
public static final int VK_CLEAR_FAVORITE_3 = 440
public static final int VK_SCAN_CHANNELS_TOGGLE = 441
public static final int VK_PINP_TOGGLE = 442
public static final int VK_SPLIT_SCREEN_TOGGLE = 443
public static final int VK_DISPLAY_SWAP = 444
public static final int VK_SCREEN_MODE_NEXT = 445
public static final int VK_VIDEO_MODE_NEXT = 446
public static final int VK_VOLUME_UP = 447
public static final int VK_VOLUME_DOWN = 448
public static final int VK_MUTE = 449
public static final int VK_SURROUND_MODE_NEXT = 450
public static final int VK_BALANCE_RIGHT = 451
public static final int VK_BALANCE_LEFT = 452
public static final int VK_FADER_FRONT = 453
public static final int VK_FADER_REAR = 454
public static final int VK_BASS_BOOST_UP = 455
public static final int VK_BASS_BOOST_DOWN = 456
public static final int VK_INFO = 457
public static final int VK_GUIDE = 458
public static final int VK_TELETEXT = 459
public static final int VK_SUBTITLE = 460

Virtual key values.
4.6  
**org.atsc.dom.html**

Specifies extensions to W3C DOM Level 2 HTML interfaces.

4.6.1  
**HTMLAnchorElementExt**

    public interface HTMLAnchorElementExt

    Provides functionality for accessing an a (anchor) element.

    An object which implements org.w3c.dom.html2.HTMLAnchorElement shall also implement this interface.

    **Note:** See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this type.

4.6.1.1  
**Methods**

4.6.1.1.1  
**getHash()**

    public java.lang.String getHash()

    This method retrieves the fragment component of the URI referenced by this anchor element.

    **Note:** See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

    Returns:

    A string denoting the fragment component.

4.6.1.1.2  
**getHost()**

    public java.lang.String getHost()

    This method retrieves the hostport component of the URI referenced by this anchor element.

    **Note:** See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

    Returns:

    A string denoting the hostport component.

4.6.1.1.3  
**getHostname()**

    public java.lang.String getHostname()

    This method retrieves the host component of the URI referenced by this anchor element.

    **Note:** See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

    Returns:

    A string denoting the host component.

4.6.1.1.4  
**getPathname()**

    public java.lang.String getPathname()
This method retrieves the *abs_path* or *rel_path* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

Returns:

A string denoting the *abs_path* or *rel_path* component.

### 4.6.1.1.5 getPort()

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

This method retrieves the *port* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

Returns:

A string denoting the *port* component.

### 4.6.1.1.6 getProtocol()

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

This method retrieves the *scheme* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

Returns:

A string denoting the *scheme* component.

### 4.6.1.1.7 getSearch()

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

This method retrieves the *query* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

Returns:

A string denoting the *query* component.

### 4.6.1.1.8 setHash(java.lang.String)

```java
public void setHash(java.lang.String hash)
```

This method sets the *fragment* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

Parameters:

- `hash` – string denoting the new *fragment* component value.

### 4.6.1.1.9 setHost(java.lang.String)

```java
public void setHost(java.lang.String host)
```
This method sets the *hostport* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*host*  
– string denoting the new *hostport* component value.

### 4.6.1.1.10 setHostname(java.lang.String)

```java
public void setHostname(java.lang.String hostname)
```

This method sets the *host* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*hostname*  
– string denoting the new *host* component value.

### 4.6.1.1.11 setPathname(java.lang.String)

```java
public void setPathname(java.lang.String pathname)
```

This method sets the *abs_path* or *rel_path* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*pathname*  
– string denoting the new *abs_path* or *rel_path* component value.

### 4.6.1.1.12 setPort(java.lang.String)

```java
public void setPort(java.lang.String port)
```

This method sets the *port* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*port*  
– string denoting the new *port* component value.

### 4.6.1.1.13 setProtocol(java.lang.String)

```java
public void setProtocol(java.lang.String protocol)
```

This method sets the *scheme* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*scheme*  
– string denoting the new *scheme* component value.

### 4.6.1.1.14 setSearch(java.lang.String)

```java
public void setSearch(java.lang.String search)
```
This method sets the *query* component of the URI referenced by this anchor element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.1, for further information on the semantics of this method.

**Parameters:**

*query* – string denoting the new *query* component value.

### 4.6.1.2 Fields

No fields are defined.

### 4.6.2 HTMLDocumentExt

**public interface HTMLDocumentExt**

Provides functionality for accessing a document instance.

An object which implements `org.w3c.dom.html2.HTMLDocument` shall also implement this interface.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this type.

#### 4.6.2.1 Methods

**4.6.2.1.1 clear()**

```java
public void clear()
```

Clear the document’s contents.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

**4.6.2.1.2 getALinkColor()**

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

Retrieve the color to be used for active *a* (anchor) elements.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a color name or RGB triplet expressed using “#RRGGBB” syntax.

**4.6.2.1.3 getBgColor()**

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

Retrieve the color to be used for the background of the *body* element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a color name or RGB triplet expressed using “#RRGGBB” syntax.
4.6.2.1.4 getFgColor()

public java.lang.String getFgColor()

Retrieve the color to be used for text within the body element.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a color name or RGB triplet expressed using “#RRGGBB” syntax.

4.6.2.1.5 getLastModified()

public java.lang.String getLastModified()

Retrieve the last modified time of the document.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a last modified time.

4.6.2.1.6 getLinkColor()

public java.lang.String getLinkColor()

Retrieve the color to be used for inactive, unvisited a (anchor) elements.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a color name or RGB triplet expressed using “#RRGGBB” syntax.

4.6.2.1.7 getLocation()

public java.lang.String getLocation()

Retrieve the URI from which the document was loaded.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a URI.

4.6.2.1.8 getVLinkColor()

public java.lang.String getVLinkColor()

Retrieve the color to be used for visited a (anchor) elements.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A string denoting a color name or RGB triplet expressed using “#RRGGBB” syntax.
4.6.2.1.9  `getWindow()`

```java
public org.atsc.dom.environment.Window getWindow()
```

Retrieve the window object in which this document is loaded.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Returns:

A window object.

4.6.2.1.10  `setALinkColor(java.lang.String)`

```java
public void setALinkColor(java.lang.String aLinkColor)
```

This method sets the color to be used for active *a* (anchor) elements.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Parameters:

`aLinkColor` — string denoting the color.

4.6.2.1.11  `setBgColor(java.lang.String)`

```java
public void setBgColor(java.lang.String bgColor)
```

This method sets the color to be used for the background of the `body` element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Parameters:

`bgColor` — string denoting the color.

4.6.2.1.12  `setFgColor(java.lang.String)`

```java
public void setFgColor(java.lang.String fgColor)
```

This method sets the color to be used for text within a `body` element.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Parameters:

`fgColor` — string denoting the color.

4.6.2.1.13  `setLinkColor(java.lang.String)`

```java
public void setLinkColor(java.lang.String linkColor)
```

This method sets the color to be used for inactive, unvisited `a` (anchor) elements.

*Note:* See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Parameters:

`linkColor` — string denoting the color.
4.6.2.1.14 setVLinkColor(java.lang.String)

public void setVLinkColor(java.lang.String vLinkColor)

This method sets the color to be used for visited a (anchor) elements.

Note: See [DASE-DA], Section 5.3.1.2.3.3, for further information on the semantics of this method.

Parameters:

vLinkColor  – string denoting the color.

4.6.2.2 Fields

No fields are defined.

4.6.3 HTMLFormElementExt

public interface HTMLFormElementExt

Provides functionality for accessing a form element.

An object which implements org.w3c.dom.html2.HTMLFormElement shall also implement this interface.

Note: See [DASE-DA], Section 5.3.1.2.3.4, for further information on the semantics of this type.

4.6.3.1 Methods

4.6.3.1.1 getEncoding()

public java.lang.String getEncoding()

Retrieve the value of the form element’s enctype attribute.

Note: See [DASE-DA], Section 5.3.1.2.3.4, for further information on the semantics of this method.

Returns:

A string denoting a form encoding content type.

4.6.3.1.2 setEncoding(java.lang.String)

public void setEncoding(java.lang.String encoding)

This method sets value of the form element’s enctype attribute.

Note: See [DASE-DA], Section 5.3.1.2.3.4, for further information on the semantics of this method.

Parameters:

encoding  – string denoting a form encoding content type.

4.6.3.2 Fields

No fields are defined.
4.6.4 HTMLImageElementExt

public interface HTMLImageElementExt

Provides functionality for accessing an *img* (image) element.

An object which implements `org.w3c.dom.html2.HTMLImageElement` shall also implement this interface.

*Note:* See [DASE-DA], Section 5.3.1.2.3.5, for further information on the semantics of this type.

4.6.4.1 Methods

4.6.4.1.1 `getComplete()`

`public boolean getComplete()`

Retrieve an indication of whether the image load operation has completed.

*Note:* See [DASE-DA], Section 5.3.1.2.3.5, for further information on the semantics of this method.

Returns:

A boolean indicating if the load is complete.

4.6.4.1.2 `getLowSrc()`

`public java.lang.String getLowSrc()`

Retrieve the URI which references an alternative source of the image’s data.

*Note:* See [DASE-DA], Section 5.3.1.2.3.5, for further information on the semantics of this method.

Returns:

A string denoting the alternative source of the image’s data.

4.6.4.1.3 `setLowSrc(java.lang.String)`

`public void setLowSrc(java.lang.String lowSrc)`

This method sets value of the *img* element’s alternative data source.

*Note:* See [DASE-DA], Section 5.3.1.2.3.5, for further information on the semantics of this method.

Parameters:

`lowSrc` – string denoting the *img* element’s alternative data source.

4.6.4.2 Fields

No fields are defined.

4.6.5 HTMLObjectElementExt

public interface HTMLObjectElementExt

Provides functionality for accessing an object element.
An object which implements org.w3c.dom.html2.HTMLObjectElement shall also implement this interface.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this type.

### 4.6.5.1 Methods

#### 4.6.5.1.1 `getComplete()`

```java
public boolean getComplete()
```

Retrieve an indication of whether the object load operation has completed.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this method.

Returns:

A boolean indicating if the load is complete.

#### 4.6.5.1.2 `getLowSrc()`

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

Retrieve the URI which references an alternative source of the object’s data.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this method.

Returns:

A string denoting the alternative source of the object’s data.

#### 4.6.5.1.3 `getSrc()`

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

Retrieve the URI which references the object’s *data* attribute, which serves to specify the source data of the object.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this method.

Returns:

A string denoting the object’s data attribute value.

#### 4.6.5.1.4 `setLowSrc(java.lang.String)`

```java
public void setLowSrc(java.lang.String lowSrc)
```

This method sets value of the *object* element’s alternative data source.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this method.

Parameters:

*lowSrc* — string denoting the *object* element’s alternative data source.

#### 4.6.5.1.5 `setSrc(java.lang.String)`

```java
public void setSrc(java.lang.String src)
```
This method sets value of the object element’s data source.

*Note:* See [DASE-DA], Section 5.3.1.2.3.7, for further information on the semantics of this method.

**Parameters:**

- `src` — string denoting the object element’s data source.

### 4.6.5.2 Fields

No fields are defined.

### 4.7 `org.atsc.dom.views`

Specifies extensions to W3C DOM Level 2 View interfaces.

#### 4.7.1 DocumentViewExt

```java
public interface DocumentViewExt
```

Provides functionality for accessing a document’s view.

*Note:* See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this type.

#### 4.7.1.1 Methods

##### 4.7.1.1.1 `getGfxHeight()`

```java
public double getGfxHeight()
```

Retrieves the graphics plane height in the normalized coordinate space.

*Note:* See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

**Returns:**

- A length in the normalized coordinate space.

##### 4.7.1.1.2 `getGfxHeightPx()`

```java
public long getGfxHeightPx()
```

Retrieves the graphics plane height in pixels.

*Note:* See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

**Returns:**

- A length in pixels.

##### 4.7.1.1.3 `getGfxPosX()`

```java
public double getGfxPosX()
```

Retrieves the graphics plane leftmost coordinate in the normalized coordinate space.

*Note:* See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.
Returns:
A coordinate in the normalized coordinate space.

4.7.1.1.4  getGfxPosY()

public double getGfxPosY()

Retrieves the graphics plane topmost coordinate in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A coordinate in the normalized coordinate space.

4.7.1.1.5  getGfxWidth()

public double getGfxWidth()

Retrieves the graphics plane width in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in the normalized coordinate space.

4.7.1.1.6  getGfxWidthPx()

public long getGfxWidthPx()

Retrieves the graphics plane width in pixels.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in pixels.

4.7.1.1.7  getRefHeightMm()

public double getRefHeightMm()

Retrieves the reference frame height in millimeters.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in millimeters.

4.7.1.1.8  getRefreshOnChange()

public boolean getRefreshOnChange()

Retrieves the view’s refresh on change flag.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.
Returns:
A boolean indicating if refresh on change is active or not.

4.7.1.1.9  getRefWidthMm()

public double getRefWidthMm()

Retrieves the reference frame width in millimeters.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in millimeters.

4.7.1.1.10  getSampleBitsA()

public long getSampleBitsA()

Retrieves the number of bits of resolution of the alpha component.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
Number of bits.

4.7.1.1.11  getSampleBitsB()

public long getSampleBitsB()

Retrieves the number of bits of resolution of the blue component.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
Number of bits.

4.7.1.1.12  getSampleBitsG()

public long getSampleBitsG()

Retrieves the number of bits of resolution of the green component.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
Number of bits.

4.7.1.1.13  getSampleBitsR()

public long getSampleBitsR()

Retrieves the number of bits of resolution of the red component.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.
Returns:
Number of bits.

4.7.1.1.14 getVidHeight()

public double getVidHeight()

Retrieves the video plane height in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in the normalized coordinate space.

4.7.1.1.15 getVidHeightPx()

public long getVidHeightPx()

Retrieves the video plane height in pixels.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in pixels.

4.7.1.1.16 getVidPosX()

public double getVidPosX()

Retrieves the video plane leftmost coordinate in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A coordinate in the normalized coordinate space.

4.7.1.1.17 getVidPosY()

public double getVidPosY()

Retrieves the video plane topmost coordinate in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A coordinate in the normalized coordinate space.

4.7.1.1.18 getVidWidth()

public double getVidWidth()

Retrieves the video plane width in the normalized coordinate space.

Note: See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.
Returns:
A length in the normalized coordinate space.

