BRAZILIAN
STANDARD

ABNT NBR 15606-4

First edition
2010.04.13

Valid from
2010.05.13

Digital terrestrial television — Data coding and transmission specification for digital broadcasting
Part 4: Ginga-J — The environment for the execution of procedural applications

ICS 33.080; 33.160.01

ISBN 978-85-07-02022-6

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Reference number
ABNT NBR 15606-4:2010
95 pages

© ABNT 2010

ABNT office
All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ABNT.

ABNT office
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 3974-2346
abnt@abnt.org.br
www.abnt.org.br

Published in Brazil

© ABNT 2010 - All rights reserved

Contents

Pages

Foreword viii
Introduction ix
1 Scope 1
2 Normative References 1
3 Terms and definitions 2
4 Abbreviations 3
5 Architecture of the Ginga middleware 3
5.1 Overview of the Ginga architecture 3
5.2 Ginga-J architecture 4
5.2.1 Context 4
5.2.2 Architecture 5
6 Content format 5
7 Ginga-J application model 5
7.1 Application model 5
7.1.1 Life cycle 5
7.1.2 Startup of applications 7
7.1.3 Finalization of applications 7
7.1.4 Support for multiple applications 8
7.1.5 The sharing of resources between applications 8
7.1.6 Controlling applications 8
7.1.7 Communication between applications 8
7.1.8 Properties of the environment 9
7.1.9 Application control codes 9
7.2 Storage and caching of applications 11
7.2.1 Storage models 11
7.2.2 Storage issues 11
7.2.3 Proactive caching 12
7.3 Transmission of applications 12
7.3.1 Signaling rules 12
7.3.2 Packaging of applications 12
7.3.3 Application authentication 12
7.3.4 Signaling the same application in different departments 12
7.3.5 Download of applications through the interactive channel 12
8 Ginga-J Platform 13
8.1 Java Platform 13
8.2 Basic considerations of the platform 13
8.2.1 Execution environment 13
8.2.2 Hierarchy of packages and classes 13
8.2.3 Notification of events 14
8.2.4 Text coding 14
8.2.5 Life cycle of the applications 14

© ABNT 2010 - All rights reserved

8.3 Common infrastructure 14
8.4 Graphical presentation and events handling 15
8.4.1 LWUIT, LightWeight user interface toolkit 15
8.4.2 Graphical user interface 16
8.4.3 Treatment of the platform planes 16
8.4.4 User events handling 22
8.5 Information and selection of services 23
8.5.1 General considerations 23
8.5.2 Integration with API protocol-independent 23
8.6 Presentation and execution of media 24
8.7 Access to data 24
8.7.1 General considerations 24
8.7.2 Access to files 24
8.7.3 Broadcast transport protocol 24
8.7.4 Persistent storage 24
8.7.5 Access to system properties 25
8.7.6 IP support over an interactive channel 25
8.7.7 MPEG-2 section filtering 25
8.8 Application management 25
8.9 Tuning 26
8.10 NCL Bridge 26
8.11 Platform properties 26
8.11.1 System Properties 26
8.11.2 User properties 27
8.12 Interactivity channel 27
8.13 Complete list of Ginga-J packages 27
8.13.1 Java platform packages 27
8.13.2 JavaTV 1.1 and JMF 1.0 specification packages 28
8.13.3 JavaDTV 1.1 specification packages 29
8.13.4 JSSE 1.0.1 specification packages 30
8.13.5 JCE 1.0.1 specification packages 30
8.13.6 SATSA 1.0.1 specification packages 31
8.13.7 Ginga-J specific packages 31
Annex A (normative) **Java DTV 1.3 specification** 32
A.1 General considerations 32
A.2 Java DTV API 32
A.2.1 com.sun.dtv.broadcast package 32
A.2.2 com.sun.dtv.smartcard package 33
A.2.3 com.sun.dtv.lwuit.events package 33
A.2.4 com.sun.dtv.filtering package 34
A.2.5 com.sun.dtv.ui.event package 35
A.2.6 com.sun.dtv.lwuit.plaf package 36
A.2.7 com.sun.dtv.media.timeline package 37

© ABNT 2010 - All rights reserved

A.2.8 com.sun.dtv.media.language package 37
A.2.9 com.sun.dtv.application package 37
A.2.10 com.sun.dtv.media.audio package 38
A.2.11 com.sun.dtv.test package 38
A.2.12 com.sun.dtv.tuner package 39
A.2.13 com.sun.dtv.lwuit.layouts package 39
A.2.14 com.sun.dtv.broadcast.event package 40
A.2.15 com.sun.dtv.lwuit.list package 40
A.2.16 com.sun.dtv.ui package 41
A.2.17 com.sun.dtv.media.control package 43
A.2.18 com.sun.dtv.media.dripfeed package 43
A.2.19 com.sun.dtv.security package 43
A.2.20 com.sun.dtv.lwuit.painter package 44
A.2.21 com.sun.dtv.locator package 44
A.2.22 com.sun.dtv.resources package 45
A.2.23 com.sun.dtv.net package 45
A.2.24 com.sun.dtv.media.text package 45
A.2.25 com.sun.dtv.media.format package 46
A.2.26 com.sun.dtv.platform package 47
A.2.27 com.sun.dtv.io package 47
A.2.28 com.sun.dtv.lwuit.animations package 47
A.2.29 com.sun.dtv.service package 48
A.2.30 com.sun.dtv.media package 48
A.2.31 com.sun.dtv.transport package 49
A.2.32 com.sun.dtv.lwuit.util package 49
A.2.33 com.sun.dtv.lwuit package 50
A.2.34 com.sun.dtv.lwuit.geom package 51
Annex B (normative) **Specification of the protocol-dependent service information API** 52
B.1 General considerations 52
B.2 Protocol-dependent service information API 52
B.2.1 br.org.sbtvd.net package 52
B.2.1.1 SBTVDLocator class 52
B.2.1.2 SBTVDNetworkBoundLocator class 54
B.2.2 br.org.sbtvd.si package 54
B.2.2.1 DescriptorTag interface 54
B.2.2.2 PMTElementaryStream interface 59
B.2.2.3 PMTService interface 60
B.2.2.4 PMTStreamType interface 61
B.2.2.5 SIBouquet interface 62
B.2.2.6 SI Broadcaster interface 62
B.2.2.7 SIEvent interface 63
B.2.2.8 SIInformation interface 65
B.2.2.9 SIIterator interface 66

© ABNT 2010 - All rights reserved

B.2.2.10 SIMonitoringListener interface 66
B.2.2.11 SIMonitoringType interface 66
B.2.2.12 SINetwork interface 67
B.2.2.13 SIRetrievalListener interface 67
B.2.2.14 SIRunningStatus interface 68
B.2.2.15 SIService interface 68
B.2.2.16 SIServiceType interface 70
B.2.2.17 SITime interface 71
B.2.2.18 SITransportStream interface 71
B.2.2.19 SITransportStreamBAT interface 72
B.2.2.20 SITransportStreamNIT interface 72
B.2.2.21 Descriptor class 72
B.2.2.22 SIDatabase class 73
B.2.2.23 SIExEventInformation class 76
B.2.2.24 SILackOfResourcesEvent class 76
B.2.2.25 SIMonitoringEvent class 77
B.2.2.26 SINotInCacheEvent class 78
B.2.2.27 SIObjectNotInTableEvent class 78
B.2.2.28 SIRequest class 78
B.2.2.29 SIRequestCancelledEvent class 78
B.2.2.30 SIRetrievalEvent class 78
B.2.2.31 SISuccessfulRetrieveEvent class 79
B.2.2.32 SITableNotFoundEvent class 79
B.2.2.33 SITableUpdatedEvent class 79
B.2.2.34 SIUtil class 80
B.2.2.35 SIException() class 80
B.2.2.36 SIIllegalArgumentException() class 80
B.2.2.37 SIInvalidPeriodException class 80
Annex C (normative) **API extension specification for tuning – Packagebr .org.sbtvd.net.tuning** 81
C.1 ChannelManager class 81
C.2 Channel class 82
Annex D (normative) **NCL Bridge API Specification** 83
D.1 General considerations 83
D.2 NCL bridge API 83
D.2.1 br.org.sbtvd.net.tuning package 83
D.2.1.1 NCLPlayer class 83
D.2.1.2 NCLPlayerEvent class 85
D.2.1.3 NCLPlayerEventListener interface 86
D.2.1.4 NCLGingaSettingsNodes class 86
D.2.1.5 NCLEdit class 87
D.2.2 br.org.sbtvd.net.bridge.ncl package 91
D.2.2.1 NodeManager class 91

© ABNT 2010 - All rights reserved

D.2.2.2 NCLEvent class 91
D.2.2.3 NCLEventListener interface 92
Annex E (normative) API specification for graphic plane support – br.org.sbtvd.ui package 93
E.1 ColorCoding class 93
E.2 StillPicture class 93
E.3 SwitchArea class 94
Bibliography 95

Figures

Figure 1 – High-level architecture of Ginga middleware 4
Figure 2 – Ginga-J context 4
Figure 3 – Ginga-J architecture and the execution environment 5
Figure 4 – Diagram with the life cycle states of an Xlet 6
Figure 5 – Structure of layers for the presentation of services 17
Figure 6 – Composition example of content display plane 22

Tables

Table 1 – Properties of the Ginga-J environment 9
Table 2 – Ginga-J application control codes 9
Table 3 – Details of functionalities for the text and grap 17
Table 4 – Details of the functionalities for video and still picture plane 18
Table 5 – Details of functionalities for the still picture plane 20
Table 6 – Details of the functionalities for the video plane 21
Table 7 – Mapping of functions of ABNT NBR 15604:2007 in the events defined by JAVADTV 1.3:2009 23
Table 8 – System Properties 26
Table A.1 – Classes of the com.sun.dtv.broadcast package 32
Table A.2 – Classes of the com.sun.dtv.smartcard package 33
Table A.3 – Classes of the com.sun.dtv.lwuit.events package 33
Table A.4 – Classes of the com.sun.dtv.filtering package 34
Table A.5 – Classes of the com.sun.ui.event package 35
Table A.6 – Classes of the com.sun.lwui.plaf package 36
Table A.7 – Classes of the com.sun.dtv.media.timeline package 37
Table A.8 – Classes of the com.sun.dtv.media.language package 37
Table A.9 – Classes of the com.sun.dtv.application package 38
Table A.10 – Classes of the com.sun.dtv.media.auto package 38
Table A.11 – Classes of the com.sun.dtv.test package 39
Table A.12 – Classes of the com.sun.dtv.tuner package 39
Table A.13 – Classes of the com.sun.dtv.lwuit.layouts package 40
Table A.14 – Classes of the com.sun.dtv.broadcast.event package 40
Table A.15 – Classes of the com.sun.dtv.lwuit.list package 41

© ABNT 2010 - All rights reserved

Table A.16 – Classes of the com.sun.dtv.ui package 41
Table A.17 – Classes of the com.sun.dtv.media.control package 43
Table A.18 – Classes of the com.sun.dtv.media.dripfeed package 43
Table A.19 – Classes of the com.sun.dtv.security package 43
Table A.20 – Classes of the com.sun.dtv.lwuit.painter package 44
Table A.21 – Classes of the com.sun.dtv.locator package 44
Table A.22 – Classes of the com.sun.dtv.resources package 45
Table A.23 – Classes of the com.sun.dtv.net package 45
Table A.24 – Classes of the com.sun.dtv.media.text package 46
Table A.25 – Classes of the com.sun.dtv.media.format package 46
Table A.26 – Classes of the com.sun.dtv.platform package 47
Table A.27 – Classes of the com.sun.dtv.io package 47
Table A.28 – Classes of the com.sun.dtv.lwuit.animations package 48
Table A.29 – Classes of the com.sun.dtv.service package 48
Table A.30 – Classes of the com.sun.dtv.media package 48
Table A.31 – Classes of the com.sun.dtv.transport package 49
Table A.32 – Classes of the com.sun.dtv.lwuit.util package 49
Table A.33 – Classes of the com.sun.dtv.lwuit package 50
Table A.34 – Classes of the com.sun.dtv.lwuit.geom package 51

© ABNT 2010 - All rights reserved

Foreword

The Associação Brasileira de Normas Técnicas (ABNT) is the Brazilian Standardization Forum. Brazilian Standards, wich content is the responsability of the Brazilian Committees (ABNT/CB), Sectorial Standardization Bodies (ABNT/ONS), and Special Studies Committees (ABNT/CEE), are drafted by Study Committees (CE). Such Studiys Committees are made up of representatives from the sectors involved and include producers, consumers and neutral entities (universities, laboratories and others).

Brazilian Standards are drafted in accordance with the rules given in the ABNT Directives (Diretivas), Part 2.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ABNT shall not be held responsible for identifying any or all such patent rights.

ABNT NBR 15606-4 was prepared within the purview of the Special Studies Committes of Digital Television (ABNT/CEE-85). The 1st Draft Standard was circulated for National Consultation in accordance with ABNT Notice (Edital) n° 09, from September 6th, 2007 to November 5th, 2007, with the number Draft 00:001.85-006/4. The 2nd Draft Standard was circulated in accordance with ABNT Notice (Edital) nº 05, from May, 19th, 2009 to July 17th, 2009, with number Draft 00:001.85-006/4. The 3rd Draft Standard circulated in accordance with ABNT Notice (Edital) nº 3 from March, 5th, 2010 to April 5th,2010, with the number 3rd Draft 85:000.00-006/4.

Should any doubts arise regarding the interpretation of the English version, the provisions in the original text in Portuguese shall prevail at all time.

This Standard is based on work carried out by the Brazilian Digital Television Forum as established by Presidential Decree nº 5820 of June 29, 2006.

This English version is equivalent to ABNT NBR 15606-4:2010, from 2010.04.13.

This version in English was published in 2010.06.15.

© ABNT 2010 - All rights reserved

Introduction

The Ginga-J definition comprises a set of Application Programming Interfaces (API) designed to provide all the functionalities necessary for the implementation of applications for digital television, from the handling of multimedia data to the access protocols.

The Ginga specification applies to receivers for terrestrial television transmission systems (over-the-air). Ginga is designed to cover a full range of implementations including integrated receiver-decoders (IRD), integrated television sets, multimedia computers and local clusters of devices connected via Home Area Networks (HAN).

This part of ABNT NBR 15606 is aimed at developers of receivers compatible with the Brazilian digital terrestrial television system (SBTVD) and developers of applications that use the functionality and Ginga API.

This part of ABNT NBR 15606 is aimed at ensuring the interoperability of Ginga applications and different Ginga implementations.

This part of ABNT NBR 15606 is consistent with international specifications, as detailed in Annex A.

© ABNT 2010 - All rights reserved

Digital terrestrial television — Data coding and transmission specification for digital broadcasting
Part 4: Ginga-J — The environment for the execution of procedural applications

1 Scope

This part of ABNT NBR 15606 specifies the requirements for the procedural part of the middleware for the Brazilian digital terrestrial television system (SBTVD).

2 Normative References

The following documents are indispensable for the application of this document. For dated references, only the cited editions apply. For undated references, the latest edition of the referenced document (including any amendments) applies.

ABNT NBR 15601:2007, *Digital terrestrial television –Transmission system*

ABNT NBR 15603:2007 (all parts), *Digital terrestrial television – Multiplexing and service information (SI) Part 1*

ABNT NBR 15604:2007, *Digital terrestrial television – Receivers*

ABNT NBR 15606-1, *Digital terrestrial television – Data coding and transmission specification for digital broadcasting – Part 1: Data coding*

ABNT NBR 15606-2:2007, *Digital terrestrial television – Data coding and transmission specification for* digital broadcasting – Part 2: Ginga-NCL for fixed and mobile receivers – XML application language for application coding

ABNT NBR 15606-3:2007, *Digital terrestrial television – Data coding and transmission specification for digital broadcasting – Part 3: Data transmission specification*

ISO 639-2, *Codes for the representation of names of languages – Part 2: alpha-3 code*

ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet N° 1*

ISO/IEC 13818-1, *Information technology – Generic coding of moving pictures and associated audio information: Systems*

ARIB STD-B10:2008, *Service information for digital broadcasting system*

ARIB STD-B23:2004, *Application execution engine platform for digital broadcasting*

ARIB STD-B31:2007, *Transmission Coding Standard*

CDC 1.1:2008, *Connected Device Configuration 1.1 (JSR 218), available in http://jcp.org/en/jsr/detail?id=218*

FP 1.1:2008, *Foundation Profile 1.1 (JSR 219), available in http://jcp.org/en/jsr/detail?id=219*

© ABNT 2010 - All rights reserved

JAR:2009, *Sun Microsystems. JAR File Specification. 2009, available in http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html*

LWUIT 1.1:2008 , *LightWeight User Interface Toolkit, Sun Microsystems*

JAVADTV 1.3:2009, *Java DTV Specification, Sun Microsystems.*

JAVATV 1.1:2008, *Java TV Specification 1.1 (JSR 927), Sun Microsystems available at: http://jcp.org/en/jsr/detail?id=927*

JCE:1.0.1:2006 , *Sun Microsystem. Security (JCE – Java Cryptography Extension) Optional Package Specification v1.0.1, available at: http://jcp.org/en/jsr/detail?id=219*

JSSE:1.0.1:2006, *Sun Microsystem. Security (JSSE – Java Secure Socket Extension) Optional Package Specification v1.0.1, available at: http://jcp.org/en/jsr/detail?id=219*

JVM:1997, *Java(TM) Virtual Machine Specification, The (2nd Edition), T Lindholm, F Yellin – 1997 – Addison-Wesley*

PBP 1.1:2008, *Personal Basis Profile 1.1 (JSR 217), available atn http://www.jcp.org/*

SATSA:1.0.1:2007, *Sun Microsystems , Security and Trust Services API for J2ME (JSR 177), available at: http://jcp.org/en/jsr/detail?id=177*

3 Terms and definitions

For the purposes of this part of ABNT NBR 15606, the following terms and definitions apply.

3.1
bytecode
intermediate form of the code interpreted by the JVM

3.2
service context
environment in which the service is displayed on the digital receiver

3.3
Java Virtual Machine
JVM
process that loads and runs the Java applications

3.4
service
set of information, which contains audio, video and/or data, to be displayed on a digital receiver

NOTE Viewers normally reference the service as a “television channel”.

3.5
zapper
resident application, normally developed by the manufacturer of the receiver, which user can activate at any time

NOTE The zapper can be used to select services and applications for execution at a later time.

© ABNT 2010 - All rights reserved

4 Abbreviations

For the purposes of this part of ABNT NBR 15606, the following abbreviations apply.

API	Application Programming Interface
CA	Conditional Access
CDC	Connected Device Configuration
CSS	Cascading Style Sheets
ECMA	European Computer Manufacturers Association
EDT	Event
EPG	Electronic Program Guide
HAN	Home Area Network
IRD	Integrated Receiver Decoder
JPEG	Joint Photographic Expert Group
MIDP	Móbile Information Devide Profile
MPEG	Moving Picture Expert Group
NCL	Nested Context Language
PBP	Personal Basis Profile
PNG	Portable Network Graphics
SBTVD	Sistema Brasileiro de Televisão Digital Terrestre
TOT	Time Offset Table
TS	Transport Stream
UTF	Unicode Transformation Format
XHTML	Extensible Hypertext Markup Language

5 Architecture of the Ginga middleware

5.1 Overview of the Ginga architecture

The universe of applications for digital television can be partitioned into two sets: the declarative application set and the procedural application set. A declarative application is one in which its “initial” entity is of the “declarative content” type. Similarly, a procedural application is one in which its “initial” entity is of the “procedural content” type.

A declarative content shall be based on a declarative language, that is, a language that emphasizes the declarative description of the problem, instead of its breakdown into an algorithmic implementation. A procedural content shall be based on a non-declarative language. Non-declarative languages can follow different paradigms. Thus, we have module-based languages, object-oriented languages, etc. The literature on digital television, however, uses the term “procedural” to represent all languages that are not declarative. In procedural programming, the computer shall be informed about each step to be executed. One can assert that in procedural languages, the developer has greater power over the code, and is able to establish the entire flow of control and execution of the program; and because there are more resources available, the degree of complexity is greater. Java is the most common language found in procedural environments of digital television systems.Ginga-NCL (or Presentation Machine) is a logical subsystem of the Ginga system that processes NCL documents. A key component of Ginga-NCL is the informative content decoding mechanism (NCL formatter). Other important modules are the XHTML-based user, which includes a style language (CSS) and ECMAScript interpreter; and the LUA mechanism, which is responsible for interpreting LUA Scripts.

Ginga-J (or Execution Machine) is a logical subsystem of the Ginga System that processes procedural applications (Java Xlets). A key component of the procedural application environment is the procedural content execution mechanism, which is based on a Java Virtual Machine.

© ABNT 2010 - All rights reserved

It is important to observe that in a uniquely Ginga-NCL ou uniquely Ginga-J implementation, either in fixed or mobile receivers, the assertion of any kinds of conformity with SBTVD is prohibited. Thus, Ginga assures the offer of profiles that are always compatible with the previous versions.

Common content decoders shall serve the needs of both procedural and informative decoding applications and presentation of the common PNG, JPEG, MPEG-type content and other formats. Ginga-Core is comprised of common content decoders and procedures to obtain content transported in MPEG-2 transport streams (TS) and through a return channel. Ginga-Core shall also support the conceptual view model described in ABNT NBR 15606-1.

Architecture (see Figure 1) and the features of the Ginga specification shall be designed for application in transmission systems and receivers for terrestrial (over-the-air) transmission. Moreover, the same architecture and features can be applied to other transport systems (such as satellite television systems or cable TV systems).

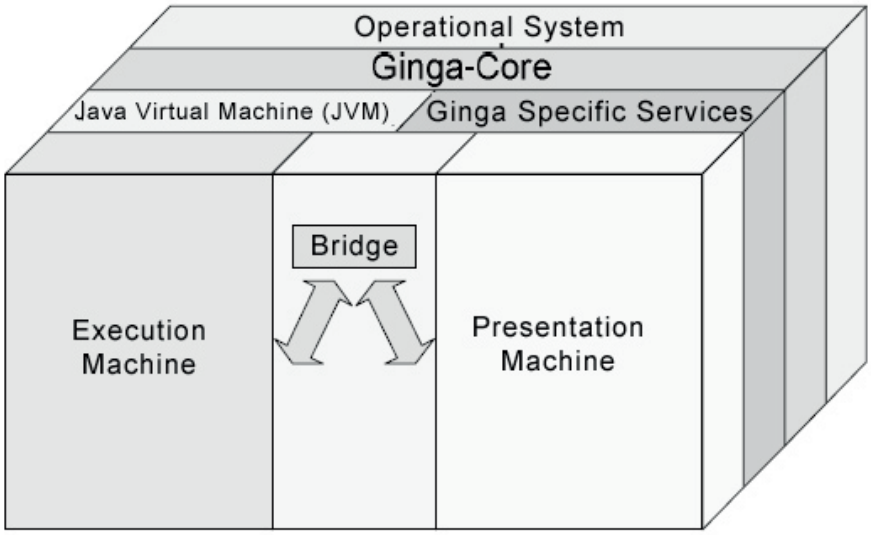

Figure 1 – High-level architecture of Ginga middleware

5.2 Ginga-J architecture

5.2.1 Context

Figure 2 presents the context in which the stack of Ginga-J software is executed. Ginga-J is a specification of distributed middleware, which resides in a Ginga device (a device that deploys the Ginga middleware - a digital television receiver).

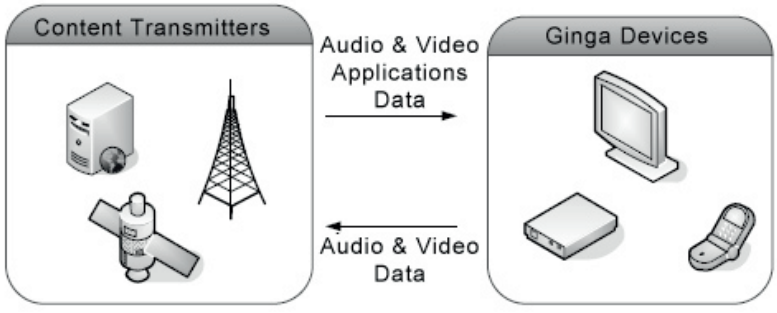

Figure 2 – Ginga-J context

© ABNT 2010 - All rights reserved

The Ginga device shall have access to streams of video, audio, data and other media that shall be transmitted over air, cable, satellite or IP networks. The information received shall be processed and displayed to the viewers.

5.2.2 Architecture

The Ginga-J model distinguishes between hardware and software entities and resources of the system and applications as described in Figure 3.

The resident applications can be implemented using non-standard functions, provided by the Operating System of the Ginga device, or by a particular Ginga implementation. Resident applications may also incorporate functionalities provided by the standardized Ginga-J API. Transmitted applications (Xlets) shall always use standardized API provided by Ginga-J.

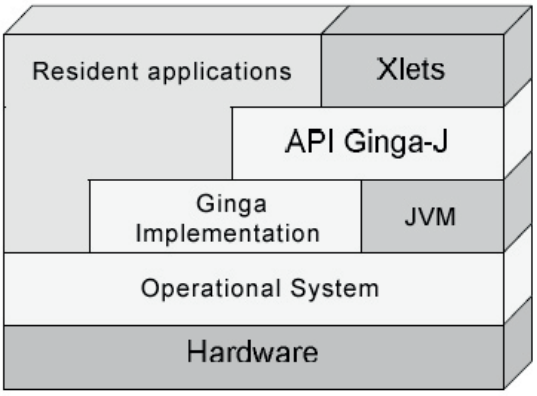

Figure 3 – Ginga-J architecture and the execution environment

In general, Ginga is extraneous to any resident applications. These resident applications include, but not limited to, the following: closed caption, conditional access system messages (CA), resident receiver menus and resident electronic programming guides (EPG).

Resident applications may have priority over Ginga applications. For example, the closed caption and emergency message shall have priority in the Ginga system.

6 Content format

The content format for the Brazilian digital terrestrial television system shall be compliant with ABNT NBR 15606-1.

7 Ginga-J application model

7.1 Application model

7.1.1 Life cycle

The Ginga-J application model shall be compliant with the model defined in JAVADTV 1.3:2009. Hence, Ginga-J applications shall contain a class implementing the javax.microedition.xlet.Xlet interface (see PBP 1.1:2008), which shall be referenced in accordance with the definitions of application signaling (see ABNT NBR 15606-3:2007, 12.16). Otherwise, the class (and the instance of the application) may be ignored.

© ABNT 2010 - All rights reserved

Ginga-J applications shall be executed in an environment geared toward services and maintained by a global application manager of the system. Every service is presented in a service context, which may be defined as its environment of execution. For Ginga-J applications, the service context is represented by an instance of the javax.microedition.xlet.ServiceContext class.

Furthermore, Ginga-J applications can be controlled either by a zapper, by the broadcaster (see ABNT NBR 15606-3:2007, 12.16), by another application using the Ginga-J API “Application Lifecycle Management and Control” (see JAVADTV 1.3:2009) or by Ginga-NCL documents. Considering that the application provides a class that implements the javax.microedition.xlet.Xlet interface, this class contains at least four methods that are called by the platform to inform the application of imminent life cycle changes. A diagram showing the life cycle of Ginga-J applications is shown in Figure 4.

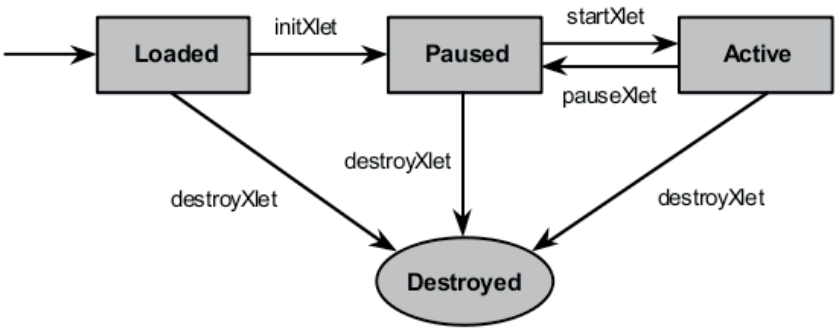

Figure 4 – Diagram with the life cycle states of an Xlet

Initially, after the data of the application is obtained, the object that implements javax.microedition.xlet.Xlet is created by using its constructor. If the default constructor returns without triggering an exception, the application instance will be considered as being in the “Loaded” state; if not, the application instance will be considered as being in the “Destroyed” state and discarded.

NOTE The initialization of the resources utilized by the application is done in the initXlet() method and not the constructor of the class. The call to the constructor is implementation-dependent.

In order to start the initXlet method application, a javax.microedition.xlet.XletContext instance object is called, which has information about the context of the execution for the application including properties and mechanisms for the notification of changes in states initiated by the application. Once the instance of the application has been successfully loaded and successfully initiated, the application manager can change the state of the application instance to “Paused.”

NOTE It’s possible that the initXlet method will be called asynchronously.

The startXlet method can then be called to inform the application that it will be converted to the “Active” state, initiating the execution thereof.

The pauseXlet method may be called to inform the application that it shall change to the “Paused” state and that it shall minimize its consumption of resources. The application can change back to the “Active” state after a new call to the startXlet method. An application instance in the “Paused” state shall reduce its consumption of resources if it intends to maximize its likelihood of survival. This does not mean that it cannot maintain any resources, but that - if it does - it shall have a lower priority for accessing resources than if it were in the “Active” state.