4.7.1.19  
\texttt{getVidWidthPx()}

\begin{verbatim}
public long getVidWidthPx()
\end{verbatim}

Retrieves the video plane width in pixels.

\textit{Note:} See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Returns:
A length in pixels.

4.7.1.20  
\texttt{refresh(java.lang.String)}

\begin{verbatim}
public boolean refresh(java.lang.String id)
\end{verbatim}

Refresh the current rendition of a document or an element and its children if an underlying resource has changed. If no change has occurred, then no side-effect is produced.

\textit{Note:} See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Parameters:
\begin{itemize}
\item \textit{id} – null, empty string, or string referencing an identified element.
\end{itemize}

Returns:
A boolean indicating if refresh occurred or not.

4.7.1.21  
\texttt{setRefreshOnChange(boolean)}

\begin{verbatim}
public void setRefreshOnChange(boolean refreshOnChange)
\end{verbatim}

Sets the view's refresh on change flag. If setting to \texttt{true} and the document instance has changed since the time the flag was set to \texttt{false}, then a presentation refresh shall occur.

\textit{Note:} See [DASE-DA], Section 5.3.1.2.4.1, for further information on the semantics of this method.

Parameters:
\begin{itemize}
\item \textit{refreshOnChange} – value to set.
\end{itemize}

4.7.1.2  
\textbf{Fields}

No fields are defined.

4.8  
\textbf{org.atsc.graphics}

This package provides the classes for implementing graphics applications. It extends the functionality of the \texttt{java.awt} package.

4.8.1  
\textbf{AtscBufferedImage}

\begin{verbatim}
public class AtscBufferedImage
extends java.awt.Image
\end{verbatim}
The `AtscBufferedImage` class describes an `Image` with an accessible buffer of image data. The image buffer has the same model as the on-screen graphics buffer. If the on-screen graphics buffer approximates color values (including the alpha component), then the image buffer will store the same approximations. Initially, a new `AtscBufferedImage` is transparent.

### 4.8.1.1 Constructors

#### 4.8.1.1.1 `AtscBufferedImage(int, int)`

```java
public AtscBufferedImage(int width, int height)
```

Constructs an `AtscBufferedImage`. Initially, the image is transparent.

**Parameters:**

- `width` – width of the created image.
- `height` – height of the created image.

### 4.8.1.2 Methods

The following methods are inherited from `java.awt.Image`:

- `getScaledInstance`.

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

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

#### 4.8.1.2.1 `flush()`

```java
public void flush()
```

Flushes all resources being used to cache optimization information. The underlying pixel data is unaffected.

**Overrides:**

- `java.awt.Image.flush()`

#### 4.8.1.2.2 `getGraphics()`

```java
public java.awt.Graphics getGraphics()
```

Creates a graphics context for drawing to this image.

**Returns:**

- a graphics context to draw to the off-screen image.

**Overrides:**

- `java.awt.Image.getGraphics()`

**See also:** `java.awt.Graphics`, `java.awt.Component.createImage(int, int)`.

#### 4.8.1.2.3 `getHeight()`

```java
public int getHeight()
```

Returns the height of the `AtscBufferedImage`.

**Returns:**

- the height of the `AtscBufferedImage`.
4.8.1.2.4 getHeight(java.awt.image.ImageObserver)

```java
public int getHeight(java.awt.image.ImageObserver observer)
```

Determines the height of the image. If the height is not yet known, this method returns -1 and the specified ImageObserver object is notified later.

Parameters:

- `observer` – an object waiting for the image to be loaded.

Returns:

the height of this image, or -1 if the height is not yet known.

Overrides:

- `java.awt.Image.getHeight(java.awt.image.ImageObserver)`

See also: `java.awt.Image.getHeight(java.awt.image.ImageObserver)`, `java.awt.image.ImageObserver`.

4.8.1.2.5 getProperty(java.lang.String, java.awt.image.ImageObserver)

```java
public java.lang.Object getProperty(java.lang.String name, java.awt.image.ImageObserver observer)
```

Gets a property of this image by name.

Individual property names are defined by the various image formats. If a property is not defined for a particular image, this method returns the value of the `java.awt.Image UndefinedProperty` field.

If the properties for this image are not yet known, this method returns null, and the ImageObserver object is notified later.

The property name "comment" should be used to store an optional comment which can be presented to the application as a description of the image, its source, or its author.

Parameters:

- `name` – a property name.
- `observer` – an object waiting for the image to be loaded.

Returns:

the value of the named property.

Overrides:


4.8.1.2.6 getRGB(int, int)

```java
public int getRGB(int x, int y)
```

Returns the RGB value of the given pixel in the default RGB color model. The alpha, red, green and blue components of the color are each scaled to be a value between 0 and 255. Bits 24-31 of the returned integer are the alpha value, bits 16-23 are the read value, bits 8-15 are the green value, and bits 0-7 are the blue value.
Parameters:
x, y – the coordinates of the pixel from which to get the pixel value.

Returns:
An integer pixel in the default RGB color model.

4.8.1.2.7  getSource()

public java.awt.image.ImageProducer getSource()

Gets the object that produces the pixels for the image. This method is called by the image filtering classes and by methods that perform image conversion and scaling.

Returns:
the image producer that produces the pixels for this image.

Overrides:
java.awt.Image.getSource()

See also: java.awt.image.ImageProducer.

4.8.1.2.8  getWidth()

public int getWidth()

Returns the width of the AtscBufferedImage.

Returns:
the width of the AtscBufferedImage.

4.8.1.2.9  getWidth(java.awt.image.ImageObserver)

public int getWidth(java.awt.image.ImageObserver observer)

Determines the width of the image. If the width is not yet known, this method returns -1 and the specified ImageObserver object is notified later.

Parameters:
observer – an object waiting for the image to be loaded.

Returns:
the width of this image, or -1 if the width is not yet known.

Overrides:
java.awt.Image.getWidth(java.awt.image.ImageObserver)

See also: java.awt.Image.getWidth(java.awt.image.ImageObserver),
java.awt.image.ImageObserver.

4.8.1.3  Fields

The following fields are inherited from java.awt.Image: SCALE_AREA_AVERAGING, SCALE_DEFAULT, SCALE_FAST, SCALE_REPLICATE, SCALE_SMOOTH, UndefinedProperty.
4.8.2 FontFactory

public class FontFactory
    extends java.lang.Object

The FontFactory class provides a mechanism for loading application-defined font resources.

4.8.2.1 Constructors

protected FontFactory()

Protected constructor.

4.8.2.2 Methods

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

4.8.2.2.1 load(javax.tv.locator.Locator,int,int)

public static java.awt.Font
    load(javax.tv.locator.Locator locator, int style, int size)
    throws FontFormatException, javax.tv.locator.InvalidLocatorException, java.io.IOException

Load font referenced by locator and as further specified by style and size arguments. The platform shall not perform font substitution if the referenced font is not available.

Parameters:

locator – locator which references font resource.
style – font style as defined by java.awt.Font.
size – font size as defined by java.awt.Font.

Returns:

An instance of java.awt.Font.

Throws:

FontFormatException – if a content validity error is detected when decoding the referenced font resource or if the font does not support the requested style and size.

javax.tv.locator.InvalidLocatorException – if locator is invalid or does not reference a font resource.

java.io.IOException – if an access exception occurs.

See also: java.awt.Font.

4.8.2.3 Fields

No fields are defined.
4.8.3 FontFormatException

public class FontFormatException
    extends java.lang.Exception

    A FontFormatException exception is thrown when a format error occurs when decoding a font resource or when the requested font does not support the specified metrics.

4.8.3.1 Constructors

4.8.3.1.1 FontFormatException()

    public FontFormatException()

    Construct a FontFormatException with no detail message.

4.8.3.1.2 FontFormatException(java.lang.String)

    public FontFormatException(java.lang.String s)

    Construct a FontFormatException with the specified detail message.

    Parameters:
    s – the detail message

4.8.3.2 Methods

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

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

4.8.3.3 Fields

    No fields are defined.

4.9 org.atsc.management

    This package defines classes and interfaces which support the state and status management functions specified by [X.731].

    Note: See [DASE-PA], Section 4.1, State and Status Management, for the semantics of this functionality.

4.9.1 AdministrativeState

    public interface AdministrativeState

    Interface that defines the administrative state attribute.

    Note: See [DASE-PA], Section 4.1.1.3, for further information regarding the semantics of this interface.
4.9.1.1 Methods

4.9.1.1.1 getAdministrativeState()

    public int getAdministrativeState()

    Retrieve the current value of the administrative state attribute.

    Returns:

    The current administrative state value.

4.9.1.1.2 setLock(boolean)

    public void setLock(boolean lock)
    throws org.atsc.security.AccessDeniedException

    Called to change the value of the administrative state.

    Parameters:

    lock – A boolean value setting the lock; true indicates LOCK, false indicates UNLOCK.

    Throws:

    org.atsc.security.AccessDeniedException – if state cannot be changed.

4.9.1.2 Fields

4.9.1.2.1 ADMIN_STATE_ID

    public static final short ADMIN_STATE_ID = 1

    Administrative state attribute ID.

4.9.1.2.2 LOCKED

    public static final int LOCKED = 2

    Note: See [X.731], Section 8.1.1.3, for further information regarding the semantics of this state.

4.9.1.2.3 SHUTTING_DOWN

    public static final int SHUTTING_DOWN = 4

    Note: See [X.731], Section 8.1.1.3, for further information regarding the semantics of this state.

4.9.1.2.4 UNLOCKED

    public static final int UNLOCKED = 1

    Note: See [X.731], Section 8.1.1.3, for further information regarding the semantics of this state.

4.9.2 AlarmStatus

    public interface AlarmStatus

    Interface that defines the alarm status attribute.
Note: See [DASE-PA], Section 4.1.2.1, for further information regarding the semantics of this interface.

4.9.2.1 Methods

4.9.2.1.1 clearAlarm()

    public void clearAlarm()

    Called to clear an outstanding alarm. The controlling process has acted on the alarm.

4.9.2.1.2 getAlarmStatus()

    public int getAlarmStatus()

    Called to get the current set of values of the alarm status attribute.

    Returns:

        The current alarm status value.

4.9.2.2 Fields

4.9.2.2.1 ALARM_OUTSTANDING

    public static final int ALARM_OUTSTANDING = 16

    Note: See [X.731], Section 8.1.2.1, for further information regarding the semantics of this status.

4.9.2.2.2 ALARM_STATUS_ID

    public static final short ALARM_STATUS_ID = 8

    Alarm status attribute ID.

4.9.2.2.3 CRITICAL

    public static final int CRITICAL = 2

    Note: See [X.731], Section 8.1.2.1, for further information regarding the semantics of this status.

4.9.2.2.4 MAJOR

    public static final int MAJOR = 4

    Note: See [X.731], Section 8.1.2.1, for further information regarding the semantics of this status.

4.9.2.2.5 MINOR

    public static final int MINOR = 8

    Note: See [X.731], Section 8.1.2.1, for further information regarding the semantics of this status.

4.9.2.2.6 UNDER_REPAIR

    public static final int UNDER_REPAIR = 1
Note: See [X.731], Section 8.1.2.1, for further information regarding the semantics of this status.

4.9.3 AvailabilityStatus

public interface AvailabilityStatus

This interface defines the availability status attribute.

Note: See [DASE-PA], Section 4.1.2.3, for further information regarding the semantics of this interface.

4.9.3.1 Methods

4.9.3.1.1 getAvailabilityStatus()

public int getAvailabilityStatus()

Called to get the current set of values of the availability status attribute.

Returns:

The current availability status value.

4.9.3.2 Fields

4.9.3.2.1 AVAILABILITY_STATUS_ID

public static final short AVAILABILITY_STATUS_ID = 32

Availability status attribute ID.

4.9.3.2.2 DEGRADED

public static final int DEGRADED = 65536

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.3 DEPENDENCY

public static final int DEPENDENCY = 32768

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.4 FAILED

public static final int FAILED = 2048

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.5 INTEST

public static final int INTEST = 1024

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.
4.9.3.2.6 **LOG_FULL**

```java
public static final int LOG_FULL = 262144
```

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.7 **NOT_INSTALLED**

```java
public static final int NOT_INSTALLED = 131072
```

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.8 **OFFDUTY**

```java
public static final int OFFDUTY = 16384
```

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.9 **OFFLINE**

```java
public static final int OFFLINE = 8192
```

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.3.2.10 **POWEROFF**

```java
public static final int POWEROFF = 4096
```

Note: See [X.731], Section 8.1.2.3, for further information regarding the semantics of this status.

4.9.4 **ManagementPermission**

```java
public final class ManagementPermission
extends org.atsc.security.AtscPermission
```

The `ManagementPermission` class is used for permissions related to all security-protected methods in this package.

The target name is either "lock" which covers the `AdministrativeState.setLock()` method or "clear" which covers the `AlarmStatus.clearAlarm()` method. The action is ignored.

4.9.4.1 **Constructors**

4.9.4.1.1 **ManagementPermission(java.lang.String, java.lang.String)**

```java
public ManagementPermission(java.lang.String name, java.lang.String action)
```

Creates a new `ManagementPermission` object with the specified target name.

Parameters:

- `name` – the target name is either "lock" or "clear".
- `action` – ignored.
4.9.4.2 Methods

The following methods are inherited from org.atsc.security.AtscPermission: equals, getActions, hashCode.

The following methods are inherited from java.security.Permission: checkGuard, getName, newPermissionCollection, toString.

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

4.9.4.2.1 implies(java.security.Permission)

public boolean implies(java.security.Permission permission)

Checks if this permission object implies the specified permission.

Parameters:

permission – the permission to check.

Returns:

The value true if this object implies the specified permission.

Overrides:

org.atsc.security.AtscPermission.implies(java.security.Permission)

4.9.4.3 Fields

No fields are defined.

4.9.5 ObjectStates

public interface ObjectStates

extends AdministrativeState, AlarmStatus, AvailabilityStatus, OperationalState, ProceduralStatus, UsageState

This interface allows objects to implement a unified interface which supports all or a suitable subset of [X.731] state and status attributes.

Individual state and status attribute values are defined using bit masks. One and only one bit shall be set for each supported state value. None or multiple bits may be set for each supported status attribute value.

4.9.5.1 Methods

The following methods are inherited from org.atsc.management.AlarmStatus: clearAlarm, getAlarmStatus.

The following methods are inherited from org.atsc.management.ProceduralStatus: getProceduralStatus.