© ABNT 2010 - All rights reserved

The destroyXlet method may be called in any state and is used to notify the application that it is about to have its application completed. The application shall save information on its state (if possible and necessary) and release previously used resources as soon as possible. This method has a Boolean parameter that indicates if it is unconditional that this application shall be shut down. If the application is being shut down due to a service selection operation, the finalization of the application will be unconditional (see 7.1.3).

NOTE An application instance may enter this state only once.

If a Xlet interface method triggers javax.microedition.xlet.XletStateChangeException (see PBP 1.1:2008), by default the Xlet remains in the state it was in immediately before the call that triggered the exception. The only exception to this rule is the destroyXlet method when the unconditional Boolean parameter is passed with a true value, In this scenario, triggering XletStateChangeException shall have no effect and the Xlet shall be destroyed.

The initXlet method shoall be called only once. Moreover, the application manager of may choose to change the Xlet to the “Destroyed” state (without calling destroyXlet) at some specific implementation time afterwards if an XletStateChangeException is triggered.

Applications in the “Destroyed” state cannot be started with the “Application Lifecycle Management and Control” API (see JAVADTV 1.3:2009) or any other mechanism available on the platform.

7.1.2 Startup of applications

When a new service is selected for display, the global application manager of the system shall check the available applications in accordance with ABNT NBR 15606-3:2007, 12.16. In particular, this manager also identifies applications that shoall be started immediately after the selection of the service in question.

If there is already an application running at the time of de-selection of service that contains it, such application can continue running if it is flagged as a valid application for the new service selected.

After the selection of a service, it’s possible that flagged alternative applications are started by the user, by the zapper or by any other applications using the “Application Lifecycle Management and Control” API (see JAVADTV 1.3:2009). In other words, the user can start an application after receiving an offer of applications through a particular user interface. Since this interface is dependent on implementation, flagged services shall indicate explicitly if they require the application to be started automatically (see ABNT NBR 15606-3:2007, 12.16).

7.1.3 Finalization of applications

Ginga-J applications can voluntarily terminate their application using the Xlet API (see PBP 1.1:2008) or can be finalized by the system’s application manager.

Additionally, an application shall have its execution interrupted unconditionally whenever any of the following conditions occur:

— the AIT table (see ABNT NBR 15603:2007) has been updated or substituted and in this new version there is no reference to the application in question;

— the AIT table (see ABNT NBR 15603:2007) is no longer referenced in the PMT table (see ABNT NBR 15603:2007) of the service being displayed.

© ABNT 2010 - All rights reserved

When an application instance is chosen to have its execution terminated, the application manager shall call its destroyXlet method. As described in 7.1.1, if this instance is being shut down due to a service selection operation, its finalization shall be unconditional.

7.1.4 Support for multiple applications

Ginga-J applications shall be executed in a multitasking environment, geared toward media events marked by broadcast and user input events. The application model was designed to be extensible. It is possible to support multiple competing applications that are cooperating (designed to communicate with each other and share resources) or non-cooperating (independent and competing for resources).

The support model for multiple applications in Ginga-J shall be based on the application model defined in JAVADTV 1.3:2009. Thus, an application cannot be started if an instance of this application is already active in the service selected for display. For cases where more than one Xlet is running, no actions that may affect the global state of the platform are permitted (see 7.1.1).

Each application instance is considered as if it were running within its own virtual machine instance. Nevertheless, it is the broadcaster's responsibility to ensure that applications executed simultaneously in a particular service are comprehensible to the user and cause no perceivable problems by mutual interference.

7.1.5 The sharing of resources between applications

Allowing the simultaneous execution of multiple applications, means that some rules shall be defined for these applications to share available resources in the system. In particular, applications being executed shall share the "Input Focus" and "Output Focus".

An application has the 'Input Focus" if, and only if, the java.awt.Component or the com.sun.dtv.lwuit. Component, which have "Input Focus", belongs to the components tree of that application. The "Input Focus" can be requested by applications calling the requestFocus method in one of the aforementioned classes, depending on the type of graphical component used.

The application that has the "Input Focus" is, in principle, capable of receiving user input events. Other applications that do not have the "Input Focus" can request the reception of a subgroup of user input events through the "TV Specific UI functionality Event" API (see JAVADTV 1.3:2009).

7.1.6 Controlling applications

It is possible to control the life cycle of an application through the "Application Lifecycle Management and Control" API (JAVADTV 1.3:2009), which provides means to permit applications to request the Application Manager to start, stop, pause and resume other applications.

7.1.7 Communication between applications

The model of communication between Ginga-J applications shall comply with the application model defined in JAVADTV 1.3:2009. Applications shall use the mechanisms defined in the "Inter-Xlet Communication" API (see PBP 1.1:2008) for such mechanisms. Communication between one application and another is established by the connection between an object to a name in javax. microedition.xlet.ixc.IxcRegistry (see PBP 1.1:2008) and another application seeking this name and invoking the methods of the object. The possible "namespaces" for registration shall comply with the definitions established by the communication interface between applications of JAVADTV 1.3:2009.

© ABNT 2010 - All rights reserved

7.1.8 Properties of the environment

In addition to the properties already defined by the Ginga-NCL model (see ABNT NBR 15606-2:2007, 7.2.4), the Ginga-J application model shall provide each application with a javax.microedition.xlet. XletContext (see PBP 1.1:2008) including a set of specific properties already defined in JAVADTV 1.3:2009, which are shown in Table 1.

Table 1 – Properties of the Ginga-J environment

Name	Type of data	Description
com.sun.dtv.persistent.root	Alphanumeric	Base directory for persistent storage on the platform
com.sun.dtv.orgid	Numeric	Unique identifier for the organization responsible for the application. It shall be the same value transmitted in the organization_id field of the application descriptor identifier (see ABNT NBR 15606-3:2007, 12.7)
com.sun.dtv.appid	Numeric	Unique identifier for the application. It shall be the same value transmitted in the application_id field of the application descriptor identifier (see ABNT NBR 15606-3:2007, 12.7)
com.sun.dtv.version	Alphanumeric	Version number of the JavaDTV specification implemented by the platform (JAVADTV 1.3:2009)
br.org.ginga.system. version	Alphanumeric	Version number of the Ginga-J specification implemented by the platform

7.1.9 Application control codes

The dynamic control of the application's life cycle is signaled by the "application_control_code" field for the application in the AIT (see ABNT NBR 15606-3:2007). The Ginga-J application control codes are shown in Table 2.

Table 2 – Ginga-J application control codes

Code	Identifier	Description
0x00	Not defined	Reserved for future use
0x01	AUTOSTART	Applications with the AUTOSTART control code start automatically upon selection of the service that contains such applications
0x02	PRESENT	Applications with the PRESENT control code shall not be started automatically and shall be added to the receiver's list of available applications. They can be started by using the "Application Life Cycle Management and Control" API (JAVADTV 1.3:2009)

© ABNT 2010 - All rights reserved

Table 2 (continued)

Code	Identifier	Description
0x03	DESTROY	Applications with the DESTROY control code shall be unconditionally shutdown by the application manager. Applications previously signaled with the STORE control code shall be removed from the cache
0x04	KILL	Applications with the KILL control code shall be shutdown by the application manager as soon as possible. If an exception of the type javax.microedition.xlet.XletStateChangeException is launched during an attempt to finalize the application, it shall continue running Applications previously signaled with the STORE control code can optionally be kept in the cache
0x05	Not defined	Reserved for future use
0x06	REMOTE	Applications with the REMOTE control code do not have their files transmitted in the current transport stream. The data source for these applications shall comply with the “Transport Protocol Descriptor” application descriptor (see ABNT NBR 15606-3:2007) Applications with the REMOTE control code shall not be started automatically and shall be added to the list of the receiver’s available applications. Can be initialized using the “Application Life Cycle Management and Control” API (see JAVADTV 1.3:2009)
0x07	UNBOUND	Applications with the UNBOUND control code are similar to applications flagged with PRESENT, except for the fact that its execution is not limited to a specific service. If the receiver does not have available storage capability to store the application or the user chooses not to install the application, this shall be assigned as non-available (it cannot be listed among the available applications). Receivers without application storage support shall ignore the applications flagged with this control code
0x08	STORE	Applications with the STORE control code shall not be started automatically, but indicate which caching techniques can be used to accelerate the loading their resources during startup (see 7.2 for further information) If the platform is not able to perform pro-active caching techniques or data storage, applications flagged as STORE shall be treated in the same way as applications flagged with PRESENT
0x09… 0xFF	Not defined	Reserved for future use

If an unknown control code is received, the application shall remain in the state it is in at that time. When a change in the control code brings about a change of state in a Ginga-J application, an event shall be generated for all Ginga-J applications that are registered to receive change notifications in the life cycle of the application in question.

© ABNT 2010 - All rights reserved

Optionally, the platform can provide a menu with a list of applications with STORED, PRESENT and UNBOUND flags (those that the user chose to install) so that the user can choose the right time to initialize them.

With the exception of those applications flagged with the field service_bound_flag initialized with 0, all applications shall be destroyed during a change of service. The execution of applications flagged with the field service_bound_flag setted for 0 is not restricted to a specific service and cannot be interrupted during the selection among various services (see 7.3.4 for further details). When changing the service, the receiver can always interrupt the execution of these applications if the liberation of resources is necessary.

7.2 Storage and caching of applications

7.2.1 Storage models

The Ginga-J application model permits applications to be stored and launched from persistent memory. Applications may be cached proactively or in response to a request from another application, subject to the user's consent and the platform's resource limits. In either case, the objective is to improve the application's load speed.

Thus, it may be useful for applications where it is desirable to have a quick display, avoiding lag in the startup thereof due to the unload time. These applications may be related to broadcast, in which case the application's life cycle is controlled by the broadcast of the AIT, or can be completely self-sufficient, in which case the AIT input for the application is stored along with the application's data.

Applications flagged with the STORE or UNBOUND control code in the AIT shall add extra information in the MANIFEST.MF file in a way as to indicate which files shall be stored. For each file that shall be stored, the input in the MANIFEST.MF for this file shall contain the value true for the "Persistent-Flag" attribute and a value for the "File-Version" attribute indicating the file version. Receivers that support the storage of applications shall verify the file version stored and update it when the application received is a later version. Furthermore, file inputs shall be added to the MANIFEST.MF for the application_id and organization_id, both flagged in the AIT for the application in question.

Stored applications shall be visible by the Application LifeCycle Management and Control" API (JAVADTV 1.3:2009), just as with any other applications.

7.2.2 Storage issues

The space available for application storage may not be sufficient to accommodate all applications flagged as persistent (STORE or UNBOUND), making it necessary to choose which of them shall be effectively persisted. Occasionally, it may be necessary to remove previously stored applications. In these cases, the application persistence and removal policy of applications flagged as STORE remains at the discretion of the receiver manufacturer's implementation. The applications flagged as UNBOUND shall be installed and removed with the user’s explicit authorization and shall have priority in relation to the applications flagged as STORE.

When the broadcaster signals a new version of a stored application, the middleware can overwrite the old version of the application with a new version. The time at which this happens is not predictable. If the old version is running at the time of updating, the platform may store both versions until the copy currently running ends. At that time, the old copy is removed from storage. If the platform chooses to delete the old copy before the application terminates, the behavior of the application currently running shall be the same (in practice, this means that all classes shall be loaded in memory before the old version is deleted).

© ABNT 2010 - All rights reserved

7.2.3 Proactive caching

When an application is flagged with the STORE control code, the platform is permitted to proactively store any files that are specified in the application description file.

However, it is not mandatory to fulfill the priority requests if the proactive application storage techniques are used. In particular, it is not mandatory for the proactive caching to store all files with critical priority.

7.3 Transmission of applications

7.3.1 Signaling rules

The default Ginga-J applications signaling rules shall comply with the provisions described in 7.1.9 and with ABNT NBR 15606-3:2007, 12.16.

7.3.2 Packaging of applications

Ginga-J applications shall be packaged, authenticated and authorized in accordance with the definitions specified in JAVADTV 1.3:2009. Each Ginga-J application may contain one or more JAR files (see JAR: 2009 and JVM: 1997). The main JAR file contains the application's class files, resource files and a manifest that describes the application and its requirements. The JAR signature mechanism permits JAR to be authenticated.

If the transmission is made by using the object carousel, the packaging should not be used in a JAR file, but the transmitted file system shall be organized in the same way.

If the application is transmitted by the interactive channel, it shall be sent in JAR files.

7.3.3 Application authentication

The authentication of Ginga-J applications shall comply with the Brazilian Standard in force at the time the application is transmitted.

NOTE The Brazilian Standard on application authentication is currently under development.

7.3.4 Signaling the same application in different departments

In order for an application to be considered as signaled in various services, the following conditions shall be met:

— the signaling shall be present in an AIT table in all services.

— the application identifier shall be the same in all services;

— the transport protocol descriptor shall be the same in all services Or the application is flagged with the UNBOUND control code and is already persisted in the receiver.

If besides the above described conditions the application is also flagged with the field service_bound_flag with the value 0, this application shall be kept running among all the services changes, unless there are any resources restrictions in the receiver.

7.3.5 Download of applications through the interactive channel

The application transport rules shall comply with ABNT NBR 15606-3.

Applications obtained through the interactive channel shall be contained in a JAR file (see 7.3.1). Compression support in the JAR file is optional.

© ABNT 2010 - All rights reserved

8 Ginga-J Platform

8.1 Java Platform

The Java platform used to run Ginga-J applications is defined according to PBP 1.1:2008.

The Java bytecode executed by the platform shall be version 45.3 to version 47.

8.2 Basic considerations of the platform

8.2.1 Execution environment

Each Ginga-J application shall be processed in an isolated environment of execution, that is, there shall be one system entity that represents one JVM where each application shall be executed without any interference in the execution of any other application. In order to do so, this execution environment shall allow each application to be executed by means of its own loader (ClassLoader) or even its own hierarchy of loaders to access classes that are not part of the Ginga-J base platform.

Competing applications shall not directly share instances of objects defined thereby. Any interaction between applications shall be possible only through a specific API. Each execution environment shall be established at the time the application is started and should be unloaded as soon as the application is destroyed. After the completion of each application, the Application Manager shall ensure that the class finalizers contained in the application are executed.

The Ginga applications can be executed in concurrent mode. Once an instance of an application has been loaded and started, the creation or initialization of another instance of such application shall not be permitted. Applications are differentiated by means of their respective organization_id and application_id values (see ABNT NBR 15606-3:2007).

Ginga-J applications should not synchronize in system classes or other static system instances. Otherwise, the expected behavior is undefined.

8.2.2 Hierarchy of packages and classes

Only methods and fields (and their respective dependencies) of the classes listed in this part of ABNT NBR 15606 shall be present in a Ginga implementation. Furthermore, where there is dependence on a specific package, the full inclusion thereof is permitted, but not mandatory. The behavior for additional classes and methods is not specified for applications sent via broadcast.

The classes, interfaces and methods listed or referenced herein that are marked as obsolete (depreciated), shall have their marks overwritten so that such classes, interfaces or methods become mandatory in this part of ABNT NBR 15606. It is strongly recommended that Ginga-J applications not use these depreciated elements, since they may no longer be supported in future versions of this part of ABNT NBR 15606.

The inclusion of any package herein does not directly involve the inclusion of its sub-packages.

Ginga-J applications should not define classes or interfaces in any package (or namespace) defined herein.

Implementation classes of the Ginga-J platform should not be contained in the empty package (default package).

© ABNT 2010 - All rights reserved

8.2.3 Notification of events

For all classes listed in this part of ABNT NBR 15606 in which there are methods for registering/ deregistering notifications, successive calls for registration of listeners shall have the same effect as a single call. Therefore, each event shall be notified only once per listener. Additionally, a request for registration cancellation should not take effect if the listener in question is not registered.

The number of processes (threads) used for the notification of events to the listeners, is implementation dependent. Ginga-J applications should not block the processing in their listeners in a way as to avoid that other listeners fail to be notified.

All classes of events listed in this part of ABNT NBR 15606 shall extend to the java.util.EventObject class.

8.2.4 Text coding

The standard text coding for the Ginga-J platform shall be UTF-8 in accordance with the java. io.DataOutput.writeUTF method (see PBP 1.1:2008). The "Latin1" standard shoall also be supported in accordance with ISO/IEC 8859-1:1998.

8.2.5 Life cycle of the applications

The state machine defined in 7.1.1 shall operate so that the behavior of applications complies with the following restrictions:

— the lag time perceived during startup of the application shall be as short as possible;

— an application can be destroyed at any time.

The Application Manager shall use the Xlet API (see PBP 1.1:2008) to order changes in the life cycle of the applications. Therefore, several factors can stimulate the Application Manager, such as:

— flags originating from the broadcasters (see 7.1.9);

— selection based on a property menu with a list of applications;

— order originated in another Ginga-J application by means of the "Application Lifecycle Management and Control" API (JAVADTV 1.3:2009) (see 7.1.6);

— order originated in NCL documents deploying one or more Xlets.

The application itself may decide to change its state. In order to do so, it shall use its instance of javax. microedition.xlet.XletContext to request such change from the Application Manager.

8.3 Common infrastructure

The APIs defined in (see CDC 1.1:2008; FP 1.1:2008 and PBP 1.1:2008) are included herein as the basis for the functioning of the Ginga-J platform.

The specific packages for digital television defined in JAVATV 1.1:2008 shall also be used in this Standard. Additionally, the following restrictions shall be compliant:

— the group of processes (threads) of a Ginga-J application shall not contain processes with priority higher than java.lang.Thread.NORM_PRIORITY;

© ABNT 2010 - All rights reserved

— the return of the System.currentTimeMillis method shall be synchronized with the date and time transmitted in the TOT;

— the return of the System.currentTimeMillis method shall have granularity less than or equal to 10 ms;

— the TimeZone used by the JVM shall be in accordance with the receiver's configurations.

— the java.util.Calendar shall be synchronized in accordance with the date and time transmitted in the TOT ;

— the default java.util.Locale of the JVM shall be defined as "pt_BR";

— the Ginga-J applications shall not use the java.util.TimeZone.setDefault method. The behavior of this method is implementation dependent;

— the System.out and System.err output streams shall be available for Ginga-J applications. However, the System.in input stream should not be available;

— the Runtime.traceInstructions and Runtime.traceMethodCalls methods shall be available without having any negative impact on the execution of the applications and without interference in the functioning of other API;

— the System.gc method is implementation dependent and has no defined behavior;

— the Runtime.gc method is implementation dependent and has no defined behavior;

— the platform should use mechanisms prescribed in java.lang.SecurityManager. checkPackageDefinition and java.lang.SecurityManager.checkPackageAccess, in order to prevent improper use of the system classes by the applications;

— the javax.tv.xlet package is considered obsolete (depreciated) for this specification. However, in order to enable greater integration with legacy applications, instances of javax.microedition. xlet.XletContext should also implement javax.tv.xlet.XletContext. Both interfaces contain similar methods and behaviors. Use of the javax.tv.xlet.XletContext interface shall be avoided for Ginga-J applications;

— the javax.tv.graphics package is considered obsolete (deprecated) for this specification. The classes and interfaces defined therein should not be used by Ginga-J applications;

— the minimum granularity for the javax.tv.util.TVTimer class shall be less than or equal to 10 ms;

— the smallest increment of repetition for the javax.tv.util.TVTimer class shall be 40 ms;

— calls to the javax.tv.util.TVTimer.scheduleTimerSpec method shall bring about exceptions of TVTimerScheduleFailedException type, if there are no timers available in the system.

8.4 Graphical presentation and events handling

8.4.1 LWUIT, LightWeight user interface toolkit

This Standard uses JAVADTV 1.3:2009 to define the graphical components and user events handling mechanism. The applications have graphical functionalities such as:

— high-level graphical components;

© ABNT 2010 - All rights reserved

— application of customizable themes to the graphical components;

— hierarchical treatment through containers and components;

— abstraction of components native to the system.

8.4.2 Graphical user interface

The graphical user interface will be made using the specific components for television applications and graphical components provided by the LWUIT 1,1:2008incorporated into JAVADTV 1.3:2009. The packages that define the graphical components are:

- com.sun.dtv.ui – defines the graphical components specifically related to television;
- com.sun.dtv.lwuit – contains the graphical components that support the creation of graphical user interfaces;
- com.sun.dtv.lwuit.animations – enables not only graphical components but also animated transitions between containers;
- com.sun.dtv.lwuit.geom – defines the basic geometric elements for drawing;
- com.sun.dtv.lwuit.layouts – defines useful types of graphic layouts;
- com.sun.dtv.lwuit.list – defines customizable list structures, used in components of other packages such as com.sun.dtv.lwuit;
- com.sun.dtv.lwuit.painter – permits graphical elements to be drawn arbitrarily based on flat, scaled and/or tiled images;
- com.sun.dtv.lwuit.plaf – permits customization of the appearance of the graphical components;
- com.sun.dtv.lwuit.util **–** utilities package.

The com.sun.dtv.ui.MatteEnabled interface, and the com.sun.dtv.ui.AnimatedMatte and com.sun.dtv.ui.StillMatte classes shall be present and implemented in order to maintain the contract with the applications. However, the functionality of graphic composition using the transparency information provided by the instances of these objects is optional for implementations of the Ginga-J platform.

8.4.3 Treatment of the platform planes

8.4.3.1 General considerations

The com.sun.dtv.ui package permits generic access to the planes offered by the platform for content display, organized into sections on the device's screen. According to ABNT NBR 15606-1, the organization of the planes, or layers, is presented as shown in Figure 5.

© ABNT 2010 - All rights reserved

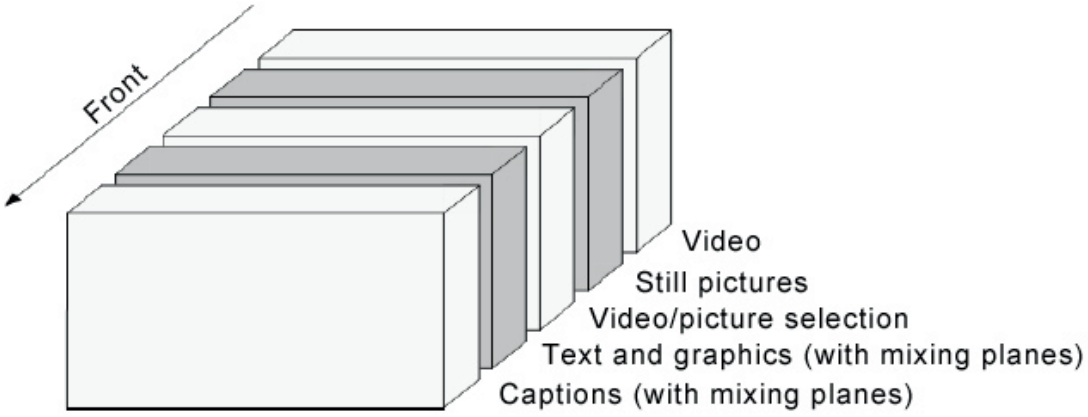

Figure 5 – Structure of layers for the presentation of services

The subtitles plane is not accessible by Ginga applications; this is a characteristic, which is native to the receiver. However, there remain four planes upon which a Java application can operate. Following this convention, they are always returned in this manner, the getAllPlanes() method of the com.sun.dtv.ui class.Screen (which in turn is obtained by means of the com.sun.dtv.ui.Device class):

— Plane[0]: Text and graphic plane;

— Plane[1]: Video/picture selection plane;

— Plane[2]: Still pictures plane;

— Plane[3]: Video plane.

For each of these planes, it is possible to obtain the characteristics thereof, and perform graphic operations thereon. These characteristics, and the operations supported by each plane, maintain compliance with the definitions set forth in ABNT NBR 15606-1. This detailing is done in the following subsections, where the return values are defined for each one of the methods, within the com.sun.dtv.ui.Capabilities object that corresponds to the respective plane.

8.4.3.2 Text and graphic plane

The text and graphic plane is the one on which the application can draw graphical elements (primitive geometrics and images) with high color definition and channel of transparency over the video.

The return of the getID (): “GraphicPlane” method is detailed in Table 3.

Table 3 – Details of functionalities for the text and grap

Function	Description
getBitsPerPixel()	32
getColorCodingModel()	ColorCoding.ARGB8888
getSupportedPixelAspectRatios()	A single Dimension(1,1) object is the recommended value (indicates pixels with a 1:1 ratio). Any value other than this is considered optional

© ABNT 2010 - All rights reserved

Table 3 (continued)

Function	Description
getSupportedPlaneAspectRatios()	A single Dimension object with construction value (16.9) or (4.3), depending on the configuration of the receiver's video output
getSupportedScreenResolutions()	A single Dimensionobject with the dimensions of the text and graphic plane at that moment
isAlphaBlendingSupported()	True
isGIFRenderingSupported()	True only for platforms that permit GIF images display in the graphic plane
isGraphicsRenderingSupported()	True
isImageRenderingSupported()	True
isJPEGRenderingSuppored()	True
isPNGRenderingSuppored()	True
isRealAlphaBlendingSupported()	True
isVideoRenderingSupported()	True only for platforms that permit display of video monomedia (or even an elementary video stream) in the graphic plane
isWidgetRenderingSupported()	True

The com.sun.dtv.ui.DTVContainer associated with this com.sun.dtv.ui.Plane shall support all types of com.sun.dtv.lwuit.component and graphical operations defined in the LWUIT API, except for the display of media types that are not permitted in the text and graphic plane (see ABNT NBR 15606-1).

8.4.3.3 Video/picture selection plane

The video/picture selection plane allows the definition of areas of precedence between the still picture plane and the video plane. In other words, in which rectangular areas of the screen the still pictures plane will be displayed over the video, or vice-versa.

The return of the getID: "SwitchingPlane" method is detailed in Table 4.

Table 4 – Details of the functionalities for video and still picture plane

Function	Description
getBitsPerPixel()	1
getColorCodingModel()	ColorCoding.ONE_BPP
getSupportedPixelAspectRatios()	Ditto as the GraphicPlane.
getSupportedPlaneAspectRatios()	Ditto as the GraphicPlane.

© ABNT 2010 - All rights reserved

Table 4 (continued)

Function	Description
getSupportedScreenResolutions()	A single Dimensionobject with the dimensions of the video at that moment.
isAlphaBlendingSupported()	True (The video/picture selection plane is the mode whereby the still picture plane implements alpha blending).
isGIFRenderingSupported()	False
isGraphicsRenderingSupported()	False
isImageRenderingSupported()	False
isJPEGRenderingSuppored()	False
isPNGRenderingSuppored()	False
isRealAlphaBlendingSupported()	False
isVideoRenderingSupported()	False
isWidgetRenderingSupported()	True

The com.sun.dtv.ui.DTVContainer associated with this com.sun.dtv.ui.Plane will support – in addition to the definition of a layout manager (setLayout () method) – only the addComponent() and removeComponent() calls, and only when the com.sun.dtv.lwuit.Component is passed as the parameter is of the br.org.sbtvd.ui.SwitchArea type (see Annex G). The other methods, although they do not generate errors in the application, have no effect because they refer to operations that aren't supported by the video/picture selection plane.

By means of the com.sun.dtv.lwuit.plaf.Style associated with the com.sun.dtv.ui.DTVContainer of this plane, the video content that will appear over the Still Picture Plane can be defined, and vice-versa. The instances of com.sun.dtv.lwuit.plaf.Style associated with this component may only contain solid colors (java.awt.Color). The color black, java.awt.Color. BLACK, represents that the video shall be displayed over the Still Picture Plane. With the use of any other color, the contents of the Still Picture Plane will appear in front of the video. Thus, the applications use a Java (RGB888 or ARGB8888) color model in order to control the video/image selection plane. This API implementation will make the conversion of the Java color model to the color model of the video/image selection plane, in accordance with the following function:

$$f(x)=\begin{cases} x = \text{java.awt.Color.BLACK} \Rightarrow \text{video selected} \\ x \neq \text{java.awt.Color.BLACK} \Rightarrow \text{still picture selected} \end{cases}$$

8.4.3.4 Still pictures plane

This is the plane upon which the application can display images with high resolution and color depth in the JPEG format. Typically, it is used to define a background plane for the application on screens where the video is resized. However, through combined use with the video/picture selection plane, it can be used to display high-resolution JPEG images over the video.

The return of the getID: "StillPlane" is detailed in Table 5.

© ABNT 2010 - All rights reserved

Table 5 – Details of functionalities for the still picture plane

Function	Description
getBitsPerPixel()	16
getColorCodingModel()	ColorCoding.YUV442
getSupportedPixelAspectRatios()	Ditto as the GraphicPlane
getSupportedPlaneAspectRatios()	Ditto as the GraphicPlane
getSupportedScreenResolutions()	A single Dimensionobject with the dimensions of the still picture plane at that moment
isAlphaBlendingSupported()	True (implemented through the video/picture selection plane)
isGIFRenderingSupported()	False
isGraphicsRenderingSupported()	False
isImageRenderingSupported()	True
isJPEGRenderingSuppored()	True
isPNGRenderingSuppored()	False
isRealAlphaBlendingSupported()	False
isVideoRenderingSupported()	False
isWidgetRenderingSupported()	True

The com.sun.dtv.ui.DTVContainer associated with this com.sun.dtv.ui.Plane will support – in addition to the definition of a layout manager (setLayout () method) – only the addComponent () and removeComponent () calls, and only when the component passed as the parameter is of the br.org. sbtvd.ui.StillPicture type (see Annex G). The other methods, although they do not generate an error in the application, have no effect because they refer to operations that are not supported by the still picture plane.