The following methods are inherited from org.atsc.management.AvailabilityStatus: getAvailabilityStatus.

The following methods are inherited from org.atsc.management.UsageState: getUsageState.

The following methods are inherited from org.atsc.management.OperationalState: getOperationalState.
The following methods are inherited from org.atsc.management.AdministrativeState: getAdministrativeState, setLock.

4.9.5.1.1   addStateChangeListener(StateChangeListener)

public void addStateChangeListener(StateChangeListener listener)

Called to register a StateChangeListener for StateChangeEvents.

Parameters:

   listener    – A listener to notify about state and status changes.

4.9.5.1.2   getCurrentState()

public int getCurrentState()

Called to get the current value of all supported states.

Returns:

   a bit-mask representing the individual values of all supported states. One and only one bit shall be set for each state value when updating states.

4.9.5.1.3   getCurrentStatus()

public int getCurrentStatus()

Called to get the current value of all supported status attributes.

Returns:

   a bit mask representing the current values of all supported status attributes.

4.9.5.1.4   getStatesReported()

public short[] getStatesReported()

Called to determine which state and status attributes are supported by the class implementing this interface.

Returns:

   an array of integers denoting the set of supported state and status attribute identifiers.

4.9.5.1.5   removeStateChangeListener(StateChangeListener)

public void removeStateChangeListener(StateChangeListener listener)

Called to deregister a StateChangeListener.

Parameters:

   listener    – A previously registered listener.

4.9.5.2   Fields

The following fields are inherited from org.atsc.management.AlarmStatus: ALARM_OUTSTANDING, ALARM_STATUS_ID, CRITICAL, MAJOR, MINOR, UNDER_REPAIR.

The following fields are inherited from org.atsc.management.ProceduralStatus: INIT_REQUIRED, INITIALIZING, NOT_INITIALIZED, PROCEDURAL_STATUS_ID, REPORTING, TERMINATING.
The following fields are inherited from `org.atsc.management.AvailabilityStatus`:

- `AVAILABILITY_STATUS_ID`
- `DEGRADED`
- `DEPENDENCY`
- `FAILED`
- `INTEST`
- `LOG_FULL`
- `NOT_INSTALLED`
- `OFFDUTY`
- `OFFLINE`
- `POWEROFF`.

The following fields are inherited from `org.atsc.management.UsualState`:

- `ACTIVE`
- `BUSY`
- `IDLE`
- `USAGE_STATE_ID`.

The following fields are inherited from `org.atsc.management.OperationalState`:

- `DISABLED`
- `ENABLED`
- `OPERATIONAL_STATE_ID`.

The following fields are inherited from `org.atsc.management.AdministrativeState`:

- `ADMIN_STATE_ID`
- `LOCKED`
- `SHUTTING_DOWN`
- `UNLOCKED`.

### 4.9.5.2.1 NOSTATUS

```java
public static final int NOSTATUS = 0
```

No status currently available.

### 4.9.6 OperationalState

```java
public interface OperationalState
```

Interface that defines the operational state attribute.

Note: See [DASE-PA], Section 4.1.1.1, for further information regarding the semantics of this interface.

#### 4.9.6.1 Methods

**4.9.6.1.1 getOperationalState()**

```java
public int getOperationalState()
```

Called to get the current value of the operational state attribute.

Returns:

The current operational state value.

#### 4.9.6.2 Fields

**4.9.6.2.1 DISABLED**

```java
public static final int DISABLED = 8
```

Note: See [X.731], Section 8.1.1.1, for further information regarding the semantics of this state.

**4.9.6.2.2 ENABLED**

```java
public static final int ENABLED = 16
```

Note: See [X.731], Section 8.1.1.1, for further information regarding the semantics of this state.

**4.9.6.2.3 OPERATIONAL_STATE_ID**

```java
public static final short OPERATIONAL_STATE_ID = 2
```

Operational state attribute ID.
4.9.7  ProceduralStatus

public interface ProceduralStatus

Interface that defines the *procedural* status attribute.

Note: See [DASE-PA], Section 4.1.2.2, for further information regarding the semantics of this interface.

4.9.7.1  Methods

4.9.7.1.1  getProceduralStatus()

public int getProceduralStatus()

Called to get the current set of values of the *procedural* status attribute.

Returns:

The current *procedural* status value.

4.9.7.2  Fields

4.9.7.2.1  INIT_REQUIRED

public static final int INIT_REQUIRED = 32

Note: See [X.731], Section 8.1.2.2, for further information regarding the semantics of this status.

4.9.7.2.2  INITIALIZING

public static final int INITIALIZING = 128

Note: See [X.731], Section 8.1.2.2, for further information regarding the semantics of this status.

4.9.7.2.3  NOT_INITIALIZED

public static final int NOT_INITIALIZED = 64

Note: See [X.731], Section 8.1.2.2, for further information regarding the semantics of this status.

4.9.7.2.4  PROCEDURAL_STATUS_ID

public static final short PROCEDURAL_STATUS_ID = 16

Procedural status ID.

4.9.7.2.5  REPORTING

public static final int REPORTING = 256

Note: See [X.731], Section 8.1.2.2, for further information regarding the semantics of this status.

4.9.7.2.6  TERMINATING

public static final int TERMINATING = 512
Note: See [X.731], Section 8.1.2.2, for further information regarding the semantics of this status.

4.9.8 SourceIndicator

public class SourceIndicator
    extends java.lang.Object

This class defines possible values signaling the cause of a state change.
See also: StateChangeEvent.

4.9.8.1 Constructors

4.9.8.1.1 SourceIndicator(java.lang.String)

protected SourceIndicator(java.lang.String cause)

4.9.8.2 Methods

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

4.9.8.3 Fields

4.9.8.3.1 EXTERNAL_CAUSE

public static final SourceIndicator EXTERNAL_CAUSE

State change caused by an activity external to the managed object (e.g. an application was resumed by an external call which caused the state change from disabled to enabled).

4.9.8.3.2 INTERNAL_CAUSE

public static final SourceIndicator INTERNAL_CAUSE

State change caused by an activity internal to the managed object (e.g. an application changed from enabled to disabled state because it lost some essential resource).

4.9.9 StateChangeEvent

public class StateChangeEvent
    extends java.util.EventObject

This event is generated when a state changes its value. It is distributed to all registered StateChangeListeners.

4.9.9.1 Constructors

4.9.9.1.1 StateChangeEvent(java.lang.Object, ...)

public StateChangeEvent(
    java.lang.Object source,
    int newValue,
    int oldValue,
SourceIndicator sourceIndicator,  
short stateId)

Parameters:

source – The object that caused this event.
newValue – The new value of the state.
oldValue – The old value of the state.
sourceIndicator – The cause of the event.
stateId – The identifier of the state attribute that changed.

4.9.9.2 Methods

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

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

4.9.9.2.1 getNewValue()

public int getNewValue()

Called to determine the new value of the state.

Returns:

The new value of the state.

4.9.9.2.2 getOldValue()

public int getOldValue()

Called to determine the original value of the state.

Returns:

The original value of the state before it changed.

4.9.9.2.3 getSourceIndicator()

public SourceIndicator getSourceIndicator()

Called to determine the cause of the event.

Returns:

The cause of the state change.

See also: SourceIndicator.

4.9.9.3 Fields

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

4.9.10 StateChangeException

public class StateChangeException  
extends java.lang.Exception
A base exception class relates to the ObjectStates interface. This exception or one of its subclasses is thrown when an invalid state change would be result.

### 4.9.10.1 Constructors

#### 4.9.10.1.1 StateChangeException(short, int)

```java
public StateChangeException(short state, int value)
```

Construct the exception with the specified state and value attributes, but no detail message.

**Parameters:**

- `state` – The state which has been violated.
- `value` – The current value of the violated state.

#### 4.9.10.1.2 StateChangeException(short, int, java.lang.String)

```java
public StateChangeException(short state, int value, java.lang.String reason)
```

Construct the exception with the specified state and value attributes, and detail message.

**Parameters:**

- `state` – The state which has been violated.
- `value` – The current value of the violated state.
- `reason` – The reason the state has been violated.

### 4.9.10.2 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)`, `wait(long,int)`, and `wait(long,int)`.

#### 4.9.10.2.1 getState()

```java
public short getState()
```

Called to determine which state consistency has been violated.

**Returns:**

A value representing the specific state.

#### 4.9.10.2.2 getValue()

```java
public int getValue()
```

Called to get the current value of the violated state.

**Returns:**

The value of the state.
4.9.10.3 Fields

No fields are defined.

4.9.11 StateChangeListener

```java
public interface StateChangeListener
extends java.util.EventListener
```

This interface is implemented by classes interested in being notified of state and status changes. If an object which implements `StateChangeListener` registers via the `addStateChangeListener` method, it will be notified by calling the `stateChange` or `statusChange` methods.

4.9.11.1 Methods

4.9.11.1.1 stateChanged(StateChangeEvent)

```java
public void stateChanged(StateChangeEvent event)
```

Called to notify a `StateChangeListener` about a state change.

Parameters:
- `event` – Information about what state has changed.

4.9.11.1.2 statusChanged(StatusChangeEvent)

```java
public void statusChanged(StatusChangeEvent event)
```

Called to notify a `StateChangeListener` about a status change.

Parameters:
- `event` – Information about what status has changed.

4.9.11.2 Fields

No fields are defined.

4.9.12 StatusChangeEvent

```java
public class StatusChangeEvent
extends java.util.EventObject
```

This event is generated when a status changes its value. It is distributed to all registered `StateChangeListeners`.

4.9.12.1 Constructors

4.9.12.1.1 StatusChangeEvent(java.lang.Object, ...)

```java
public StatusChangeEvent(
    java.lang.Object source,
    int newValue,
    int oldValue,
    SourceIndicator sourceIndicator,
    short statusId)
```
Parameters:

- **source**  – The object that caused this event.
- **newValue**  – The new value of the status.
- **oldValue**  – The old value of the status.
- **sourceIndicator**  – The cause of the event.
- **statusId**  – The identifier of the status attribute that changed.

### 4.9.12.2 Methods

The following methods are inherited from `java.util.EventObject`:\n\n**getSource**, **toString**.

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

#### 4.9.12.2.1 getNewValue()

```java
public int getNewValue()
```

Called to determine the new value of the status.

**Returns:**

The new value of the status.

#### 4.9.12.2.2 getOldValue()

```java
public int getOldValue()
```

Called to determine the original value of the status.

**Returns:**

The original value of the status before it changed.

#### 4.9.12.2.3 getSourceIndicator()

```java
public SourceIndicator getSourceIndicator()
```

Called to determine the cause of the event.

**Returns:**

The cause of the status change.

See also: `SourceIndicator`.

### 4.9.12.3 Fields

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

### 4.9.13 UsageState

```java
public interface UsageState
```

This interface defines the usage state attribute.

**Note:** See [DASE-PA], Section 4.1.1.2, for further information regarding the semantics of this interface.
4.9.13.1 Methods

4.9.13.1.1 getUsageState()

    public int getUsageState()
    
    Called to get the current value of the usage state attribute.
    
    Returns:
    
    The current usage state value.

4.9.13.2 Fields

4.9.13.2.1 ACTIVE

    public static final int ACTIVE = 64
    
    Note: See [X.731], Section 8.1.1.2, for further information regarding the semantics of this state.

4.9.13.2.2 BUSY

    public static final int BUSY = 128
    
    Note: See [X.731], Section 8.1.1.2, for further information regarding the semantics of this state.

4.9.13.2.3 IDLE

    public static final int IDLE = 32
    
    Note: See [X.731], Section 8.1.1.2, for further information regarding the semantics of this state.

4.9.13.2.4 USAGE_STATE_ID

    public static final short USAGE_STATE_ID = 4
    
    Usage state attribute ID.

4.10 org.atsc.net

This package provides the classes for implementing networking applications. It extends the functionality of the java.net package.

4.10.1 DatagramSocketBufferControl

    public class DatagramSocketBufferControl
        extends java.lang.Object
        
    This class is intended to set the buffer size option for a specific DatagramSocket. The SO_RCVBUF option is used by the platform's networking code as a hint for the size to use to allocate the underlying network I/O buffers.
        
    See also: java.net DatagramSocket.
4.10.1.1 Constructors

4.10.1.1.1 DatagramSocketBufferControl()

protected DatagramSocketBufferControl()

Protected constructor.

4.10.1.2 Methods

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

4.10.1.2.1 getReceiveBufferSize(java.net.DatagramSocket)

public static int getReceiveBufferSize(java.net.DatagramSocket d) throws java.net.SocketException

Get value of the SO_RCVBUF option for this socket; i.e., the buffer size used by the platform for input on this socket.

Parameters:

\( d \) – The DatagramSocket for which to query the receive buffer size.

Returns:

The size of the receive buffer, in bytes.

Throws:

java.net.SocketException – If there is an error when querying the SO_RCVBUF option.

4.10.1.2.2 setReceiveBufferSize(java.net.DatagramSocket, int)

public static void setReceiveBufferSize(java.net.DatagramSocket d, int size) throws java.net.SocketException

Sets the SO_RCVBUF option to the specified value for this DatagramSocket. The SO_RCVBUF option is used by the platform's networking code as a hint for the size to use to allocate set the underlying network I/O buffers.

Increasing buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data. For UDP, this sets the maximum size of a packet that may be sent on this socket.

Because SO_RCVBUF is a hint, applications that want to verify what size the buffers were set to should call getReceiveBufferSize.

Parameters:

\( d \) – The DatagramSocket for which to change the receive buffer size.

size – The requested size of the receive buffer, in bytes.

Throws:

java.net.SocketException – If there is an error when setting the SO_RCVBUF option.

java.lang.IllegalArgumentException – If size is zero or is negative.
4.10.1.3 Fields

No fields are defined.

4.11 org.atsc.preferences

This package defines a set of interfaces and classes which provide a mechanism to define a set of preferences either at the system (receiver) level or at the end-user level.

4.11.1 FavoriteChannelsPreference

```java
public interface FavoriteChannelsPreference
        extends Preference, javax.tv.service.navigation.FavoriteServicesName
```

This interface represents a list of favorite channels in the form of a user preference. Channels are represented by Java TV locators. There may be more than one such a preference in the registry. Individual instances of this preference are identified by their names.

Note: In the present context, a channel is to be understood as a virtual channel or a service, and not as a physical transmission channel.

See also: `javax.tv.service.navigation.FavoriteServicesName`, `javax.tv.service.Service`.