Through a com.sun.dtv.lwuit.plaf.Style associated to the com.sun.dtv.ui.DTVContainer object instance of this plane, a solid back color (without transparency) can be defined. This color shall be displayed as a content of the still picture plane in all regions of this plane that are not fulfilled by objects of the br.org.sbtvd.ui.StillPicture kind. This color shall be defined in the RGB888, color model, which is natural of Java's environment, being the conversion to a YUV442 color model made by the platform.

8.4.3.5 Video plane

The video plane displays the elementary video stream of the service, or (optionally) video monomedia. The content displayed in this plane can be handled through the JMF controls.

The return of the getID: "VideoPlane" is detailed Table 6.

© ABNT 2010 - All rights reserved

Table 6 – Details of the functionalities for the video plane

Function	Description
getBitsPerPixel()	Launches a SetupException.
getColorCodingModel()	Launches a SetupException.
getSupportedPixelAspectRatios()	Ditto as the GraphicPlane.
getSupportedPlaneAspectRatios()	Ditto as the GraphicPlane.
getSupportedScreenResolutions()	A single Dimensionobject with the dimensions of the video at that moment.
isAlphaBlendingSupported()	False (it is the plane located farthest back in the chain).
isGIFRenderingSupported()	False
isGraphicsRenderingSupported()	False
isImageRenderingSupported()	False
isJPEGRenderingSuppored()	False
isPNGRenderingSuppored()	False
isRealAlphaBlendingSupported()	False
isVideoRenderingSupported()	True
isWidgetRenderingSupported()	False

There is no com.sun.dtv.ui.DTVContainer object associated with this com.sun.dtv.ui.Plane. The invocation of any method other than getCapabilities () in this instance in particular, will have an implementation-dependent effect.

8.4.3.6 Planes Compostion

Picture 6 represents an example of composition between different planes. It is possible to observe how the video content is displayed by the black region in the video/image selection plane in the final composition. Similarly, the contents of the static images plane are displayed in front of the video in the video/image selection plane “white” region.

The text and graphic plane content is always displayed in front of the other planes.

© ABNT 2010 - All rights reserved

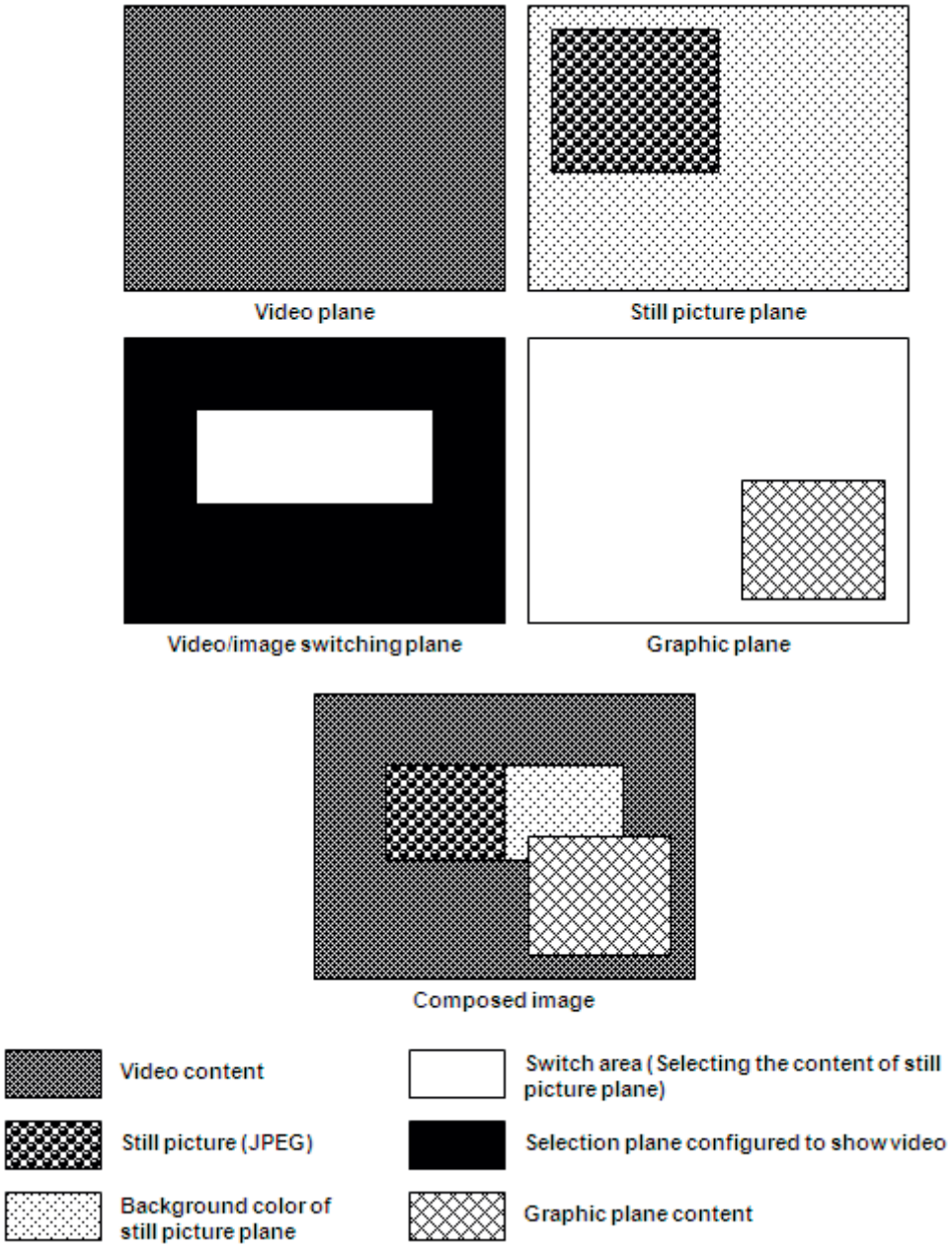

Figure 6 – Composition example of content display plane

8.4.4 User events handling

The user events handling mechanism is provided by specialized television components and the components provided by the LWUIT 1.1:2008 incorporated in to JAVADTV 1.3:2009. The packages that define these components are:

- com.sun.dtv.ui.event – specific television events handling mechanism;
- com.sun.dtv.lwuit .events – events handling mechanism related to the graphical components defined in the LWUIT 2008 incorporated into JAVADTV 1.3:2009.

The entire user events handling mechanism is based on the EDT (Event Dispatch Thread) model, giving the platform thorough knowledge of the events and passing them on to the applications and their specialized treatment.

© ABNT 2010 - All rights reserved

ABNT NBR 15604:2007, 7.2.28.3, defines the minimum functions on the remote control for receivers that feature an interactivity mechanism. These functions shall be mapped for the event types defined in the com.sun.dtv.lwuit.event.RemoteControlEvent class of JAVADTV 1.3:2009. Table 7 shows this mapping.

Table 7 – Mapping of functions of ABNT NBR 15604:2007 in the events defined by JAVADTV 1.3:2009

Functions defined in ABNT NBR 15604:2007		com.sun.dtv.lwuit.event.RemoteControlEvent class
Confirm		VK_CONFIRM
Exit		VK_ESCAPE
Back		VK_BACK
Directional/Arrow	Up	VK_UP & VK_KP_UP
	Down	VK_DOWN & VK_KP_DOWN
	Left	VK_LEFT & VK_KP_LEFT
	Right	VK_RIGHT e VK_KP_RIGHT
Coloured	Red	VK_COLORED_KEY_0
	Green	VK_COLORED_KEY_1
	Yellow	VK_COLORED_KEY_2
	Blue	VK_COLORED_KEY_3

8.5 Information and selection of services

8.5.1 General considerations

The API service information for this specification is responsible for providing the applications with access to the information contained in the MPEG service information tables. Such information shoall include the audio and video streams present in each multiplex service, also including a textual description of the services and events that comprise it, among others.

This API is based on the definition of the APIs described in ARIB STD-B.23: 2006, Annex M, which is based on the GEM and provides functionalities of access to service information in accordance with the reference model of the Japanese digital TV standard ARIB STD-B10: 2008, the same as extended by the SBTVD. However, since the core of this standard is the JAVADTV 1.3:2009 specification, it was necessary to make changes in the elements of the jp.or.arib.tv.si and jp.or. arib.tv.net package, aimed at integrating the functionality of protocol-dependent access to service information. Annex B provides details on these functionalities.

8.5.2 Integration with API protocol-independent

The protocol-independent API service information shall comply with JAVADTV 1.3:2009, 6.4.

© ABNT 2010 - All rights reserved

8.6 Presentation and execution of media

The service information API adopts the Java Media Framework (JMF) 1.0 and optionally may incorporate additional API defined by Java Media Framework (JMF) 2.1. The JMF2.1 is backward compatible with the JMF 1.0.

8.7 Access to data

8.7.1 General considerations

The java.io API (see PBP 1.1:2008) shall be used to access data objects in a generic way. Classes and interfaces contained in this package related to files and file systems shall comply with the following restriction:

— the java.io.ObjectInputStream.readLine method is marked as obsolete (depreciated) by this Standard. Therefore, it should not be used by applications sent via broadcast.

8.7.2 Access to files

For an application sent via broadcast and signaled in a particular service, data objects accessed by means of relative paths shall be queried based on the path indicated in the application's location descriptor “ginga_j_application_location_descriptor” (see ABNT NBR 15606-3:2007, 12.18.2), in the base_directory field. The path defined in the descriptor in question shall be considered as the application's base directory.

8.7.3 Broadcast transport protocol

When the application is transmitted via broadcast protocol, using the DSMCC Object Carousel or DSMCC Data Carousel protocols, they shall be supported by the extensions provided by the "Broadcast file and streams handling" (see JAVADTV 1.3:2009). This API provides access to internal stream data and files as an extension to the set of functionalities already available in the java.io package (see PBP 1.1:2008).

Clarifications about input/output operations for objects sent via broadcast can be found in the documentation of the com.sun.broadcast.BroadcastFile class (see JAVADTV 1.3:2009), as well as JAVADTV 1.3:2009 , 8.2.7.

The aforementioned clarifications shall also be applied to operations with data objects of the java.io.File type (see PBP 1.1:2008).

8.7.4 Persistent storage

For the purposes of persistent storage, the javax.microedition.io and java.io packages shall be available (see PBP 1.1:2008), as well as the "Persistent storage access rights and properties" API (see JAVADTV 1.3:2009).

The "com.sun.dtv.persistent.root” property should be accessible by means of the java.lang.System. getProperty method (see PBP 1.1:2008) and shall identify the root directory for persistent storage. Relative paths should not be used to access objects in persistent storage, under penalty of undefined functioning across platforms. The structure of directories prescribed for persistent storage is described in detail in JAVADTV 1.3:2009, 6.5.

© ABNT 2010 - All rights reserved

Access to files or directories above the application's base directory shall always result in an exception of the java.lang.SecurityException type (see PBP 1.1:2008) for Ginga-J applications.

Ginga-J applications that are signed, authenticated and have authorized permissions to access persistent storage, shall have the privileges provided in JAVADTV 1.3:2009, 6.5 granted.

For applications with access permissions granted, the directories and sub-directories that have access shall be created automatically by the platform if they do not already exist. It is recommended that required directories and sub-directories be created as soon as the permissions are granted. The identifier of the owner of the directories and sub-directories created automatically by the platform shall have the same value of the application_id (see ABNT NBR 15606-3:2007, 12.7) of the application in question.

Ginga-J applications shall create files or directories only where they have "write" permissions.

Details about the release of space by the platform are implementation-dependent. However, the persistence of the files shall be guaranteed while the application is running or flagged in the AIT table with a control code other than KILL or DESTROY.

8.7.5 Access to system properties

Access to the system properties will be done by the java.lang.System.getProperty,java.lang.System. getProperties and java.lang.System. setProperty methods.

The use of the java.lang.System.setProperties () method is not permitted for Ginga-J applications.

8.7.6 IP support over an interactive channel

For devices where the establishment (setup) of a connection is necessary, the direct use of java.net. Socket or java.net.URLConnection shall result in an attempt to connect according to the parameters sent in the application's permissions file. For further information, see "Per Policy Application Schema" (see JAVADTV 1.3:2009).

For handling of the interactivity devices available on the platform, the "Extensive communication control device" API shall be used (see JAVADTV 1.3:2009). Additional definitions on the use of the java.net default connection API (see PBP 1.1:2008) are found in the class documentation com.sun.dtv.net. NetworkDevice (see JAVADTV 1.3:2009), as well as in JAVADTV 1.3:2009, 6.2.5.

8.7.7 MPEG-2 section filtering

To get the MPEG-2 section filters, the "Support MPEG-2 Section Filtering" API shall be used (see JAVADTV 1.3:2009).

8.8 Application management

In this Standard, the digital television applications are called Xlets (see JAVATV 1.1:2008), and their life cycle is managed as described in 7.1. These applications execute in an services-oriented environment controlled by an Application Manager (see JAVADTV 1.3:2009), and this component is responsible for the actions of loading, configuring, instantiating and executing applications for digital television, as well as for controlling the life cycle and states of these applications. It is also the responsibility of the applications manager to attribute execution priority levels to the applications, as well as identify and mitigate any failures that occur during the execution thereof.

© ABNT 2010 - All rights reserved

All this management and monitoring is conducted internally by the system. Television applications duly signed and certified, as described in JAVADTV 1.3:2009 clause 6.2, are permitted to control and/or monitor the life cycle of these applications. These features are accessible in JAVADTV 1.3:2009 through the package:

- com.sun.dtv.application

8.9 Tuning

The br.org.sbtvd.net.tuning API is an extension of the com.sun.dtv.tuner package of JAVADTV 1.3:2009. The new functions are interactive channel zapping and scanning of all existing network interfaces on a receiver. Further details about these functionalities are available in Annex C.

8.10 NCL Bridge

The NCL bridge API contains the set of classes available for the bridge between information and process applications, in the Ginga environment. The functions available in the classes that are described below permit the development of Ginga-J procedure applications, including Ginga-NCL applications, and vice-versa. The Ginga-J NCL bridge API controls the presentation of an NCL document and offers resources so that a Ginga-J application included in an NCL document is notified about the occurrence of transition events made on the media node that encapsulates it.

If Java is instantied from NCL, the applications management shall be in compliance with ABNT NBR 15606-2:2007, 8.5. If NCL is instantied from Java, the applications management shall be in compliance with which is described in 8.8.

Complementary information can be found in ABNT NBR 15606-2:2007, 11.2 and 10.3.4.3 and Annex D.

8.11 Platform properties

8.11.1 System Properties

Ginga-J applications shall have access to platform properties that indicate general characteristics, such as supported profiles and/or features. Therefore, in addition to the properties already specified in the java.lang.System.getProperties (see PBP 1.1:2008) and the Ginga-NCL methods (see ABNT NBR 15606-2:2007, 7.2.4), the properties listed on Table 8 shall be available through the getProperty () and getProperties () methods of the java.lang.System class (see PBP 1.1:2008).