4.11.1.1 Methods

The following methods are inherited from `Preference`: `addPreferenceChangeListener`, `getPreferenceName`, `removePreferenceChangeListener`.

The following methods are inherited from `javax.tv.service.navigation.FavoriteServicesName`: `getName`.

4.11.1.1.1 addChannel(javax.tv.locator.Locator)

```java
public void addChannel(javax.tv.locator.Locator locator)
        throws org.atsc.security.AccessDeniedException,
                javax.tv.locator.InvalidLocatorException
```

This method allows an insertion of a new favorite channel to the list.

Parameters:

`locator` – A locator representing the favorite channel.

Throws:

`org.atsc.security.AccessDeniedException` – when the caller does not have sufficient permission.

`javax.tv.locator.InvalidLocatorException` – when the locator does not represent a valid channel locator.

4.11.1.1.2 getChannelList()

```java
public javax.tv.locator.Locator[] getChannelList()
```

Returns a list of favorite channels in the form of Locators.

Returns:

An array of locators representing user favorite channels.
4.11.1.3  isFavorite(javax.tv.locator.Locator)

public boolean isFavorite(javax.tv.locator.Locator locator)

This method determines whether the specified channel is on the list of favorite channels.

Parameters:

locator – A locator representing the channel in question.

Returns:

The value true if the specified channel is on this list of favorite channels; otherwise, false.

4.11.1.4  removeChannel(javax.tv.locator.Locator)

public void removeChannel(javax.tv.locator.Locator locator)

throws org.atsc.security.AccessDeniedException,
javax.tv.locator.InvalidLocatorException

This method removes a channel from the favorite channels list.

Parameters:

locator – A locator representing the channel.

Throws:

org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.

javax.tv.locator.InvalidLocatorException – when the locator does not represent a valid channel locator.

4.11.2  Fields

The following fields are inherited from PreferenceNames: FAVORITE_CHANNELS, LANGUAGE,
PERSONAL_DATA, RATING.

4.11.2  InvalidPreferenceException

public class InvalidPreferenceException
extends java.lang.Exception

This exception signals that a preference of the specified name is invalid in the current context, e.g., it is already present in the repository and cannot be inserted again, or does not exist and cannot be retrieved, etc.

4.11.2.1  Constructors

4.11.2.1.1  InvalidPreferenceException()

public InvalidPreferenceException()

Constructor with no detail message.

4.11.2.1.2  InvalidPreferenceException(java.lang.String)

public InvalidPreferenceException(java.lang.String reason)

Constructor taking a detail message.
4.11.2.2 Methods
The following methods are inherited from `java.lang.Throwable`: `fillInStackTrace`, `getLocalizedMessage`, `getMessage`, `printStackTrace()`, `printStackTrace(java.io.PrintStream)`, `printStackTrace(java.io.PrintWriter)`, `toString`.

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

4.11.2.3 Fields
No fields are defined.

4.11.3 LanguagePreference

```java
class LanguagePreference extends Preference
```

This interface permits the specification of a language preference according to distinct language use scopes. An ordered list of language tags which adhere to [LANG-TAGS] is used to specify preferred languages. The first language in the list is the most desirable language.

4.11.3.1 Methods

The following methods are inherited from `Preference`: `addPreferenceChangeListener`, `getPreferenceName`, `removePreferenceChangeListener`.

4.11.3.1.1 `getLanguage(LanguageScope)`

```java
public java.lang.String[] getLanguage(LanguageScope scope)
```

This method returns an ordered list (most desirable first) of language tags according to the specified scope, each of which adhere to the syntax specified by [LANG-TAGS].

Parameters:

- `scope` – Scope of the preferred language (audio, text, etc.).

Returns:

An array of strings representing language tags.

4.11.3.1.2 `setLanguage(LanguageScope, java.lang.String[])`

```java
public void setLanguage(LanguageScope scope, java.lang.String[] valueList)
```

This method allows an application to change the language preference.

Parameters:

- `scope` – Scope of the preferred language (audio, text, etc.).
- `valueList` – An ordered list of strings representing language code.
Throws:

org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.

4.11.3.2 Fields

The following fields are inherited from PreferenceNames: FAVORITE_CHANNELS, LANGUAGE, PERSONAL_DATA, RATING.

4.11.4 LanguageScope

public class LanguageScope
    extends java.lang.Object

This class defines an enumeration of the areas where different languages may be used.

4.11.4.1 Constructors

4.11.4.1.1 LanguageScope(java.lang.String)

protected LanguageScope(java.lang.String name)

Parameters:

name – An implementation defined name which designates the specific enumeration entry (e.g., "audio", "subtitle", and "text").

4.11.4.2 Methods

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

4.11.4.3 Fields

4.11.4.3.1 AUDIO

public static final LanguageScope AUDIO

Audio language.

4.11.4.3.2 SUBTITLE

public static final LanguageScope SUBTITLE

Subtitle language.

4.11.5 PersonalDataPreference

public interface PersonalDataPreference
    extends Preference

This is an extensible preference object for user personal data, such as address, etc. The following keys are pre-defined: "name.first", "name.middle", "name.last", "address.street", "address.city", "address.state", "address.postal.code", "address.country", "address.email", and "phone.home".
The syntax of keys used for personal data preferences shall take the form of a *dotted* string such as is used as a key for Java system properties, and as used by the pre-defined keys listed above.

*Note:* The term *key* used here and throughout this specification is not related to the use of this term in security architectures; rather, it is used in the sense of a *lookup index*, e.g., as a key to a hash table entry.

*Note:* The semantics of the pre-defined keys listed above are not defined by this specification.

### 4.11.5.1 Methods

The following methods are inherited from Preference: addPreferenceChangeListener, getPreferenceName, removePreferenceChangeListener.

#### 4.11.5.1.1 getKeys()

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

This method returns an array of strings representing the supported keys of personal data preferences.

Returns:

An array of strings representing personal data preference keys.

#### 4.11.5.1.2 get(java.lang.String)

```java
public java.lang.String get(java.lang.String key)
throws org.atsc.security.AccessDeniedException, InvalidPreferenceException
```

This method returns the value of the specified personal information.

Parameters:

- `key` – A key of the personal information.

Returns:

A string representing the value of the specified personal information.

Throws:

- `org.atsc.security.AccessDeniedException` – when the caller does not have sufficient permission.
- `InvalidPreferenceException` – when the information with the specified key is not available.

#### 4.11.5.1.3 setValue(java.lang.String, java.lang.String)

```java
public void setValue(java.lang.String key, java.lang.String value)
throws org.atsc.security.AccessDeniedException, InvalidPreferenceException
```

This method sets a new value for the specified personal information.

Parameters:

- `key` – A key of the personal information.
- `value` – A new value for the specified key.
Throws:

org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.

InvalidPreferenceException – when the information with the specified key is not available.

4.11.5.2 Fields

The following fields are inherited from PreferenceNames: FAVORITE_CHANNELS, LANGUAGE, PERSONAL_DATA, RATING.

4.11.6 Preference

public interface Preference
extends PreferenceNames

This is a top-level interface which is common for all preference/settings subinterfaces. It supports the listener model and provides the preference name. This interface is expected to be extended to support specific preferences with specific access methods.

4.11.6.1 Methods

4.11.6.1.1 addPreferenceChangeListener(PreferenceChangeListener)

public void addPreferenceChangeListener(PreferenceChangeListener aListener)

This method allows applications interested in changes to this preference to register for preference change events.

Parameters:

aListener – A preference change listener to be notified of changes related to this preference.

4.11.6.1.2 getPreferenceName()

public java.lang.String getPreferenceName()

This method returns a unique preference name.

Returns:

A string representing the name of this preference.

See also: PreferenceNames.

4.11.6.1.3 removePreferenceChangeListener(PreferenceChangeListener)

public void removePreferenceChangeListener(PreferenceChangeListener aListener)

This method allows a preference change listener to remove itself from the list of listeners.

Parameters:

aListener – A previously registered listener.
4.11.6.2 Fields

The following fields are inherited from PreferenceNames: FAVORITE_CHANNELS, LANGUAGE, PERSONAL_DATA, RATING.

4.11.7 PreferenceChangeEvent

public class PreferenceChangeEvent
extends java.util.EventObject

A PreferenceChangeEvent is delivered whenever a preference changes a property. The getSource() method is used to determine which Preference has been changed.
4.11.8.1 Constructors

4.11.8.1.1 PreferenceChangeEvent(java.lang.Object, ...)

    public PreferenceChangeEvent(
        java.lang.Object source,
        java.lang.String key)

    Parameters:
    
    source – The Preference object which was changed.
    
    key – A string representing the changed preference value.

4.11.8.2 Methods

    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), and wait (long, int).

4.11.8.2.1 getKey()

    public java.lang.String getKey()

    This method returns the key of the property that was changed. May be null if multiple 
    preference values have changed or if no key is available.

    Note: See PersonalDataPreference for examples of preference value keys.

    Returns:

    A string representing the changed preference value.

4.11.8.3 Fields

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

4.11.9 PreferenceChangeListener

    public interface PreferenceChangeListener
    extends java.util.EventListener

    This interface is implemented by classes interested in receiving preference change events. 
    Preference change events are generated when the values of preferences are changed.

4.11.9.1 Methods

4.11.9.1.1 preferenceChanged(PreferenceChangeEvent)

    public void preferenceChanged(PreferenceChangeEvent event)

    This method is called when a preference value is changed.

    Parameters:

    event – A PreferenceChangeEvent object describing the event source and the 
    preference value that has changed.
4.11.9.2 Fields
No fields are defined.

4.11.10 PreferenceNames

    public interface PreferenceNames

This interface contains a list of predefined preference names.

4.11.10.1 Methods

4.11.10.2 Fields

4.11.10.2.1 FAVORITE_CHANNELS

    public static final java.lang.String FAVORITE_CHANNELS = "Favorite Channels"

    Favorite channels preference.

4.11.10.2.2 LANGUAGE

    public static final java.lang.String LANGUAGE = "Language"

    Preferred language preference.

4.11.10.2.3 PERSONAL_DATA

    public static final java.lang.String PERSONAL_DATA = "Personal Data"

    Personal data preference.

4.11.10.2.4 RATING

    public static final java.lang.String RATING = "Rating"

    Rating preference.

4.11.11 PreferencePermission

    public final class PreferencePermission
        extends org.atsc.security.AtscPermission

    The class PreferencePermission is used for permissions related to all security-protected
    methods in this preferences package.

    The target name is the specific preference name (e.g. "Personal Data"). Wildcards are
    allowed. The actions parameter contains a comma-separated list of the actions granted for
    the specified target. The action is either "read" for reading the specified preference value, "write"
    to change the specified preference value, "create" to add the specified preference to the
    registry, or "delete" to remove the preference from the registry.

4.11.11.1 Constructors

4.11.11.1.1 PreferencePermission(java.lang.String, java.lang.String)

    public PreferencePermission(java.lang.String name, java.lang.String action)

    Creates a new PreferencePermission object with the specified actions.
Parameters:

- **name** – the name of the preference (e.g. "Personal Data", or a wildcard ".*").
- **action** – comma-separated list of the actions granted for the specified target. The action is either "read" for reading the specified preference value, "write" to change the specified preference value, "create" to add the specified preference to the registry, or "delete" to remove the preference from the registry.

4.11.11.2 Methods

The following methods are inherited from `org.atsc.security.AtscPermission`: equals, getActions, hashCode.

The following methods are inherited from `java.security.Permission`: checkGuard, getName, newPermissionCollection, toString.

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

4.11.11.2.1 implies(java.security.Permission)

```java
public boolean implies(java.security.Permission permission)
```

Checks if this permission object implies the specified permission.

Parameters:

- **permission** – the permission to check.

Returns:

The value `true` if this object implies the specified permission.

Overrides:

```java
org.atsc.security.AtscPermission.implies(java.security.Permission)
```

4.11.11.3 Fields

No fields are defined.

4.11.12 PreferenceRegistry

```java
public interface PreferenceRegistry
extends org.atsc.registry.Registry
```

This interface represents a repository of all settings and preferences that can be shared by multiple applications.

If the `PreferenceRegistry` is obtained via the `UserProfile.getPreferences()` method, it represents the preferences of that end-user. If the `PreferenceRegistry` is obtained via the `RegistryFactory.getRegistry()` method, it represents the preferences of the current user or the common preferences when a multi-user environment is not supported.

See also: UserProfile.
4.11.12.1 Methods

The following methods are inherited from org.atsc.registry.Registry:
addRegistryChangeListener, getRegistryType, removeRegistryChangeListener.

4.11.12.1.1 addPreference(Preference)

```java
public void addPreference(Preference preference)
throws org.atsc.security.AccessDeniedException,
InvalidPreferenceException
```

This method allows an application to insert a new preference object into the repository. The new preference must not be already associated with this repository.

Parameters:

`preference` – The new preference to be added to the repository.

Throws:

org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.

InvalidPreferenceException – when this preference is already in the repository.

4.11.12.1.2 getPreference(java.lang.String)

```java
public Preference[] getPreference(java.lang.String preferenceName)
throws InvalidPreferenceException
```

This method returns the preference(s) of the specified name. In most cases, there is only one preference corresponding to a given preference name; however, other preferences, such as FavoriteChannelsPreference, may contain more than one object, in which case they identified by their FavoriteServiceName.

Parameters:

`preferenceName` – The name of the requested preference.

Returns:

Array of the specified Preference objects.

Throws:

InvalidPreferenceException – if the preference name is unknown to the repository.

See also: PreferenceNames.

4.11.12.1.3 listPreferences()

```java
public Preference[] listPreferences()
```

This method returns a list of all Preferences currently stored in the repository.

Returns:

An array of preferences currently contained in the repository.

4.11.12.1.4 removePreference(Preference)

```java
public void removePreference(Preference preference)
throws org.atsc.security.AccessDeniedException,
InvalidPreferenceException
```
This method allows an application to remove a preference object from the repository.

Parameters:

preference

– The preference to be removed from the repository.

Throws:

org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.

InvalidPreferenceException – when this preference does not exist in the repository.

4.11.12.2 Fields

No fields are defined.

4.11.13 PreferenceRegistryEvent

public class PreferenceRegistryEvent
extends org.atsc.registry.RegistryChangeEvent

This event informs the preference registry listener about changes in the PreferenceRegistry.