The system properties are described in Table 8 and cannot be changed by the Ginga-J applications.

Table 8 – System Properties

Name	Type of data	Description
com.sun.dtv.version	Alphanumeric	The version number of the JavaDTV specification implemented by the platform (see JAVADTV 1.3:2009)
com.sun.dtv.net.default.timeout	Numeric	Maximum wait time (timeout) for establishing a connection
system.gingaj_version	Alphanumeric	Version number of the Ginga-J specification implemented by the platform

© ABNT 2010 - All rights reserved

Table 8 (continued)

Name	Type of data	Description
system.internet_access	Boolean	Indicates whether the Internet access profile is available. (Possible values: “true” or “false”)
system.gingaj_profile	Alphanumeric	The Ginga-J specification profile number supported by the platform

8.11.2 User properties

In addition to the set of system properties, Ginga-J applications can maintain user-specific platform properties. For such, the "User preferences" API shall be used (see JAVADTV 1.3:2009).

The "User preferences" API (see JAVADTV 1.3:2009) shall also be used to receive notifications of changes to any system properties defined in 8.11.1.

8.12 Interactivity channel

Through this API, the Ginga system has the ability to identify the existence of the interactivity channel devices on the receiver, and may establish a connection if it is disconnected. Data communication after the established connection takes place through the java.net API. *

8.13 Complete list of Ginga-J packages

8.13.1 Java platform packages

The following packages (see CDC 1.1:2008; FP 1.1:2008; PBP 1.1:2008) are included by this part of ABNT NBR 15606:

- java.awt
- java.awt.color
- java.awt.event
- java.awt.font
- java.awt.im
- java.awt.image
- java.beans
- java.io
- java.lang
- java.lang.ref
- java.lang.reflect

© ABNT 2010 - All rights reserved

- java.math
- java.net
- java.rmi
- java.rmi.registry
- java.security
- java.security.acl
- java.security.cert
- java.security.interfaces
- java.security.spec
- java.text
- java.util
- java.util.jar
- java.util.zip
- javax.microedition.io
- javax.microedition.pki
- javax.microedition.xlet
- javax.microedition.xlet.ixc
- javax.security.auth.x500

8.13.2 JavaTV 1.1 and JMF 1.0 specification packages

The following packages (see JAVATV 1.1:2008) are included by this part of ABNT NBR 15606:

- javax.media
- javax.media.protocol
- javax.tv.graphics
- javax.tv.locator
- javax.tv.media
- javax.tv.net
- javax.tv.service

© ABNT 2010 - All rights reserved

- javax.tv.service.guide
- javax.tv.service.navigation
- javax.tv.service.selection
- javax.tv.service.transport
- javax.tv.util
- javax.tv.xlet

8.13.3 JavaDTV 1.1 specification packages

The following packages (see JAVADTV 1.3:2009) are included by this part of ABNT NBR 15606:

- com.sun.dtv.application
- com.sun.dtv.broadcast
- com.sun.dtv.broadcast.event
- com.sun.dtv.filtering
- com.sun.dtv.io
- com.sun.dtv.locator
- com.sun.dtv.lwuit
- com.sun.dtv.lwuit.animations
- com.sun.dtv.lwuit.events
- com.sun.dtv.lwuit.geom
- com.sun.dtv.lwuit.layouts
- com.sun.dtv.lwuit.list
- com.sun.dtv.lwuit.painter
- com.sun.dtv.lwuit.plaf
- com.sun.dtv.lwuit.util
- com.sun.dtv.media
- com.sun.dtv.media.audio
- com.sun.dtv.media.control
- com.sun.dtv.media.dripfeed

© ABNT 2010 - All rights reserved

- com.sun.dtv.media.format
- com.sun.dtv.media.language
- com.sun.dtv.media.text
- com.sun.dtv.media.timeline
- com.sun.dtv.net
- com.sun.dtv.platform
- com.sun.dtv.resources
- com.sun.dtv.security
- com.sun.dtv.service
- com.sun.dtv.smartcard
- com.sun.dtv.test
- com.sun.dtv.transport
- com.sun.dtv.tuner
- com.sun.dtv.ui
- com.sun.dtv.ui.event

8.13.4 JSSE 1.0.1 specification packages

The following packages (see JSSE 1.0.1) are included by this part of ABNT NBR 15606:

- com.sun.net.ssl
- javax.net
- javax.net.ssl
- javax.security.cert

8.13.5 JCE 1.0.1 specification packages

The following packages (see JCE 1.0.1) are included by this part of ABNT NBR 15606:

- javax.crypto
- javax.crypto.interfaces
- javax.crypto.spec

© ABNT 2010 - All rights reserved

8.13.6 SATSA 1.0.1 specification packages

The following package (see SATSA 1.0.1) is included by this part of ABNT NBR 15606:

- javax.microedition.apdu

8.13.7 Ginga-J specific packages

The following GINGA-J specific packages are included by this part of ABNT NBR 15606:

- br.org.sbtvd.net
- br.org.sbtvd.net.si
- br.org.sbtvd.net.tuning
- br.org.sbtvd.bridge
- br.org.sbtvd.ui

The minimum requirements for a receiver compatible with Ginga shall comply with ABNT NBR 15604:2007.

© ABNT 2010 - All rights reserved

Annex A
(normative)

Java DTV 1.3 specification

A.1 General considerations

The JAVADTV 1.3:2009 specification is comprised of the Java DTV API added to the component base of the Java execution environment, also including Connected Device Configuration, Foundation Profile, and Personal Basis Profile.

Java DTV is a specification that offers functionality of a digital television receiver for the development of Java-based applications. In this Annex, the main elements of the JAVADTV 1.3:2009 specification are described.

A.2 Java DTV API

NOTE For further details on the Java DTV API, see http://java.sun.com/javame/technology/javatv/index.jsp

A.2.1 com.sun.dtv.broadcast package

The com.sun.dtv.broadcast package permits access to broadcast files and streams and shall be implemented in accordance with JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.1, shall be used.

Table A.1 – Classes of the com.sun.dtv.broadcast package

Class	Description
BroadcastFile class	Represents file or directory data obtained from the broadcast channel
BroadcastFileEvent class	Indicates notifications of change for the BroadcastFile data
BroadcastFilesystem class	Represent instances of the file system obtained from the broadcast channel and assembled within the local file system
BroadcastFileListener interface	Implemented by application classes that wish to receive notifications of changes to the BroadcastFile data
BroadcastStream class	Represents streams obtained through files obtained from the broadcast channel
BroadcastException class	All broadcast-related exceptions shall use this subclass (which in turn is an extension of java.io.IOException) in order to facilitate the identification of the reasons that brought about the exception

© ABNT 2010 - All rights reserved

A.2.2 com.sun.dtv.smartcard package

The com.sun.dtv.smartcard package provides supports for additional functionalities for the use of smart cards.

The implementation of this package shall be in compliance with JAVADTV 1.3:2009, including the interfaces and classes described in Table A.2.

Table A.2 – Classes of the com.sun.dtv.smartcard package

Classes	Description
CardTerminalListener interface	Listener interface, responsible for receiving events originating from the device's smart card readers
PassThroughAPDUConnection interface	Marker interface that identifies the specific connection objects of the APDU (Application Protocol Data Unit), which perform direct communication with the smart card
TerminalFactory class	Provides access to the smart card reader device(s) implemented or connected to the device
CardTerminalEvent class	Defines an event object that informs the listeners about changes of state occurring in the smart card reader
CardTerminal class	Encapsulates the functionalities of the physical part of the smart card reader

A.2.3 com.sun.dtv.lwuit.events package

The com.sun.dtv.lwuit.events package implements the Observable design standard (also used in 1.1 AWT API) that defines an architecture to fire and handling events on graphical interfaces. All events are triggered by a thread called Event Dispatch Thread (EDT). See documentation on the API for further details.

This package shall be implemented according to JAVADTV 1.3:2009, where the interfaces and classes of this package, described in Table A.3, shall be used.

Table A.3 – Classes of the com.sun.dtv.lwuit.events package

Classes	Description
DataChangedListener interface	Interface callback events handling, which are invoked when changes in the state occur and indicate that the display shall be updated
FocusListener interface	Focus-change event listener for Form, permits features to be attributed according to the current focus of the component
SelectionListener interface	Invoked to indicate a change in the default list selection
StyleListener interface	Invoked to indicate changes in a property
ActionEvent class	Object event triggered when a callback is invoked
ActionListener interface	Callback events interface invoked when a component action occurs

© ABNT 2010 - All rights reserved

A.2.4 com.sun.dtv.filtering package

The com.sun.dtv.filtering package offers support for access to MPEG-2 transport stream sections. In addition to permitting various types of filtering, the API uses an asynchronous model that provides notification about events or errors in reading the sections.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.4, shall be used.

Table A.4 – Classes of the com.sun.dtv.filtering package

Classes	Description
DataSectionFilterCollection class	This class represents a collection of data section filters, where such filters may be activated and deactivated as a single operation
FilterTimedOutEvent class	This event is triggered if data section filter operations expire (time-out) according to the period specified by the setTimeout () method
DataSectionFilterException class	Base class for launching exceptions of the data section filter API
DataSectionAvailableEvent class	This event indicates that a data section has been completely processed
DataSectionFilter class	This is the root class of all filter classes
DataSection class	This class encapsulates a transport stream data section, where the objects of this class are also cloneable
IncompatibleSourceException class	Class that informs when an incompatible source stream is supplied
DataSectionDataBuffer class	This class encapsulates a part of the loaded transport stream section
FilteringStoppedEvent class	This class is used to inform the end of a filtering operation, with one exception: It is not informed when a SimpleSectionFilter ends in normal conditions (e.g., after a filter section has been performed successfully)
DataUnavailableException class	Informs when the information of a DataSection object are not available
DataSectionFilterEvent class	This is a base class for filter section API events
FilterInterruptException class	Signals that a filter was interrupted before all the data have been filtered
DisconnectedException class	Indicates that when the DataSectionFilterCollection lost the connection because of a flaw in the call of the startFiltering () method
InvalidFilterException class	Signals that a filter has been defined incorrectly

© ABNT 2010 - All rights reserved

Table A.4 (continued)

Classes	Description
FilterResourceUnavailableException class	Informs that the necessary requirements for an filtering operation have not been met
CircularFilter class	This class defines a filter section designed to capture continuous data streams without the need to restart the filter continuously
FilterEventListener interface	This listener interface can be implemented by classes that demand filtering events
ListFilter class	This class defines a filter section that can process a complete set of segments of a data section that represent a simple section table
SingleFilter class	This class defines a filter section destined to capture a single data section

A.2.5 com.sun.dtv.ui.event package

The com.sun.dtv.ui.event sub-package has the function of treating graphical user interface events specific to digital television.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.5, shall be used.

Table A.5 – Classes of the com.sun.ui.event package

Classes	Description
KeyListener interface	Functions only as an encapsulator to introduce java. awt.event.KeyListener in the LWUIT API, since it does not support key event objects and provides only an ActionListener
PlaneSetupEvent class	Event sent to all registered consumers when the configuration of a Plane is changed
MouseEvent class	Extends java.awt.event.MouseEvent implementing the com. sun.dtv.ui.eventUserInputEvent interface. Works only as an marker interface, because UserInputEvent is derived from the ScarceResource interface
UserInputEvent interface	Abstraction for user input events sent to all registered consumers whenever a new user input event occurs through a UserInputDevice
UserInputEventListener interface	Shall be implemented by classes that wish to receive UserInputEvent

© ABNT 2010 - All rights reserved

Table A.5 (continued)

Classes	Description
KeyEvent class	Extends java.awt.event.KeyEvent implementing the com. sun.dtv.ui.eventUserInputEvent interface. Works only as an marker interface, because UserInputEvent is derived from the ScarceResource interface
MouseListener interface	Functions only as an encapsulator to introduce java.awt. event.MouseListener in the LWUIT API, since it does not support mouse event objects and provides only an ActionListener
UserInputEventManager class	The instances of this class are events managers that treat user input events generated by graphical components
RemoteControlEvent class	Extends com.sun.dtv.ui.event.KeyEvent adding the television-specific key codes
PlaneSetupListener interface	Used to monitor changes in the configuration of a Plane

A.2.6 com.sun.dtv.lwuit.plaf package

The com.sun.dtv.lwuit.plaf package permits customization of the application's appearance ("look and feel"). In this case, it uses a rendering layer abstraction that can be coupled and configured (with themes, for example) in run time.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.6, shall be used.

Table A.6 – Classes of the com.sun.lwui.plaf package

Classes	Description
LookAndFeel class	Allows programmers to completely customize the appearance (look and feel) of the application, through appropriate methods of image superimposition and dimensioning
UIManager class	The central point for managing the look and feel of the application allows customization of styles (themes) and appearance
Style class	Represents the appearance of a particular component
Border class	Base class that permits the creation of a frame for a component, whereby the frame is drawn before the component filling in its marginal region
DefaultLookAndFeel class	Used to render the default appearance of the LWUIT

© ABNT 2010 - All rights reserved

A.2.7 com.sun.dtv.media.timeline package

The com.sun.dtv.media.timeline package permits the obtainment and definition of media timelines to be notified about time events launched during media playback.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.7, shall be used.

Table A.7 – Classes of the com.sun.dtv.media.timeline package

Classes	Description
MediaTimeListener interface	Listening interface for receipt of media time events
MediaTimePositionControl interface	Controls the definition and obtainment of mediaTime from a media stream. The maximum duration of the media can be obtained based on the Duration interface if it is implemented by the type of media in playback
MediaTimeEvent interface	Event that informs the application about changes in media time

A.2.8 com.sun.dtv.media.language package

The com.sun.dtv.media.language package provides the basic consulting and languages definition, for the audio, subtitles and Closed Captioning. The languages are coded in three languages in accordance with ISO 639-2 and ABNT NBR 15603-2 for use in service descriptors.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.8, shall be used.

Table A.8 – Classes of the com.sun.dtv.media.language package

Classes	Description
LanguageNotSupportedException class	Exception raised when a requested language is not supported
LanguageControl interface	Control for consulting supported languages for defining a specific language for a player

A.2.9 com.sun.dtv.application package

The com.sun.dtv.application package defines the JavaDTV applications model, its life cycle and management. JavaDTV applications run through an environment for Java DTV multi-task execution, and it is through this package that the developer can access this environment to verify which applications are available (AppManager), the attributes of each application (Application) and perform monitoring and control over such an application (ApplicationProxy). This package also implements the application's life cycle through the use of the JavaTV API classes.

This package shall be implemented according to JAVADTV 1.3:2009. The following interfaces and classes of this package, described in Table A.9, shall be included.