4.11.13.1 Constructors

4.11.13.1.1 PreferenceRegistryEvent(java.lang.Object, ...)

public PreferenceRegistryEvent(
java.lang.Object source,
PreferenceChangeCause cause,
Preference preference)

Parameters:

source

– The object that caused this event.

cause

– The cause of this event.

preference

– The preference object that caused the change in the Preference registry.

4.11.13.2 Methods

The following methods are inherited from org.atsc.registry.RegistryChangeEvent:

cause, registryType.

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

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

4.11.13.2.1 getPreference()

public Preference getPreference()

Returns the preference that caused the change in the repository. This change concerns the repository, such as adding or removing a Preference from the PreferenceRepository, not a change in the value of the Preference.
Returns:

The Preference that was added to or removed from the Preference repository.

### 4.11.13.3 Fields

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

### 4.11.14 RatingPreference

```java
class RatingPreference extends Preference
```

This interface represents parental rating settings.

### 4.11.14.1 Methods

The following methods are inherited from Preference: addPreferenceChangeListener, getPreferenceName, removePreferenceChangeListener.

#### 4.11.14.1.1 isBlocked(javax.tv.service.guide.ContentRatingAdvisory)

```java
public boolean isBlocked(javax.tv.service.guide.ContentRatingAdvisory rating)
```

This method indicates whether the specified rating satisfied or violates the current parental rating settings.

Parameters:

- `rating` – Content rating advisory obtained from a specific program event.

Returns:

- The value `true` when a program event with the specified rating will be blocked due to the current parental rating settings; otherwise, `false` when the event will not be blocked.

See also: javax.tv.service.guide.ProgramEvent.

### 4.11.14.2 Fields

The following fields are inherited from PreferenceNames: FAVORITE_CHANNELS, LANGUAGE, PERSONAL_DATA, RATING.

### 4.12 org.atsc.registry

This package provides a set of classes and interfaces that provide persistent registry functionality.

**Note:** The determination of the degree of persistence of a registry’s state is implementation dependent.

#### 4.12.1 Registry

```java
public interface Registry
```

This interface provides a base interface for all specialized registry interfaces. It is provided so that the RegistryFactory can return a base type.
4.12.1.1 Methods

4.12.1.1.1 addRegistryChangeListener(RegistryChangeListener)

```java
public void addRegistryChangeListener(RegistryChangeListener listener)
```

Called to register for events generated by the Registry.

Parameters:

- `listener` – object registering for Registry events.

See also: RegistryChangeEvent.

4.12.1.1.2 getRegistryType()

```java
public RegistryType getRegistryType()
```

Called to determine the type of registry implemented by the object returned by the method RegistryFactory.getRegistry().

Returns:

An object representing the type of registry.

See also: RegistryType.

4.12.1.1.3 removeRegistryChangeListener(RegistryChangeListener)

```java
public void removeRegistryChangeListener(RegistryChangeListener listener)
```

Called to deregister for events generated by the Registry.

Parameters:

- `listener` – object previously registered as a listener for Registry events.

4.12.1.2 Fields

No fields are defined.

4.12.2 RegistryChangeCause

```java
public abstract class RegistryChangeCause
    extends java.lang.Object
```

An abstract class that defines the common type for all registry change cause classes. Each registry class will have an associated change cause class which defines the reasons for changes to that registry.

See also: RegistryChangeEvent.

4.12.2.1 Constructors

4.12.2.1.1 RegistryChangeCause(java.lang.String)

```java
protected RegistryChangeCause(java.lang.String nameString)
```

Parameters:

- `nameString` – The change cause represented as a string.
4.12.2.2  Methods

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

4.12.2.3  Fields

No fields are defined.

4.12.3  RegistryChangeEvent

```java
class RegistryChangeEvent extends java.util.EventObject
```

A generic registry change event which is extended by all specific registries in order to provide specific information about a change.

4.12.3.1  Constructors

4.12.3.1.1  `RegistryChangeEvent(java.lang.Object, ...)`

```java
public RegistryChangeEvent(
    java.lang.Object source,
    RegistryType registryType,
    RegistryChangeCause cause)
```

Parameters:

- `source` – The object that caused this event.
- `registryType` – The type of the registry that caused this event.
- `cause` – The specific cause of this event.

4.12.3.2  Methods

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)`, and `wait(long,int)`.

4.12.3.2.1  `getCause()`

```java
public RegistryChangeCause getCause()
```

Returns the cause of the `RegistryChangeEvent`. Each registry will define a set of causes appropriate for the registry it represents.

Returns:

- An object representing the cause of this event.

4.12.3.2.2  `getRegistryType()`

```java
public RegistryType getRegistryType()
```

Returns the type of a registry.
Returns:
A string representing the type of the registry.
See also: RegistryType.

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

4.12.4 RegistryChangeListener

public interface RegistryChangeListener
extends java.util.EventListener

This interface allows an object to listen to changes made to the Registry or events generated by the Registry.

4.12.4.1 Methods

4.12.4.1.1 registryChanged(RegistryChangeEvent)

public void registryChanged(RegistryChangeEvent event)

This method is called when a RegistryChangeEvent is generated.
Parameters:

  event  -- The event that is to be delivered to the listener.

4.12.4.2 Fields
No fields are defined.

4.12.5 RegistryFactory

public class RegistryFactory
extends java.lang.Object

This class provides a mechanism to create objects that implement specific Registry interfaces, such as XletRegistry.

4.12.5.1 Constructors

4.12.5.1.1 RegistryFactory()

protected RegistryFactory()

A protected constructor that creates a class representing the RegistryFactory.

4.12.5.2 Methods

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

public Registry getRegistry(RegistryType registryType)
    throws org.atsc.security.AccessDeniedException

Returns an instance of an object which implements the specified registry interface. The type of the returned object will be one of the derived Registry types, such as XletRegistry.

Parameters:
registryType – the type of registry.

Returns:
A reference to an existing Registry of the specified type. Returns null when specified registry does not exist or cannot be created.

Throws:
org.atsc.security.AccessDeniedException – when the caller does not have a permission to access the specified registry.

4.12.5.3 Fields

No fields are defined.

4.12.6 RegistryType

public class RegistryType
    extends java.lang.Object

This class defines names for different registry types, such as a user preference registry, an Xlet registry, etc.

4.12.6.1 Constructors

4.12.6.1.1 RegistryType(java.lang.String)

protected RegistryType(java.lang.String name)

Protected constructor.

Parameters:
name – registry type name.

4.12.6.2 Methods

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

4.12.6.3 Fields

4.12.6.3.1 XLET_REGISTRY

public static final RegistryType XLET_REGISTRY

Xlet registry.
4.12.6.3.2 **PREFERENCE_REGISTRY**

    public static final RegistryType PREFERENCE_REGISTRY
    Preference registry.

4.12.6.3.3 **USER_REGISTRY**

    public static final RegistryType USER_REGISTRY
    User registry.

4.13 **org.atsc.security**

    This package provides classes and interfaces that supporting the certain security facilities.

4.13.1 **AccessDeniedException**

    public class AccessDeniedException
    extends java.security.AccessControlException

    This exception signals that the caller does not have enough privileges to perform the
    requested operation.

4.13.1.1 **Constructors**

4.13.1.1.1 **AccessDeniedException()**

    public AccessDeniedException()
    Constructor with no detail message.

4.13.1.1.2 **AccessDeniedException(java.lang.String)**

    public AccessDeniedException(java.lang.String reason)
    Constructor taking a detail message.
    Parameters:
    reason – The reason this exception was thrown.

4.13.1.2 **Methods**

    The following methods are inherited from java.lang.Throwable: fillInStackTrace,
    getLocalizedMessage, getMessage, printStackTrace(), printStackTrace(java.io.-
    PrintStream), printStackTrace(java.io.PrintWriter), toString.
    The following methods are inherited from java.lang.Object: clone, equals, finalize,
    getClass, hashCode, notify, notifyAll, wait(), wait(long), and wait(long, int).

4.13.1.3 **Fields**

    No fields are defined.

4.13.2 **AtscAllPermission**

    public final class AtscAllPermission
    extends AtscPermission
The AtscAllPermission is a permission that implies all other permissions derived from AtscPermission.

See also: java.security.AllPermission, java.lang.SecurityManager.

4.13.2.1 Constructors

4.13.2.1.1 AtscAllPermission()
public AtscAllPermission()

Creates a new AtscAllPermission.

4.13.2.2 Methods

The following methods are inherited from org.atsc.security.AtscPermission: equals, getActions, hashCode.

The following methods are inherited from java.security.Permission: checkGuard, getName, newPermissionCollection, toString.

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

4.13.2.2.1 implies(java.security.Permission)
public boolean implies(java.security.Permission permission)

Checks if this permission object implies the specified permission.

Parameters:
permission – the permission to check.

Returns:

The value true if this object implies the specified permission.

Overrides:
AtscPermission.implies(java.security.Permission)

4.13.2.3 Fields

No fields are defined.

4.13.3 AtscPermission

public abstract class AtscPermission extends java.security.Permission

The AtscPermission class extends the java.security.Permission class, and is used as the base class for permissions related to APIs defined by this specification.

See also: java.lang.SecurityManager, java.security.Permission.

4.13.3.1 Constructors

4.13.3.1.1 AtscPermission(java.lang.String)
protected AtscPermission(java.lang.String name)
Creates a new `AtscPermission`. 

Parameters:

`name` – provided to superclass.

4.13.3.1.2 `AtscPermission(java.lang.String, java.lang.String)`

protected `AtscPermission(java.lang.String name, java.lang.String action)`

Creates a new `AtscPermission`. 

Parameters:

`name` – name of permission, provided to superclass.

`action` – action(s) to be permitted.

4.13.3.2 Methods

The following methods are inherited from `java.security.Permission`: `checkGuard`, `getName`, `newPermissionCollection`, `toString`.

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

4.13.3.2.1 `equals(java.security.Permission)`

public boolean `equals(java.lang.Object obj)`

Checks a permission object for equivalence with this object.

Parameters:

`obj` – the object being tested for equivalence with this object.

Returns:

The value `true` if the specified object is an equivalent permission.

Overrides:

`java.security.Permission.equals(java.lang.Object)`

4.13.3.2.2 `getActions()`

public `java.lang.String getActions()`

Obtain the actions of this permission object.

Returns:

A string representation of this permission object's actions.

Overrides:

`java.security.Permission.getActions()`

4.13.3.2.3 `hashCode()`

public `int hashCode()`

Obtain the hash code value of this permission object.

Returns:

A hash code value for this permission object.
4.13.3.2.4 overridden(java.security.Permission)

public boolean implies(java.security.Permission permission)

Checks if this permission object implies the specified permission.

Parameters:
permission – the permission to check.

Returns:
the value true if this object implies the specified permission.

Overrides:
java.security.Permission.implies(java.security.Permission)

4.13.3.3 Fields

No fields are defined.

4.13.4 HAViPermission

public final class HAViPermission
extends AtscPermission

This class is for used for certain HAVi related functions.

Note: See [HAVI-UI-API] for more information about these functions.

The following table lists all the possible target names and function controlled by the specific target name.

<table>
<thead>
<tr>
<th>Permission Target Name</th>
<th>What the Permission Allows</th>
</tr>
</thead>
<tbody>
<tr>
<td>setBackgroundConfiguration</td>
<td>Setting a device's background configuration with HBackgroundDevice.setBackgroundConfiguration().</td>
</tr>
<tr>
<td>setVideoConfiguration</td>
<td>Setting a video configuration with HVideoDevice.setVideoConfiguration().</td>
</tr>
<tr>
<td>setCoherentScreenConfigurations</td>
<td>Modifying an HScreenDevice with a set of HScreenConfigurations, with HScreen.setCoherentScreenConfigurations().</td>
</tr>
</tbody>
</table>

4.13.4.1 Constructors

4.13.4.1.1 HAViPermission(java.lang.String)

public HAViPermission(java.lang.String name)
Creates a new HAViPermission with the specified name. The name is the symbolic name of the HAViPermission, such as "setVideoConfiguration", "setGraphicsConfiguration", etc.

Parameters:

name – the name of the HAViPermission as specified by Table 1 HAViPermission Targets.

4.13.4.1.2 HAViPermission(java.lang.String, java.lang.String)

public HAViPermission(java.lang.String name, java.lang.String action)

Creates a new HAViPermission object with the specified name.

Parameters:

name – the name of the HAViPermission as specified by Table 1 HAViPermission Targets.

action – ignored.

4.13.4.2 Methods

The following methods are inherited from AtscPermission: equals, getActions, hashCode.

The following methods are inherited from java.security.Permission: checkGuard, getName, newPermissionCollection, toString.

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

4.13.4.2.1 implies(java.security.Permission)

public boolean implies(java.security.Permission permission)

Checks if this permission object implies the specified permission.

Parameters:

permission – the permission to check.

Returns:

The value true if this object implies the specified permission.

Overrides:

AtscPermission.implies(java.security.Permission)

4.13.4.3 Fields

No fields are defined.

4.14 org.atsc.system

This package represents classes and interfaces that characterize the DASE System and its underlying receiver platform.
4.14.1 Receiver

```java
public class Receiver
    extends java.lang.Object
    implements org.atsc.management.ObjectStates
```

Class representing the DASE System and its underlying receiver platform.

See also: ObjectStates, AvailabilityStatus.

### 4.14.1.1 Constructors

4.14.1.1.1 Receiver()

```java
protected Receiver()
```

Protected constructor.

### 4.14.1.2 Methods

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

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

The following methods are inherited from `org.atsc.management.ObjectStates`:

- `addStateChangeListener`, `getCurrentState`, `getCurrentStatus`, `getStatesSupported`, `removeStateChangeListener`.

The following methods are inherited from `org.atsc.management.AlarmStatus`:

- `clearAlarm`, `getAlarmStatus`.

The following methods are inherited from `org.atsc.management.ProceduralStatus`:

- `getProceduralStatus`.

The following methods are inherited from `org.atsc.management.AvailabilityStatus`:

- `getAvailabilityStatus`.

The following methods are inherited from `org.atsc.management.OperationalState`:

- `getOperationalState`.

The following methods are inherited from `org.atsc.management.AdministrativeState`:

- `getAdministrativeState`, `setLock`.

4.14.1.2.1 getInstance()

```java
public static Receiver getInstance()
```

Retrieve the `Receiver` object.

Returns:

- The receiver instance.

### 4.14.1.3 Fields

The following fields are inherited from `org.atsc.management.ObjectStates`: `NOSTATUS`.

The following fields are inherited from `org.atsc.management.AlarmStatus`:

- `ALARM_OUTSTANDING`, `ALARM_STATUS_ID`, `CRITICAL`, `MAJOR`, `MINOR`, `UNDER_REPAIR`.
The following fields are inherited from `org.atsc.management.ProceduralStatus`:

- INIT_REQUIRED
- INITIALIZING
- NOT_INITIALIZED
- PROCEDURAL_STATUS_ID
- REPORTING
- TERMINATING.

The following fields are inherited from `org.atsc.management.AvailabilityStatus`:

- AVAILABILITY_STATUS_ID
- DEGRADED
- DEPENDENCY
- FAILED
- INTEST
- LOG_FULL
- NOT_INSTALLED
- OFFDUTY
- OFFLINE
- POWEROFF.

The following fields are inherited from `org.atsc.management.UsageState`:

- ACTIVE
- BUSY
- IDLE
- USAGE_STATE_ID.

The following fields are inherited from `org.atsc.management.OperationalState`:

- DISABLED
- ENABLED
- OPERATIONAL_STATE_ID.

The following fields are inherited from `org.atsc.management.AdministrativeState`:

- ADMIN_STATE_ID
- LOCKED
- SHUTTING_DOWN
- UNLOCKED.

### 4.14.2 ReceiverPropertyNames

```java
public interface ReceiverPropertyNames
{

    // This interface defines a set of DASE specific system properties. They are used as keys to
    // retrieve their value using java.lang.System.getProperty(key).

    // Note: See [DASE-PA], Annex C, Java System Properties, for more information on the
    // required or recommended values for these properties.

    // Methods

    // Fields

    // DASE_DELIVERY_SYSTEM
    public static final java.lang.String DASE_DELIVERY_SYSTEM =
        "dase.delivery.system"

    // DASE_IMPLEMENTATION_LEVEL
    public static final java.lang.String DASE_IMPLEMENTATION_LEVEL =
        "dase.implementation.level"

    // DASE_IMPLEMENTATION_NAME
    public static final java.lang.String DASE_IMPLEMENTATION_NAME =
        "dase.implementation.name"

    // DASE_IMPLEMENTATION_VENDOR
    public static final java.lang.String DASE_IMPLEMENTATION_VENDOR =
        "dase.implementation.vendor"

    // DASE_IMPLEMENTATION_VERSION
    public static final java.lang.String DASE_IMPLEMENTATION_VERSION =
        "dase.implementation.version"
}
4.14.2.2.6 **DASE_SPECIFICATION_NAME**

public static final java.lang.String DASE_SPECIFICATION_NAME =
"dase.specification.name"

4.14.2.2.7 **DASE_SPECIFICATION_VENDOR**

public static final java.lang.String DASE_SPECIFICATION_VENDOR =
"dase.specification.vendor"

4.14.2.2.8 **DASE_SPECIFICATION_VERSION**

public static final java.lang.String DASE_SPECIFICATION_VERSION =
"dase.specification.version"

4.15 **org.atsc.trigger**

Provides APIs for Xlets to receive triggers from the broadcast.

Triggers are brief messages sent in the broadcast that cause the generation of events on which applications may loosely synchronize their behavior. Trigger messages follow a format indicated by their trigger type. The APIs of this package are driven by triggers that belong to a single, known type. Such triggers generate trigger events encapsulating a target and a property list of key/value pairs. The target value may be used by the Xlet to identify the purpose or scope of the TriggerEvent; the property list may be used to communicate an arbitrary set of application-specific String data.

A TriggerSource object represents a broadcast data stream that carries triggers. Receipt of a trigger at a TriggerSource causes one or more TriggerEvent objects to be generated and dispatched. The TriggerSource class provides mechanisms by which applications implementing the TriggerListener interface may subscribe to receive TriggerEvents matching one or more target values.

Note: See [DASE-PA], Section 4.3, Trigger Processing, for further information on trigger dispatching.

4.15.1 **TriggerEvent**

public class TriggerEvent
extends java.util.EventObject

A TriggerEvent represents an event that is generated (i.e., "triggered") as the result of the receipt of a trigger message from a TriggerSource. Each trigger event is associated with a specific target value. TriggerListener objects may filter the receipt of TriggerEvent objects based on one or more target values. Additionally, TriggerEvent objects carry a table of arbitrary string properties that may be retrieved by a string key.

4.15.1.1 **Constructors**

4.15.1.1.1 **TriggerEvent(TriggerSource,java.lang.String,java.util.Properties)**

public TriggerEvent(
TriggerSource source,
java.lang.String target,
java.util.Properties properties)

Creates a new TriggerEvent object. The resulting TriggerEvent will carry a copy of the specified property list, made via properties.clone().
Note: See [DASE-PA], Section 4.3.2, Generic Event Processing, for information on the construction of a TriggerEvent.

Parameters:

source – The TriggerSource from which the event was generated.

target – The target value of the event.

properties – A property list of key/value string pairs.

4.15.1.2 Methods

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

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

4.15.1.2.1 getProperties()

public java.util.Properties getProperties()

Retrieves the properties of event. The returned properties object shall be a deep clone of the event’s properties.

Returns:

A Properties object.

4.15.1.2.2 getProperty(java.lang.String)

public java.lang.String getProperty(java.lang.String key)

Retrieves the property string associated with the specified key.

Parameters:

key – The string by which to retrieve the desired property.

Returns:

The property string corresponding to key, or null if no property is associated with key.

Note: See [DASE-PA], Section 4.3.2, Generic Event Processing, for information on how trigger event properties are derived.

Throws:

java.lang.IllegalArgumentException – If key is null.

4.15.1.2.3 getTarget()

public java.lang.String getTarget()

Reports the target value of the event.

Returns:

The string target value.

4.15.1.2.4 getTriggerSource()

public TriggerSource getTriggerSource()

Reports the TriggerSource of the event.
Returns:
The source of the event.

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

4.15.2 TriggerListener

public interface TriggerListener
extends java.util.EventListener

The interface TriggerListener defines a mechanism by which applications may receive trigger events of interest, as previously subscribed at a TriggerSource.

4.15.2.1 Methods

4.15.2.1.1 eventTriggered(TriggerEvent)

public void eventTriggered(TriggerEvent triggerEvent)

Notifies the TriggerListener when a TriggerEvent of interest is posted.
Parameters:
triggerEvent – The TriggerEvent that was posted.

4.15.2.2 Fields
No fields are defined.

4.15.3 TriggerSource

public abstract class TriggerSource
extends java.lang.Object

The class TriggerSource represents a source of trigger events obtained from a broadcast stream. Each TriggerSource object is uniquely identified by a Locator. The TriggerSource class provides mechanisms by which applications implementing the TriggerListener interface may subscribe to receive trigger events matching one or more target values.

Xlets may be associated with a single source of triggers through application signaling. In such a case, the Xlet may obtain the corresponding TriggerSource object through its XletContext.

See also: javax.tv.xlet.XletContext.

4.15.3.1 Constructors

4.15.3.1.1 TriggerSource()

protected TriggerSource()
Protected constructor.
4.15.3.2 Methods

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

4.15.3.2.1 addListener(TriggerListener)

    public abstract void addListener(TriggerListener listener)

Subscribes the specified TriggerListener to receive trigger events of all target values. If listener is already subscribed using this method, no action is performed. If listener is already subscribed to receive events of one or more specific target values, it is first unsubscribed as in the method removeListener(TriggerListener), and then re-subscribed to receive events of all target values.

Parameters:

listener – The TriggerListener to which to send subsequent trigger events.

Throws:

java.lang.IllegalArgumentException – If listener is null.

4.15.3.2.2 addListener(TriggerListener, java.lang.String)

    public abstract void addListener(TriggerListener listener, java.lang.String target)

Subscribes the specified TriggerListener to receive trigger events of the specified target value. Trigger events whose corresponding target value does not match the specified target value will not be posted. If listener is already subscribed using the method addListener(TriggerListener), no action is performed. If listener is already subscribed to receive events of a target value equal to target, no action is performed.

Parameters:

listener – The TriggerListener to which to send subsequent trigger events.

target – The target value of the trigger on which to filter.

Throws:

java.lang.IllegalArgumentException – If listener or target is null.

4.15.3.2.3 getLocator()

    public javax.tv.locator.Locator getLocator()

Provides a Locator that uniquely identifies this TriggerSource.

Returns:

A Locator referencing the TriggerSource.

4.15.3.2.4 getTriggerSource(javax.tv.locator.Locator)

    public TriggerSource getTriggerSource(javax.tv.locator.Locator locator)
    throws javax.tv.locator.InvalidLocatorException

Gets the TriggerSource referenced by the given Locator.

Parameters:

locator – A locator referencing a broadcast data stream carrying triggers.
Returns:

The TriggerSource referenced by the given Locator.

Throws:

javax.tv.locator.InvalidLocatorException – If locator does not reference a valid source of trigger events.

4.15.3.2.5 removeListener(TriggerListener)

public abstract void removeListener(TriggerListener listener)

Unsubscribe the specified TriggerListener from receiving trigger events of any target value.

Parameters:

listener – The TriggerListener to unsubscribe.

Throws:

java.lang.NullPointerException – If listener is null.

4.15.3.2.6 removeListener(TriggerListener, java.lang.String)

public abstract void removeListener(TriggerListener listener, java.lang.String target)

Unsubscribe the specified TriggerListener from receiving trigger events of the specified target value.

Parameters:

listener – The TriggerListener to unsubscribe.

target – The target value for which to unsubscribe the TriggerListener.

Throws:

java.lang.IllegalArgumentException – If listener or target is null.

4.15.3.3 Fields

4.15.3.3.1 KEY

public static final java.lang.String KEY = "org.atsc.trigger.source.default"

Defines a key to be used to retrieve a TriggerSource object provided for an Xlet. The TriggerSource object is retrieved via XletContext.getXletProperty(TriggerSource.KEY).

See also: javax.tv.xlet.XletContext.

4.16 org.atsc.user

This package provides classes and interfaces necessary for end-user management functions such as obtaining a list of defined end-users and getting information about the current end-user.

4.16.1 InvalidCapabilityException

public class InvalidCapabilityException

extends java.lang.Exception
This exception signals that a capability of this name is invalid in the current context.

4.16.1 Constructors

4.16.1.1 InvalidCapabilityException()

    public InvalidCapabilityException()
    Constructor with no detail message.

4.16.1.1.2 InvalidCapabilityException(java.lang.String)

    public InvalidCapabilityException(java.lang.String reason)
    Constructor taking a detail message.
    Parameters:
    reason – The reason this exception was thrown.

4.16.2 Methods

    The following methods are inherited from java.lang.Throwable: fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace(), printStackTrace(java.io.-PrintStream), printStackTrace(java.io.PrintWriter), toString.
    The following methods are inherited from java.lang.Object: clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait(), wait(long), and wait(long, int).

4.16.1.3 Fields

    No fields are defined.

4.16.2 InvalidUserException

    public class InvalidUserException
    extends java.lang.Exception

    This exception signals that a preference of this name is invalid in the current context (e.g. already present in the repository and cannot be inserted again, or does not exist and cannot be retrieved, etc.)

4.16.2.1 Constructors

4.16.2.1.1 InvalidUserException()

    public InvalidUserException()
    Constructor with no detail message.

4.16.2.1.2 InvalidUserException(java.lang.String)

    public InvalidUserException(java.lang.String reason)
    Constructor taking a detail message.
    Parameters:
    reason – The reason this exception was thrown.
4.16.2.2 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)`.

4.16.2.3 Fields

No fields are defined.

4.16.3 UserCapabilities

`public interface UserCapabilities`

This interface defines a list of user capabilities that can be conferred upon a user. It is currently empty and serves as a placeholder for future definitions.

4.16.3.1 Methods

4.16.3.2 Fields

No fields are defined.

4.16.4 UserChangeCause

`public class UserChangeCause`

`extends org.atsc.registry.RegistryChangeCause`

This class defines the possible causes for the `UserRegistryEvent`.

See also: `UserRegistryEvent`.

4.16.4.1 Constructors

4.16.4.1.1 `UserChangeCause(java.lang.String)`

`protected UserChangeCause(java.lang.String nameString)`

Parameters:

`nameString` – the change cause represented as a string.

4.16.4.2 Methods

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

4.16.4.3 Fields

4.16.4.3.1 `NEW_CURRENT_USER`

`public static final UserChangeCause NEW_CURRENT_USER`
A new user became active (or current).

### 4.16.4.3.2 USER_ADDED

```java
public static final UserChangeCause USER_ADDED
```

A new user was added to the User Registry.

### 4.16.4.3.3 USER_REMOVED

```java
public static final UserChangeCause USER_REMOVED
```

An existing user was removed from the User Registry.

### 4.16.5 UserPermission

```java
public final class UserPermission
    extends org.atsc.security.AtscPermission
```

The `UserPermission` class is used for permissions related to all security-protected methods in this package.

The target name may be "user" with action "create", "delete", "read", "write", which covers the `UserRegistry` methods, or the name may be a user capability (from the `UserCapabilities` interface) with actions "confer" or "retract", which covers the `UserProfile` methods.

The actions parameter contains a comma-separated list of the actions granted for the specified target.

See also: `UserCapabilities`.

### 4.16.5.1 Constructors

### 4.16.5.1.1 UserPermission(java.lang.String, java.lang.String)