© ABNT 2010 - All rights reserved

Table A.9 – Classes of the com.sun.dtv.application package

Classes	Description
AppManagerListener interface	Listener that notifies about changes in the available applications
AppManager class	Provides control and access to the applications
ApplicationProxy interface	Provides control over an application through the application manager
Application interface	Contains the attributes of an application
AppFilter class	AppFilter is called to permit an application in a filter, or not
AppManagerPermission class	Required for queries or control of applications
AppProxyListener interface	Listener that receives notifications of changes of state of the application

A.2.10 com.sun.dtv.media.audio package

The com.sun.dtv.media.audio package offers functionalities regarding audio language control, for example:

— consult the languages that can be selected for association with a Service Component;

— select the language associated with a Service Component

— permit the applications to receive notifications when the language of a Service Component is changed.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.10, shall be used.

Table A.10 – Classes of the com.sun.dtv.media.auto package

Classes	Description
AudioEvent interface	Indicates changes related to audio language selection
AudioControl interface	Provides audio control to register specific audio events
AudioListener interface	Listener for change in audio

A.2.11 com.sun.dtv.test package

The com.sun.dtv.test package provides a framework for compliance testing. Since it is usually found in Java test environments, the environment defined for using the package involves a test client and server that can use any means of communication.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.11, shall be used.

© ABNT 2010 - All rights reserved

Table A.11 – Classes of the com.sun.dtv.test package

Classes	Description
TestCase interface	Defines the necessary methods that shall be implemented by the tests in order to be able to execute the test framework
TestHarness class	Provides an entry point and a set of tools for test cases
Report class	Adds the result of a test: the code, the reference for the test and the related reasons

A.2.12 com.sun.dtv.tuner package

The com.sun.dtv.tuner package provides an API for access and control of a broadcast network interface (or "tuner") used to receive transport streams.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.12, shall be used.

Table A.12 – Classes of the com.sun.dtv.tuner package

Classes	Description
Tuner class	Represents a network interface or “tuner” for receiving broadcast transport streams
TuningCompletedEvent class	Event that indicates the successful completion of a tuning operation
TuningFailedEvent class	Event that indicates the unsuccessful completion of a tuning operation
TuningException class	Exception that indicates synchronous reports of tuning failure
TuningEvent class	Base class for events generated by tuning operations
TunerListener interface	Listener that receives tuning events coming from the Tuner
TuningInitiatedEvent class	Indicates the start of a tuning operation

A.2.13 com.sun.dtv.lwuit.layouts package

The com.sun.dtv.lwuit.layouts package contains the management model of the LWUIT layout, which is similar to the model used by the AWT/Swing APIs. In this case, the layout managers permit a Container to arrange its components through a set of pre-defined rules that can be adapted to specific font or screen sizes.

This package shall be implemented in accordance with JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.13, shall be used.

© ABNT 2010 - All rights reserved

Table A.13 – Classes of the com.sun.dtv.lwuit.layouts package

Classes	Description
GridLayout class	By using this class as a layout manager, the components are arranged on a grid of equal size, based on available space
BorderLayout class	Defines a container that organizes and redimensions its components in a layout based on five regions: north, south, east, west and center
LayoutStyle class	LayoutStyle is used to determine how much space is used between components
BoxLayout class	Arranges elements in a row or column, according to a box orientation
FlowLayout class	Arranges the components in a row so that when a line end is reached, the components go on to the next line
Layout class	Abstract class, which can be used to organize components in a container using a predefined algorithm
GroupLayout class	Layout manager that groups hierarchical components
CoordinateLayout class	Permits the components based on absolute positions and sizes to be adjusted based on available space for the layout

A.2.14 com.sun.dtv.broadcast.event package

Treats events generated by the broadcast channel. In this case the BroadcastEventManager class deals with all events and passes them on to the registered BroadcastEventListener.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.14, shall be used.

Table A.14 – Classes of the com.sun.dtv.broadcast.event package

Classes	Description
BroadcastReceivedEvent class	Represents the events that were received through the broadcast channel
BroadcastEventManager class	Represents the events obtained through the file system of the broadcast channel
BroadcastEventListener Interface	Implemented by the application classes that request notification of receipt of BroadcastEvent data

A.2.15 com.sun.dtv.lwuit.list package

The com.sun.dtv.lwuit.list package related to the handling of List that uses the same Swing MVC model, including the renderer design standard. The package has two interfaces and two classes. The ListCellRenderer interface permits customization of the appearance of the List and the ListModel defines the representation of a data structure that is used by the List as the source of its information. The classes are default implementations of the interfaces.

© ABNT 2010 - All rights reserved

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.15, shall be used.

Table A.15 – Classes of the com.sun.dtv.lwuit.list package

Classes	Description
DefaultListCellRenderer class	Default implementation of the renderer based on a label
ListModel interface	Represents the data structure of the list, thus permitting a list to represent any potential data source through the reference to several implementations of this interface
ListCellRenderer interface	Interface that defines a label renderer of a List. In this case, represents only the marker of each cell selected in the List
DefaultListModel class	Default implementation of the list model based on an elements vector

A.2.16 com.sun.dtv.ui package

The com.sun.dtv.ui package is a package with graphical user interface functionalities specific to digital television.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.16, shall be used.

Table A.16 – Classes of the com.sun.dtv.ui package

Classes	Description
MatteException class	A MatteException is triggered when an application, for any reason, is unable to perform an association with a graphical element
DTVContainerPattern class	The DTVContainerPattern is a means to describe the characteristics of a valid DTVContainer
Device class	This class is the representation of the television device
Screen class	This class is the representation of the television device screen
DefaultTextLayoutManager class	This class provides a standard mechanism for text rendering
FontSpecificationException class	Exception launched when an attempt to specify characteristics of a font in an incorrect manner. In this case, only for fonts that are not defined in a file
ViewOnlyComponent interface	This class represents any type of non-interactive component of the system. Also uses a mechanism for configuring the appearance thereof
UserInputDevice class	Basis for all input devices that can be used to control the device screen
Mouse class	This class represents the mouse that can be used to control a particular screen of a UserInputDevice

© ABNT 2010 - All rights reserved

Table A.16 (continued)

Classes	Description
AlphaComposite class	Implements all of the alpha composition (transparency) rules.
DTVContainer class	A high-level container in the hierarchy of components of the Java DTV API that represents a graphical element containing anything visible on a particular Plane. In this case, there is only one DTVCointaner for each Plane
TextLayoutManager interface	Defines the functionalities for the layout of texts and the display thereof on screen
Capabilities class	Describes the capabilities of a Plane
SetupException class	Exception that can be launched in several situations where an attempt is made to perform an illegal change in the configuration of one or more planes of a Screen
SophisticatedTextLayoutManager class	This class provides a TextLayoutManager with more functionalities in a way as to permit a more sophisticated layout
PlaneSetupPattern class	This class provides a means of describing the configurations of a viewing plane, specifying various properties and the importance thereof for the application
AnimatedMatte class	This class represents an animated matte with a dynamic image mask, where the pixel values determine the transparency of the matte at any given time
RemoteControl class	This class represents a television remote control that can be used to control a particular screen as a UserInputDevice
Interface TextOverflowListener	Notifies if a character string does not fit into a component during the attempt to render it
Keyboard class	This class represents a keyboard that can be used to control certain screens as a UserInputDevice
FontFileException class	This exception will be launched in an attempt to read a font file with an inappropriate format
PlaneSetup class	Describes the characteristics of a Plane
DownloadableFont class	Introduces the possibility of downloading fonts
Matte interface	Basic interface for all Matte classes
Animated interface	This interface provides methods to define and obtain parameters of an animation
MatteEnabled interface	Permits components to make matte composition
StaticMatte class	This class represents non-animated mattes, for example, mattes that do not change during an increment of time
Plane class	Represents a video output from a television device

© ABNT 2010 - All rights reserved

A.2.17 com.sun.dtv.media.control package

The com.sun.dtv.media.control package has additional controls to obtain information about the media being displayed.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.17, shall be used.

Table A.17 – Classes of the com.sun.dtv.media.control package

Classes	Description
FrameRateControl interface	A control to obtain the frame rate
MpegAudioControl interface	A control to obtain the parameters of an MPEG audio stream
BitRateControl interface	A control to obtain the bit rate

A.2.18 com.sun.dtv.media.dripfeed package

The com.sun.dtv.media.dripfeed package allows the delivery of data from a still picture to a JMF Player, in a way that permits the application to have control over the data. The pictures can be from frames captured from a video source.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.18, shall be used.

Table A.18 – Classes of the com.sun.dtv.media.dripfeed package

Classes	Description
DripFeedControl interface	Permits the progressive feeding of parts of a video on a Player
DripFeedPermission class	Represents the permissions to access the DripFeedControl

A.2.19 com.sun.dtv.security package

The com.sun.dtv.secutity package includes additional security functionalities. The basic functions are provided by the classes in the java.security package.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.19, shall be used.

Table A.19 – Classes of the com.sun.dtv.security package

Classes	Description
CallbackHandler interface	An application implements a CallbackHandler and passes it on to the security services. Thus, the application can interact with the security services with the aim of retrieving specific authentication information such as username and password, or to display certain information such as error messages or warnings

© ABNT 2010 - All rights reserved

Table A.19 (continued)

Classes	Description
Callback interface	Implementations of this interface are passed to a CallbackHandler, allowing security services to interact with the application called and to retrieve specific authentication information such as username and password, or to display certain information such as error messages or warnings
AuthProvider class	This class defines login and logout methods for a provider
LoginException class	Exception related to login operations
UnsupportedCallbackException class	Triggered when a callback call is passed and cannot be treated by the receiver of the call

A.2.20 com.sun.dtv.lwuit.painter package

The com.sun.dtv.lwuit.painter package contains classes that extend functionalities of the com.sun.dtv.lwuit.Painter interface and that permit one to draw arbitrary graphic elements in the background of components (com.sun.dtv.lwuit.Component).

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.20, shall be used.

Table A.20 – Classes of the com.sun.dtv.lwuit.painter package

Classes	Description
PainterChain class	Permits the linking of several painters so as to achieve a “layering” effect and where each painter draws only one element
BackgroundPainter class	Draws the screen background of a component based on its style

A.2.21 com.sun.dtv.locator package

The com.sun.dtv.locator package defines all of the Locators to be used in the Java DTV system.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.21, shall be used.

Table A.21 – Classes of the com.sun.dtv.locator package

Classes	Description
EntityLocator class	Entity locator in the stream transport sectors
URLLocator class	URL-based locator
TransportDependentLocator interface	Locator that references the entities of a transport stream
NetworkBoundLocator Classe	Locator that references entities that are network bound

© ABNT 2010 - All rights reserved

A.2.22 com.sun.dtv.resources package

The com.sun.dtv.resources package provides a basic framework for devices with limited resources.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.22, shall be used.

Table A.22 – Classes of the com.sun.dtv.resources package

Classes	Description
TimeoutException class	Signals when a timeout occurs
ScarceResourceListener interface	Notifies about the release of a particular scarce resource
ScarceResource interface	Represents resources that need special treatment to reserve and release
ScarceResourcePermission class	Used to deal with the various permissions relating to scarce resources
ResourceTypeListener interface	Notifies the status of changes occurring in resources of the same object type to which the listener has connected

A.2.23 com.sun.dtv.net package

The com.sun.dtv.net package extends the java.net package to support the extensive communication control of with devices. In this case it represents a device through the NetworkDevice class.

This package shallbe implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.23, shall be used.

Table A.23 – Classes of the com.sun.dtv.net package

Classes	Description
NetworkDeviceStatusListener interface	Listener for events related to network devices
NetworkDevicePermission class	Used to deal with the various permissions relating to scarce resources of network devices
NetworkDevice class	Represents each physical instance of any network interface in IP protocol support (TCP, UDP) The communication can be obtained through the platform

A.2.24 com.sun.dtv.media.text package

The package com.sun.dtv.media.text permits access to the Control key and "Closed Captioning". This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.24, shall be used.

© ABNT 2010 - All rights reserved

Table A.24 – Classes of the com.sun.dtv.media.text package

Classes	Description
OverlayTextEvent interface	Events that report changes in: “OverlayText”, “Subtitle” and “Closed Captioning”
OverlayTextControl interface	Provides control over “Subtitles and “Closed Captioning”
OverlayTextListener interface	Receives events related to “OverlayText “

A.2.25 com.sun.dtv.media.format package

The com.sun.dtv.media.format package is responsible for the video format settings.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.25, shall be used.

Table A.25 – Classes of the com.sun.dtv.media.format package

Classes	Description
VideoFormatControl interface	Provides the means to obtain information about the format and aspect ratio of the video
VideoFormatListener interface	Listening interface that receives notification of events on changes in the video display (presentation)
AspectRatioEvent Interface	The VideoFormatEvent interface informs on changes occurring in the aspect ratio
VideoPresentationControl interface	Provides the means to consult and manipulate the video display (presentation)
VideoPresentationEvent class	Event that informs about changes occurring in the video display
ActiveFormatEvent interface	The VideoFormatEvent informs about the changes in the Active Format
VideoPresentationListener interface	Reports on changes in the video display, as well as all types of ContollerEvents
DecoderFormatEvent interface	Event that informs that the decoder format has changed
ClippingControl interface	Control that retrieves and defines the rectangular cut of the video
VideoFormatEvent interface	Event that informs about changes the in video format
BackgroundVideoPresentationControl interface	Control of videos shown in the screen background
ArbitraryVideoScalingControl interface	Control for retrieving the arbitrary factors of the video scale
Transformation class	Represents a container for transformation information for a video

© ABNT 2010 - All rights reserved

A.2.26 com.sun.dtv.platform package

The com.sun.dtv.platform package provides classes that are specific to the Java DTV platform, in particular the classes related to the treatment of users.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.26, shall be used.

Table A.26 – Classes of the com.sun.dtv.platform package

Classes	Description
UserPropertyPermission class	Describes permissions for the user's properties
UserPropertyListener interface	As an alternative, an application can attach a UserPropertyListener to the sub-system user properties, in order to be notified of any changes in the user properties
User class	Contains several fields and methods that are specific to each user of the platform

A.2.27 com.sun.dtv.io package

The com.sun.dtv.io package extends the java.io package, providing access to the rights and properties of the files.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.27, shall be used.

Table A.27 – Classes of the com.sun.dtv.io package

Classes	Description
FileProperties class	Used to associate properties (or methods) to a file identified by its "pathname" in a given file system
FileAccessRights class	Provides a means to define groups of levels of access rights to a file or directory

A.2.28 com.sun.dtv.lwuit.animations package

In the com.sun.dtv.lwuit.animations package, all of the components are potential animations and can be executed in run time; transitions between Forms are also treated as part of package. The animation threads are treated uniformly in order to reduce the complexity for execution on computationally limited devices.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.28, shall be used.