```java
public UserPermission(java.lang.String name, java.lang.String action)
```

A `UserPermission` is used for permissions related to all security-protected methods in this user package.

Parameters:

- `name` – the target name may be "user" with action "create", "delete", "read", "write", which covers the `UserRegistry` methods, or the name may be a user capability (from the `UserCapabilities` interface) with actions "confer" or "retract", which covers the `UserProfile` methods.

- `action` – contains a comma-separated list of the actions granted for the specified target. See the name parameter for details.

See also: `UserCapabilities`.

### 4.16.5.2 Methods

The following methods are inherited from `org.atsc.security.AtscPermission`: `equals`, `getActions`, `hashCode`.

The following methods are inherited from `java.security.Permission`: `checkGuard`, `getName`, `newPermissionCollection`, `toString`. 
The following methods are inherited from java.lang.Object: clone, finalize, getClass, notify, notifyAll, wait(), wait(long), and wait(long, int).

4.16.5.2.1 implies(java.security.Permission)

public boolean implies(java.security.Permission permission)

Checks if this permission object implies the specified permission.

Parameters:
permission – the permission to check.

Returns:
The value true if this object implies the specified permission.

Overrides:
org.atsc.security.AtscPermission.implies(java.security.Permission)

4.16.5.3 Fields

No fields are defined.

4.16.6 UserProfile

public interface UserProfile
extends UserCapabilities

This interface represents a container of information about a single user, such as name, settings and preferences, etc.

4.16.6.1 Methods

4.16.6.1.1 authenticate()

public boolean authenticate()

This method is called to authenticate a user. It invokes an implementation specific mechanism for user authentication. It may use a user dialog to ask for a password or PIN, or other authentication mechanisms.

Returns:
The value true if authentication succeeded; otherwise, false.

4.16.6.1.2 conferCapability(java.lang.String)

public void conferCapability(java.lang.String newCapability)
throws org.atsc.security.AccessDeniedException, InvalidCapabilityException

This method is called to confer (add) a new capability upon the user.

Parameters:
newCapability – The name of the capability to be conferred upon this user.

Throws:
org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.
InvalidCapabilityException – when the specified capability does not exist.
See also: UserCapabilities.

4.16.6.1.3  getName()

public java.lang.String getName()

Returns the name of this user.

Returns:
A string representing the name of this user.

4.16.6.1.4  getPreferences()

public org.atsc.preferences.PreferenceRegistry getPreferences()

This method returns this user’s PreferenceRegistry. All operations performed on the returned list of preferences will be reflected in this UserProfile.

Returns:
The PreferenceRegistry associated with this user.

4.16.6.1.5  retractCapability(java.lang.String)

public void retractCapability(java.lang.String capabilityName)
throws org.atsc.security.AccessDeniedException,
InvalidCapabilityException

This method is called to retract (remove) an existing capability from the user.
Parameters:
capabilityName – The name of the capability to be retracted from this user.

Throws:
org.atsc.security.AccessDeniedException – when the caller does not have sufficient permission.
InvalidCapabilityException – when the specified capability does not exist.
See also: UserCapabilities.

4.16.6.2  Fields

No fields are defined.

4.16.7  UserRegistry

public interface UserRegistry
extends org.atsc.registry.Registry

The UserRegistry interface provides access to all users defined on the system.

4.16.7.1  Methods

The following methods are inherited from org.atsc.registry.Registry:
addRegistryChangeListener, getRegistryType, removeRegistryChangeListener.
4.16.7.1.1 **createUser**(java.lang.String name)

```java
public void createUser(java.lang.String name)
throws org.atsc.security.AccessDeniedException,
InvalidUserException
```

This method will create a new user of the specified name. It is expected that the receiver will associate the new user with an authentication mechanism (e.g. a PIN code).

Parameters:

`name` – The name of the new user.

Throws:

- `org.atsc.security.AccessDeniedException` – when the caller does not have permission to create users.
- `InvalidUserException` – when the specified user already exists.

4.16.7.1.2 **deleteUser**(java.lang.String name)

```java
public void deleteUser(java.lang.String name)
throws org.atsc.security.AccessDeniedException,
InvalidUserException
```

This method will delete a specified user from the registry.

Parameters:

`name` – The name of the user to be deleted.

Throws:

- `org.atsc.security.AccessDeniedException` – when the caller does not have permission to delete users.
- `InvalidUserException` – when the specified user does not exist or is currently active.

4.16.7.1.3 **getCurrentUser**()

```java
public UserProfile getCurrentUser()
```

This method returns the user who is currently registered as the active user.

Returns:

- The `UserProfile` of the current user.

4.16.7.1.4 **getUser**(java.lang.String name)

```java
public UserProfile getUser(java.lang.String name)
throws org.atsc.security.AccessDeniedException,
InvalidUserException
```

This method returns a `UserProfile` object of the specified name.

Parameters:

`name` – The name of the user of interest.

Returns:

- The `UserProfile` of the specified user.
Throws:

org.atsc.security.AccessDeniedException – when the caller does not have permission to access the specified user profile.

InvalidUserException – when such user does not exist.

4.16.7.1.5 getUserNames()

public java.lang.String[] getUserNames()

This method returns a list of all known users in the registry.

Returns:

An array of strings representing all registered users.

4.16.7.1.6 setCurrentUser(java.lang.String newUser)

public void setCurrentUser(java.lang.String name)
throws org.atsc.security.AccessDeniedException, InvalidUserException

This method defines a new current user. The implementation may prompt the user for a password or some other method of authentication. Setting a new current user makes the user who was current until now inactive.

Parameters:

newUser – Name of the newly active user.

Throws:

org.atsc.security.AccessDeniedException – when the called does not have the permission to change the current user.

InvalidUserException – when the specified user does not exist or cannot be made current (e.g. authentication failed).

4.16.7.2 Fields

No fields are defined.

4.16.8 UserRegistryEvent

public class UserRegistryEvent
extends org.atsc.registry.RegistryChangeEvent

This event indicates a change in the UserRegistry to all active user registry listeners.

4.16.8.1 Constructors

4.16.8.1.1 UserRegistryEvent(java.lang.Object, ...)

public UserRegistryEvent(
    java.lang.Object source,
    UserChangeCause cause,
    java.lang.String userName)

Parameters:

source – The object that caused this event.
cause – The cause of the UserRegistryEvent.

userName – The name of the end-user that caused the event to be sent.

4.16.8.2 Methods

The following methods are inherited from org.atsc.registry.RegistryChangeEvent: getCause, getRegistryType.

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), and wait(long, int).

4.16.8.2.1 getUserName()

public java.lang.String getUserName()

Returns the name of the end-user that this event is in reference to.

Returns:

A string representing the name of the affected user.

4.16.8.3 Fields

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

4.17 org.atsc.xlet

This package includes classes and interfaces related to the management of Xlets.

4.17.1 InvalidXletException

public class InvalidXletException
extends XletAvailabilityException

This exception is thrown when an invalid Xlet is specified. Some reasons may include cases where the specified locator is invalid or the location is not accessible, or the Xlet no longer exists.

4.17.1.1 Constructors

4.17.1.1.1 InvalidXletException()

public InvalidXletException()

Constructor with no detail message.

4.17.1.1.2 InvalidXletException(java.lang.String)

public InvalidXletException(java.lang.String reason)

Constructor taking a detail message.

Parameters:

reason – the reason this exception was thrown
4.17.1.2 Methods

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

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

4.17.1.3 Fields

No fields are defined.

4.17.2 XletAlreadyRegisteredException

public class XletAlreadyRegisteredException extends XletAvailabilityException

This exception is thrown when an attempt is made to register an already registered Xlet.

4.17.2.1 Constructors

4.17.2.1.1 XletAlreadyRegisteredException()

public XletAlreadyRegisteredException()

Constructor with no detail message.

4.17.2.1.2 XletAlreadyRegisteredException(java.lang.String)

public XletAlreadyRegisteredException(java.lang.String reason)

Constructor taking a detail message.

Parameters:

reason – the reason this exception was thrown

4.17.2.2 Methods

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

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

4.17.2.3 Fields

No fields are defined.

4.17.3 XletAvailabilityException

public class XletAvailabilityException extends java.lang.Exception

This exception is thrown when the requested Xlet availability condition was violated. For instance, if a method on an XletProxy is called but the Xlet that proxy represents no longer exists, this exception is thrown.
4.17.3.1 Constructors

4.17.3.1.1 XletAvailabilityException()

    public XletAvailabilityException()
    Constructor with no detail message.

4.17.3.1.2 XletAvailabilityException(java.lang.String)

    public XletAvailabilityException(java.lang.String reason)
    Constructor taking a detail message.
    Parameters:
    reason – the reason this exception was thrown

4.17.3.2 Methods

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

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

4.17.3.3 Fields

    No fields are defined.

4.17.4 XletChangeCause

    public class XletChangeCause
    extends org.atsc.registry.RegistryChangeCause
    
    This class defines the possible causes of an XletRegistryEvent.
    
    See also: XletRegistryEvent.

4.17.4.1 Constructors

4.17.4.1.1 XletChangeCause(java.lang.String)

    protected XletChangeCause(java.lang.String nameString)
    
    Protected constructor.
    Parameters:
    nameString – the change cause represented as a string.

4.17.4.2 Methods

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

4.17.4.3.1 DEREGISTERED

public static final XletChangeCause DEREGISTERED

Xlet was deregistered in the repository.

4.17.4.3.2 FAILED

public static final XletChangeCause FAILED

The system failed to start the Xlet.

4.17.4.3.3 REGISTERED

public static final XletChangeCause REGISTERED

Xlet was registered in the repository.

4.17.4.3.4 STARTED

public static final XletChangeCause STARTED

Xlet was started.

4.17.5 XletComponentPresenterProxy

public interface XletComponentPresenterProxy
extends XletProxy, javax.tv.service.selection.ServiceContentHandler

This interface extends javax.tv.service.selection.ServiceContentHandler by adding the XletProxy methods. It is returned by the ServiceContext.getServiceContentHandlers() method for Xlets which are part of the selected service.

See also: javax.tv.service.selection.ServiceContentHandler, javax.tv.service.selection.ServiceContext.

4.17.5.1 Methods

The following methods are inherited from XletProxy: getLocator, resume, stop, suspend.

The following methods are inherited from org.atsc.management.ObjectStates: addStateChangeListener, getCurrentState, getCurrentStatus, getStatesSupported, removeStateChangeListener.

The following methods are inherited from org.atsc.management.AlarmStatus: clearAlarm, getAlarmStatus.

The following methods are inherited from org.atsc.management.ProceduralStatus: getProceduralStatus.

The following methods are inherited from org.atsc.management.AvailabilityStatus: getAvailabilityStatus.

The following methods are inherited from org.atsc.management.UsageState: getUsageState.

The following methods are inherited from org.atsc.management.OperationalState: getOperationalState.
The following methods are inherited from `org.atsc.management.AdministrativeState`:
getAdministrativeState, setLock.

The following methods are inherited from `javax.tv.service.selection.ServiceContentHandler`:
getServiceContentLocators.

### 4.17.5.2 Fields

The following fields are inherited from `org.atsc.management.ObjectStates`: NOSTATUS.

The following fields are inherited from `org.atsc.management.AlarmStatus`:
ALARM_OUTSTANDING, ALARM_STATUS_ID, CRITICAL, MAJOR, MINOR, UNDER_REPAIR.

The following fields are inherited from `org.atsc.management.ProceduralStatus`:
INIT_REQUIRED, INITIALIZING, NOT_INITIALIZED, PROCEDURAL_STATUS_ID, REPORTING, TERMINATING.

The following fields are inherited from `org.atsc.management.AvailabilityStatus`:
AVAILABILITY_STATUS_ID, DEGRADED, DEPENDENCY, FAILED, INTEST, LOG_FULL, NOT_INSTALLED, OFFDUTY, OFFLINE, POWEROFF.

The following fields are inherited from `org.atsc.management.UsageState`:
ACTIVE, BUSY, IDLE, USAGE_STATE_ID.

The following fields are inherited from `org.atsc.management.OperationalState`:
DISABLED, ENABLED, OPERATIONAL_STATE_ID.

The following fields are inherited from `org.atsc.management.AdministrativeState`:
ADMIN_STATE_ID, LOCKED, SHUTTING_DOWN, UNLOCKED.

### 4.17.6 XletContextExt

```java
public interface XletContextExt
    extends javax.tv.xlet.XletContext
```

This interface represents extensions to the `javax.tv.XletContext` interface. In a DASE System, an object that implements this interface will be specified in the `initXlet()` call.

#### 4.17.6.1 Methods

The following methods are inherited from `javax.tv.xlet.XletContext`:
getProperty, notifyDestroyed, notifyPaused, resumeRequest.

##### 4.17.6.1.1 getDataService()

```java
public javax.tv.service.SIElement getDataService()
```

This method allows an Xlet to discover details about the data service through which it was delivered.

**Returns:**

An object representing a data service. The precise nature of this retrieved object is determined by the application delivery system through which the application that contains this Xlet is delivered.

##### 4.17.6.1.2 getXletInformation()

```java
public XletInformation getXletInformation()
```
This method allows an Xlet to discover details about itself, such as its locator as well as information about the containing application.

Returns:

An object implementing the XletInformation interface and representing information about the Xlet associated with this Xlet context.

4.17.6.1.3 stateChanged(short, int)

public void stateChanged(short stateId, int newState)
    throws java.lang.IllegalArgumentException

This method allows the Xlet to report changes in the [X.731] state attribute indicated by stateId to a new value indicated by newState.

Parameters:

stateId – denotes the state that changed. The values of stateId may be one of OperationalState.OPERATIONAL_STATE_ID or UsageState.- USAGE_STATE_ID.

newState – represents the new value of the state variable indicated by stateId.

Throws:

java.lang.IllegalArgumentException – if any other value of stateId is specified.

4.17.6.1.4 statusChanged(short, int)

public void statusChanged(short statusId, int newStatus)
    throws java.lang.IllegalArgumentException

This method allows the Xlet to report any changes in the [X.731] status attribute indicated by the statusID to a new value indicated by newStatus.

Parameters:

statusId – denotes the status that changed. The values of statusId may be one of AlarmStatus.ALARM_STATUS_ID, ProceduralStatus.PROCEDURAL_STATUS_ID, or AvailabilityStatus.AVAILABILITY_STATUS_ID.

newStatus – represents the new value of the status variable indicated by statusId.

Throws:

java.lang.IllegalArgumentException – if any other value of statusId is specified.

4.17.6.2 Fields

The following fields are inherited from javax.tv.xlet.XletContext:ARGS.

4.17.7 XletInformation

public interface XletInformation

This interface provides additional information about an Xlet.
4.17.7.1 Methods

4.17.7.1.1 getLocator()

public javax.tv.locator.Locator getLocator()

Returns the locator which refers to the Java class file from which this Xlet was loaded.

4.17.7.1.2 getApplicationInformation()

public org.atsc.application.ApplicationInformation getApplicationInformation()

Called to information about the application in which this Xlet is operating.

Returns:

An object that implements ApplicationInformation, where the object represents information regarding the application with which this Xlet is associated.

See also: ApplicationInformation.

4.17.7.2 Fields

No fields are defined.

4.17.8 XletNotRegisteredException

public class XletNotRegisteredException
extends XletAvailabilityException

This exception is thrown when an attempt is made to access an Xlet which has not been registered.

4.17.8.1 Constructors

4.17.8.1.1 XletNotRegisteredException()

public XletNotRegisteredException()

Constructor with no detail message.

4.17.8.1.2 XletNotRegisteredException(java.lang.String)

public XletNotRegisteredException(java.lang.String reason)

Constructor taking a detail message.

Parameters:

reason – the reason this exception was thrown

4.17.8.2 Methods

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

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

No fields are defined.

4.17.9 XletPermission

```java
public final class XletPermission
    extends org.atsc.security.AtscPermission
```

The `XletPermission` class is used for permissions related to all security-protected methods in this Xlet package.

The target name is either a URI referencing the Java class file from which the Xlet was loaded or a special wild card token "*". The actions parameter contains a comma-separated list of the actions requested for the specified target. The actions are "start", "stop", "pause", "resume", "register", "deregister", and "get".

**Note:** See `XletProxy` for information on the use of actions "stop", "pause", and "resume". See `XletRegistry` for information on the use of actions "start", "register", "deregister", and "get".

4.17.9.1 Constructors

4.17.9.1.1 `XletPermission(java.lang.String, java.lang.String)`

```java
public XletPermission(java.lang.String name, java.lang.String action)
```

Creates a new `XletPermission` object with the specified actions.

Parameters:

- `name` – the name is either a URI referencing the Java class file from which the Xlet was loaded or a special wild card token "*".
- `action` – comma-separated list of the actions requested for the specified target. The actions are "start", "stop", "pause", "resume", "register", "deregister", "get".

4.17.9.2 Methods

The following methods are inherited from `org.atsc.security.AtscPermission`: `equals`, `getActions`, `hashCode`.

The following methods are inherited from `java.security.Permission`: `checkGuard`, `getName`, `newPermissionCollection`, `toString`.

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

4.17.9.2.1 `implies(java.security.Permission)`

```java
public boolean implies(java.security.Permission permission)
```

Checks if this permission object implies the specified permission.

Parameters:

- `permission` – the permission to check.

Returns:

The value `true` if this object implies the specified permission.
Overrides:

\[
\text{org.atsc.security.AtscPermission.implies(java.security.Permission)}
\]

4.17.9.3 Fields

No fields are defined.

4.17.10 XletProxy

\[
\text{public interface XletProxy}
\]

\[
\quad \text{extends org.atsc.management.ObjectStates}
\]

This interface represents a proxy for an Xlet. It provides a basic Xlet lifecycle support and Xlet management. Additional descriptive information about the Xlet in the form of an XletInformation class is also available. This class implements the ObjectStates interface in order to add management capability to an Xlet. This interface provides a uniform mechanism to manage any object in a standard way. An Xlet may support a subset of these states as appropriate to the specific Xlet.

The managed XletProxy interface implementation includes a reference to the Xlet interface. The XletProxy acts as a proxy object to the Xlet. All other APIs handle XletProxy references and not Xlet references. This protects the Xlet from rogue Xlets. It also removes any unnecessary state management burden (such as maintaining listeners, etc.) from the Xlet while still enabling Xlet manageability. The Xlet can communicate its state changes with the XletContext interface extended by XletContext. The Xlet manager is expected to maintain a single consistent set of [X.731] state/status attributes per Xlet; this information may be reported through zero or more XletProxy instances.

The XletProxy resume() method is implemented by the Xlet startXlet() method, the suspend() method is implemented by the pauseXlet() method, and the stop() method is implemented by the destroyXlet() method. The Xlet signals the state/status change through invoking the changeState() method on the XletContext object. Since the XletProxy represents a running Xlet, it does not have a start() method. If an Xlet needs to be started, it must be done via the XletRegistry startXlet() method.

Note that only those Xlets that have a corresponding permission can control the state of other Xlets.

See also: XletPermission.

4.17.10.1 Methods

The following methods are inherited from org.atsc.management.ObjectStates:
addStateChangeListener, getCurrentState, getCurrentStatus, getStatesSupported, removeStateChangeListener.

The following methods are inherited from org.atsc.management.AlarmStatus:
clearAlarm, getAlarmStatus.

The following methods are inherited from org.atsc.management.ProceduralStatus:
getProceduralStatus.

The following methods are inherited from org.atsc.management.AvailabilityStatus:
getAvailabilityStatus.

The following methods are inherited from org.atsc.management.UsageState:
getUsageState.
The following methods are inherited from `org.atsc.management.OperationalState`:
getOperationalState.

The following methods are inherited from `org.atsc.management.AdministrativeState`:
getAdministrativeState, setLock.

4.17.10.1.1  `getLocator()`

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

Called to determine Xlet identification represented as a locator.

Returns:
Locator representing this Xlet.

4.17.10.1.2  `resume()`

```java
public void resume()
```

Throws:
- `org.atsc.management.StateChangeException` – when this method would cause an illegal state change.
- `org.atsc.security.AccessDeniedException` – when the caller does not have permission for this operation, where permission is controlled by the “resume” action of XletPermission.
- `XletAvailabilityException` – when the referenced Xlet is no longer executing.

See also: XletPermission.

4.17.10.1.3  `stop()`

```java
public void stop()
```

Throws:
- `org.atsc.management.StateChangeException` – when this method would cause an illegal state change.
- `org.atsc.security.AccessDeniedException` – when the caller does not have permission for this operation, where permission is controlled by the “stop” action of XletPermission.
- `XletAvailabilityException` – when the referenced Xlet is no longer executing.

See also: XletPermission.
4.17.10.1.4 suspend()

public void suspend()
    throws org.atsc.management.StateChangeException,
          org.atsc.security.AccessDeniedException,
          XletAvailabilityException

Called to temporarily pause an execution of this Xlet. This method maps to the
pauseXlet() method.

Throws:

org.atsc.management.StateChangeException – when this method would cause an illegal
state change.

org.atsc.security.AccessDeniedException – when the caller does not have permission
for this operation, where permission is controlled by the “suspend” action of
XletPermission.

XletAvailabilityException – when the referenced Xlet is no longer executing.

See also: XletPermission.

4.17.10.2 Fields

The following fields are inherited from org.atsc.management.ObjectStates: NOSTATUS.

The following fields are inherited from org.atsc.management.AlarmStatus: ALARM_OUTSTANDING, ALARM_STATUS_ID, CRITICAL, MAJOR, MINOR, UNDER_REPAIR.

The following fields are inherited from org.atsc.management.ProceduralStatus: INIT_REQUIRED, INITIALIZING, NOT_INITIALIZED, PROCEDURAL_STATUS_ID, REPORTING, TERMINATING.

The following fields are inherited from org.atsc.management.AvailabilityStatus: AVAILABILITY_STATUS_ID, DEGRADED, DEPENDENCY, FAILED, INTEST, LOG_FULL, NOT_INSTALLED, OFFDUTY, OFFLINE, POWEROFF.

The following fields are inherited from org.atsc.management.UsageState: ACTIVE, BUSY, IDLE, USAGE_STATE_ID.

The following fields are inherited from org.atsc.management.OperationalState: DISABLED, ENABLED, OPERATIONAL_STATE_ID.

The following fields are inherited from org.atsc.management.AdministrativeState: ADMIN_STATE_ID, LOCKED, SHUTTING_DOWN, UNLOCKED.

4.17.11 XletRegistry

public interface XletRegistry
    extends org.atsc.registry.Registry

This interface provides a limited access to the Xlet registry. It allows other Xlets to get
information about existing Xlets, to show an interest in a particular Xlet, and to get access to Xlets
via a proxy.

Prior to initializing an Xlet with javax.tv.xlet.Xlet.initXlet(XletContext), the Xlet
instance shall be automatically registered in the Xlet registry if it is not already registered.
4.17.11.1 Methods

The following methods are inherited from org.atsc.registry.Registry: addRegistryChangeListener, getRegistryType, removeRegistryChangeListener.

4.17.11.1.1 deregisterXlet(javax.tv.locator.Locator)

```java
public void deregisterXlet(javax.tv.locator.Locator locator)
throws XletAvailabilityException,
org.atsc.security.AccessDeniedException
```

Called to remove this Xlet from the registry.

Parameters:

`locator` – identifying the Xlet.

Throws:

- `XletAvailabilityException` – when the locator does not represent a valid already registered Xlet.
- `org.atsc.security.AccessDeniedException` – when the caller does not have permission for this operation, where permission is controlled by the "deregister" action of `XletPermission`.

See also: `XletPermission`.

4.17.11.1.2 getXletInformation()

```java
public XletInformation[] getXletInformation()
```

Called to obtain descriptions of all Xlets registered by the current DASE Application.

Returns:

A array of `XletInformation` objects representing all currently registered Xlets.

4.17.11.1.3 getXletInformation(javax.tv.locator.Locator)

```java
public XletInformation
getXletInformation(javax.tv.locator.Locator locator)
throws XletAvailabilityException
```

Called to obtain a description of the specified Xlet. The Xlet is identified by a locator.

Parameters:

`locator` – identifying the Xlet.

Returns:

- `XletInformation` associated with the specified Xlet.

Throws:

- `XletAvailabilityException` – when the locator does not represent a known Xlet.

4.17.11.1.4 getXletProxies()

```java
public XletProxy[] getXletProxies()
```

This method allows a retrieval of all Xlet proxies representing currently active Xlets. This method is protected via the security mechanisms in order to protect unauthorized access to Xlets.
Returns:

An array of Xlet proxies.

Throws:

org.atsc.security.AccessDeniedException – when the caller does not have the required permission, where permission is controlled by the "get" action of XletPermission.

See also: XletPermission.

4.17.11.1.5  getXletProxy(javax.tv.locator.Locator)

public XletProxy getXletProxy(javax.tv.locator.Locator locator)
throws org.atsc.security.AccessDeniedException,
XletAvailabilityException

Called to get access to a specified Xlet via a proxy. The Xlet must be currently active in order to return a proxy to it. This method is protected via the security mechanisms in order to protect unauthorized access to Xlets.

Parameters:

locator – A locator identifying the Xlet.

Returns:

An Xlet proxy representing the specified Xlet.

Throws:

org.atsc.security.AccessDeniedException – when the caller does not have the required permission, where permission is controlled by the "get" action of XletPermission.

XletAvailabilityException – when the locator does not represent an active Xlet.

See also: XletPermission.

4.17.11.1.6  registerXlet(javax.tv.locator.Locator)

public void registerXlet(javax.tv.locator.Locator locator)
throws XletAvailabilityException,
org.atsc.security.AccessDeniedException

Called to add the specified Xlet to the registry. The Xlet is specified by the locator which identifies the Java class file from which the Xlet may be loaded. The registry is responsible for locating the Xlet and notifying the caller of its availability. This is a non-blocking method; it will return immediately after checking the request. The XletRegistryEvent will be sent to all XletRegistry listeners with an indication of the result of registering this Xlet.

Parameters:

locator – locator identifying the Xlet of interest.

Throws:

XletAvailabilityException – when the Locator does not represent a valid Xlet or if the Xlet has already been registered.

org.atsc.security.AccessDeniedException – when the caller does not have the required permission, where permission is controlled by the "register" action of XletPermission.
See also: XletPermission, RegistryChangeListener.

4.17.11.1.7 \texttt{startXlet(javax.tv.locator.Locator, java.lang.String[])}

\begin{verbatim}
public XletProxy startXlet(javax.tv.locator.Locator locator, java.lang.String[] args)
throws org.atsc.security.AccessDeniedException,
XletAvailabilityException,
org.atsc.management.StateChangeException
\end{verbatim}

Called to activate a specific Xlet. The method call returns as soon as the requested Xlet starts executing. This method is protected via the security mechanisms in order to protect unauthorized access to Xlets.

Parameters:

\begin{itemize}
  \item \texttt{locator} – locator identifying the Xlet.
  \item \texttt{args} – An array of arguments to be provided to the Xlet.
\end{itemize}

Returns:

An \texttt{XletProxy} representing the started Xlet which can be used to manage it.

Throws:

\begin{itemize}
  \item \texttt{org.atsc.security.AccessDeniedException} – when the caller does not have the required permission, where permission is controlled by the "start" action of \texttt{XletPermission}.
  \item \texttt{XletAvailabilityException} – when the Locator does not represent a valid registered Xlet.
  \item \texttt{org.atsc.management.StateChangeException} – for case related to the state of the Xlet.
\end{itemize}

See also: \texttt{XletPermission}.

4.17.11.2 \texttt{XletRegistryEvent}

\begin{verbatim}
public class XletRegistryEvent
extends org.atsc.registry.RegistryChangeEvent
\end{verbatim}

This event extends the \texttt{RegistryChangeEvent} with Xlet specific methods. It is delivered to \texttt{XletRegistry} listeners when changes in the \texttt{XletRegistry} occur.

4.17.12 \texttt{XletRegistryEvent} \texttt{(java.lang.Object, ...)}

\begin{verbatim}
public XletRegistryEvent(
java.lang.Object source,
XletChangeCause cause,
XletInformation xletInfo)
\end{verbatim}

Parameters:

\begin{itemize}
  \item \texttt{source} – The object that caused this event.
  \item \texttt{cause} – The cause of the Xlet registry event.
\end{itemize}
xletInfo — The XletInformation object representing the Xlet changed in the Xlet Registry.

See also: XletInformation.

4.17.12.2 Methods

The following methods are inherited from org.atsc.registry.RegistryChangeEvent: getCause, getRegistryType.

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), and wait(long,int).

4.17.12.2.1 getXletInformation()

public XletInformation getXletInformation()

Called to determine which Xlet caused the event.

Returns:

Object representing the XletInformation of the Xlet related to the change.

4.17.12.3 Fields

The following fields are inherited from java.util.EventObject: source.
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 2 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>4</td>
<td>Change references to org.w3c.dom.html to org.w3c.dom.html2.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.events package.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.events.KeyEvent.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.events.KeyModifiers.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.events.VirtualKeys.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.html.HTMLImageElementExt.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.dom.views.DocumentViewExt.refresh(String).</td>
</tr>
<tr>
<td>4</td>
<td>Correct references to java.net.DatagramSocket in method signatures.</td>
</tr>
<tr>
<td>4</td>
<td>Remove javatv.* system properties.</td>
</tr>
<tr>
<td>4</td>
<td>Add org.atsc.xlet.XletContextExt.getXletInformation().</td>
</tr>
<tr>
<td>4</td>
<td>Clarify semantics for automatic Xlet registration.</td>
</tr>
<tr>
<td>4</td>
<td>Remove extraneous [] from return value of XletRegister.getXletInformation(javax.tv.locator.Locator)</td>
</tr>
</tbody>
</table>