© ABNT 2010 - All rights reserved

Table A.28 – Classes of the com.sun.dtv.lwuit.animations package

Classes	Description
Transition class	Represents an animated transition between two Forms, this class is used internally by a DTVContainer to reproduce an animation when shifting from one Form to the next
CommonTransitions class	Contains common transition animations
Motion class	Abstraction of the notion of physical movement over time between two points represented by numerical values
Animation interface	Permits any component to receive animation events at time increments and to be updated

A.2.29 com.sun.dtv.service package

The com.sun.dtv.service package provides an interface to access the SI (Service Information) database.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.29, shall be used.

Table A.29 – Classes of the com.sun.dtv.service package

Classes	Description
SIDatabase class	Generically provides access to the SI database that resides on the platform

A.2.30 com.sun.dtv.media package

The com.sun.dtv.media package is for the relevant functionalities and for the freeze and resume controls.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.30, shall be used.

Table A.30 – Classes of the com.sun.dtv.media package

Classes	Description
FreezeResumeListener interface	Permits the application to execute playback “freeze” and “resume” events
FreezeResumeEvent interface	Indicates if the freeze or resume events happened and identifies if they such events originated from an application or a user
FreezeEvent interface	Indicates whether an freeze action happened originating from an application or a user
MediaPresentedEvent interface	This event is generated after a javax.media.Player has been transferred to the initial state

© ABNT 2010 - All rights reserved

Table A.30 (continued)

Classes	Description
FreezeResumeException class	This exception indicates that either the freeze method or resume method was unsuccessful
FreezeResumeControl interface	Shall be implemented to permit the application to freeze the Player
ConditionalAccessException class	Indicates that a media on the control of a Player or Data Source is protected by conditional access
ResumeEvent interface	Indicates that the action to continue playback happened originating from an application or a user

A.2.31 com.sun.dtv.transport package

The com.sun.dtv.transport package provides access to the entities contained in a transport stream.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.31, shall be used.

Table A.31 – Classes of the com.sun.dtv.transport package

Classes	Description
TransportStream class	Representation of a transport stream and its associated services
ConditionalAccessDeniedException class	This class is launched upon a request for access to information that is coded and whose access is not permitted by the security system
ElementaryStream class	Representation of an elementary stream
Service class	Representation of a service contained in the transport stream

A.2.32 com.sun.dtv.lwuit.util package

The com.sun.dtv.lwuit.util package provides utility functionalities that are domain-specific or are not suited for any other package in the API.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.32, shall be used.

Table A.32 – Classes of the com.sun.dtv.lwuit.util package

Classes	Description
Log class	Permits the developer – through a pluggable logging framework – to utilize log functionalities using the file connector API
Resources class	This is related to the loading resources (animations, images, themes, fonts, etc.) from a binary file generated in the build process

© ABNT 2010 - All rights reserved

A.2.33 com.sun.dtv.lwuit package

The com.sun.dtv.lwuit package contains the main hierarchy of graphical elements composition (com.sun.dtv.lwuit.Component and com.sun.dtv.lwuit.Container) of the LWUIT API that follows the same model as the Swing/AWT API. However, unlike Swing/AWT, a system of full-screen windows is not used. In this case, a model is used similar to the MIDP API, which uses a display abstraction in which the graphics can be arranged.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.33, shall be used.

Table A.33 – Classes of the com.sun.dtv.lwuit package

Classes	Description
MediaComponent class	Permits the insertion and control of rich-media content
StaticAnimation class	An image capable of animation
Graphics class	Abstracts the graphic context platform, permitting portability between MIDP and CDC devices
Container class	Implements the Composite default design for com.sun.dtv.lwuit. Component so as to permits the arrangement and relationship of components using an architecture of pluggable layout managers
ComboBox class	Graphic element that represents a list that permits only one selection at a time through the user's choice
Font class	A simple abstraction of platform fonts and library that permits use of fonts that are not supported by the device
Painter interface	This interface can be used to draw on screen background components.
RadioButton class	Specific type of com.sun.dtv.lwuit.Button that maintains a state of selection exclusively of a com.sun.dtv.lwuit.KeyGroup
Calendar class	Permits the selection of date and time values
TabbedPane class	Permits the user to toggle between a group of components by clicking on a tab with a particular title and/or icon
Command class	Action relating to the "soft buttons" and device menu, similar to the abstraction of the MIDP command and Swing actions
CheckBox class	Button that can be marked or unmarked and simultaneously display its status to the user
Form class	High-level component that is the base class for LWUIT1.1:2008 graphical user interfaces. The container is divided into 3 (three) parts: Title (title bar, usually located at the top), ContentPane (central space available for the layout of Title and MenuBar) elements and MenuBar (menu bar usually located at the bottom)

© ABNT 2010 - All rights reserved

Table A.33 (continued)

Classes	Description
Dialog class	A type of Form that occupies a part of the screen and appears as a modal entity to the developer
Image class	Abstraction of the platform that treats images, permitting the handling thereof as uniform objects
TextField class	Component for receiving user text input that uses a lighter API, without using the device's native text support
ButtonGroup class	This class is used to create a multiple-exclusion scope for a set of Radio Buttons
Label class	Permits the display labels and images with different alignment options, also functions as a base class for alignment of layout components
Button class	Base component for other graphical elements that are clickable
List class	A set (list) of elements that are created using a ListCellRenderer and are extracted through the ListModel
Component class	Base class for all graphical elements of the LWUIT. Uses the Composite default design in a similar way to the AWT's relation of Container and Component
TextArea class	Graphical component that permits text input with multiple editable lines also permits display and editing of text
AWTComponent class	Extends the com.sun.dtv.lwuit.Component class as a special variant that delegates the rendering actions to java.awt.Component

A.2.34 com.sun.dtv.lwuit.geom package

Contains classes related to the geometric location and calculation of dimensions of graphical components.

This package shall be implemented according to JAVADTV 1.3:2009. The interfaces and classes of this package, described in Table A.34, shall be used.

Table A.34 – Classes of the com.sun.dtv.lwuit.geom package

Classes	Description
Point class	Represents a location in space of x and y coordinates. Its accuracy is based on whole numbers
Rectangle class	Represents a rectangular with the size based on width and height, it is useful for measuring coordinates within an application
Dimension class	Utility class that stores values of width and height and represents a dimension of a graphical component or element

© ABNT 2010 - All rights reserved

Annex B
(normative)

Specification of the protocol-dependent service information API

B.1 General considerations

This Annex describes the protocol-dependent service information API in Ginga-J. This API is based on changes in specification ARIB STD-B23: 2004, Annex M. This is due to the adoption of ISDB-T (see ARIB STD-B31: 2007) based on ABNT NBR 15601:2007. Parameters relevant to the information about digital television service are associated with the transmission method used; hence, the Brazilian service information specified by ABNT NBR 15603:2007 is largely consistent with specification ARIB STD-B10.2008. However, the ARIB STD-B23: 2004 is based on the GEM API. This Standard is compatible with the Java DTV platform, which makes it necessary to introduce adjustments in the SI API. Therefore, a new API was defined and specified.

B.2 Protocol-dependent service information API

B.2.1 br.org.sbtvd.net package

B.2.1.1 SBTVDLocator class

SBTVD Locator encapsulates SBTVD URL in the object. This class extends the com.sun.dtv.locator. EntityLocator class (see JAVADTV 1.3:2009).

The public methods of the SBTVDLocator class are:

- SBTVDLocator(java.lang.String url) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid) throws javax.tv.locator. InvalidLocatorException
    - Generation of SBTVD Locator based on the following format.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid) throws javax.tv.locator. InvalidLocatorException
    - Generation of SBTVD Locator based on the following format.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid) throws javax. tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on the following format.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on the following format.

© ABNT 2010 - All rights reserved

- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on one of the following formats.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int[] componenttags) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on one of the following formats.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int[] componenttags, java.lang.String filePath) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on one of the following formats.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, int channelid) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on one of the following formats.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, java.lang.String modulename) throws javax.tv.locator.InvalidLocatorException
    - Generation of SBTVD Locator based on one of the following formats.
- SBTVDLocator(java.lang.String scheme, int onid, int tsid, int serviceid, int contentid, int eventid, int componenttag, java.lang.String modulename, java.lang.String resourcename)
    - Generation of SBTVD Locator based on one of the following formats.
- int getChannelId ()
    - Retrieves channel id.
- int[] getComponentTags()
    - Retrieves component_tag array.
- int getContentId()
    - Retrieves the content_id.
- int getEventId()
    - Retrieves the event_id.
- java.lang.String getFilePath()
    - Retrieves part of the file name locator path.
- java.lang.String getModuleName()
    - To acquire moduleName.

© ABNT 2010 - All rights reserved

- int getOriginalNetworkID()
    - Retrieves the original_network_id.
- java.lang.String getResourceName()
    - Retrieves the resourceName.
- java.lang.String getScheme()
    - Retrieves the scheme.
- int getServiceID()
    - Retrieves the service_id.
- int getTransportStreamID()
    - Retrieves the transport_stream_id.
- java.net.URL getURL()
    - Retrieves the SBTVD URL encapsulated in the SBTVDLocator object.

B.2.1.2 SBTVDNetworkBoundLocator class

SBTVDLocator is connected to the network. This object type uniquely identifies a certain entity including the distribution of the system that transmits the entity. For example, if two types of networks transmit a particular service, it can be identified as a common service on the SBTVDLocator. However, each service transmitted has a different SBTVDNetworkBoundLocator. This class implements the com.sun.dtv.locator.TransportDependentLocator interface (see JAVADTV 1.3:2009) and extends the br.org.sbtvd.net.SBTVDLocator class.

The public methods of the SBTVDNetworkBoundLocator class are:

- SBTVDNetworkBoundLocator(SBTVDLocator unboundLocator, int networkId)
    - Generation of the network bound locator.
- int getNetworkID()
    - Retrieves the network_id.

B.2.2 br.org.sbtvd.si package

B.2.2.1 DescriptorTag interface

The DescriptorTag interface defines the constants that correspond to the most common values of the descriptor tag.

The public static constants of the DescriptorTag class are:

- static short AUDIO_COMPONENT
    - The constant indicates the value of the audio component descriptor tag specified in ARIB STD-B10.

© ABNT 2010 - All rights reserved

- static short BASIC_LOCAL_EVENT
    - The constant indicates the value of the basic local event descriptor tag specified in ARIB STD-B10.
- static short BOARD_INFORMATION
    - The constant indicates the value of the board information descriptor tag specified in ARIB STD-B10.
- static short BOUQUET_NAME
    - The constant indicates the value of the bouquet name descriptor tag specified in ARIB STD-B10.
- static short BROADCASTER_NAME
    - The constant indicates the value of the broadcaster name descriptor tag specified in ARIB STD-B10.
- static short CA_CONTRACT_INFO
    - The constant indicates the value of the CA contractor information descriptor tag specified in ARIB STD-B10.
- static short CA_EMM_TS
    - The constant indicates the value of the CA_EMM_TS descriptor tag specified in ARIB STD-B10.
- static short CA_IDENTIFIER
    - The constant indicates the value of the CA identification descriptor tag specified in ARIB STD-B10.
- static short CA_SERVICE
    - The constant indicates the value of the CA service descriptor tag specified in ARIB STD-B10.
- static short CABLE_DELIVERY_SYSTEM
    - The constant indicates the value of the cable delivery system descriptor tag specified in ARIB STD-B10.
- static short CAROUSEL_COMPATIBLE_COMPOSITE
    - The constant indicates the value of the carousel compatible composite descriptor tag specified in ARIB STD-B10.
- static short COMPONENT
    - The constant indicates the value of the component descriptor tag specified in ARIB STD-B10.
- static short COMPONENT_GROUP
    - The constant indicates the value of the component group descriptor tag specified in ARIB STD-B10.

© ABNT 2010 - All rights reserved

- static short CONNECTED_TRANSMISSION
    - The constant indicates the value of the connected transmission descriptor tag specified in ARIB STD-B10.
- static short CONTENT
    - The constant indicates the value of the content descriptor tag specified in ARIB STD-B10.
- static short CONTENT_AVAILABILITY
    - The constant indicates the value of the contents availability descriptor tag specified in ARIB STD-B10.
- static short COUNTRY_AVAILABILITY
    - The constant indicates the value of the country receiving availability descriptor tag specified in ARIB STD-B10.
- static short DATA_COMPONENT
    - The constant indicates the value of the data component descriptor tag specified in ARIB STD-B10.
- static short DATA_CONTENTS
    - The constant indicates the value of the data contents descriptor tag specified in ARIB STD-B10.
- static short DIGITAL_COPY_CONTROL
    - The constant indicates the value of the digital copy control descriptor tag specified in ARIB STD-B10.
- static short DOWNLOAD_CONTENT
    - The constant indicates the value of the download contents descriptor tag specified in ARIB STD-B10.
- static short EMERGENCY_INFORMATION
    - The constant indicates the value of the emergency information descriptor tag specified in ARIB STD-B10.
- static short EVENT_GROUP
    - The constant indicates the value of the event group descriptor tag specified in ARIB STD-B10.
- static short EXTENDED_BROADCASTER
    - The constant indicates the value of the extended broadcaster descriptor tag specified in ARIB STD-B10.
- static short EXTENDED_EVENT
    - The constant indicates the value of the extended event descriptor tag specified in ARIB STD-B10.

© ABNT 2010 - All rights reserved

- static short HIERARCHICAL_TRANSMISSION
    - The constant indicates the value of the hierarchical transmission descriptor tag specified in ARIB STD-B10.
- static short HYPER_LINK
    - The constant indicates the value of the hyper link descriptor tag specified in ARIB STD-B10.
- static short LDT_LINKAGE
    - The constant indicates the value of the LDT linkage descriptor tag specified in ARIB STD-B10.
- static short LINKAGE
    - The constant indicates the value of the linkage descriptor tag specified in ARIB STD-B10.
- static short LOCAL_TIME_OFFSET
    - The constant indicates the value of the local time offset descriptor tag specified in ARIB STD-B10.
- static short LOGO_TRANSMISSION
    - The constant indicates the value of the logo transmission descriptor tag specified in ARIB STD-B10.
- static short MOSAIC
    - The constant indicates the value of the mosaic descriptor tag specified in ARIB STD-B10.
- static short NETWORK_IDENTIFICATION
    - The constant indicates the value of the network identification tag descriptor specified in ARIB STD-B10.
- static short NETWORK_NAME
    - The constant indicates the value of the network name descriptor tag specified in ARIB STD-B10.
- static short NODE_RELATION
    - The constant indicates the value of the node relation descriptor tag specified in ARIB STD-B10.
- static short NVOD_REFERENCE
    - The constant indicates the value of the NVOD reference service descriptor tag specified in ARIB STD-B10.
- static short PARENTAL_RATING
    - The constant indicates the value of the parental rating descriptor tag specified in ARIB STD-B10.

© ABNT 2010 - All rights reserved

- static short PARTIAL_RECEPTION
    - The constant indicates the value of the partial reception descriptor tag specified in ARIB STD-B10.
- static short PARTIAL_TRANSPORT_STREAM
    - The constant indicates the value of the partial transport stream descriptor tag specified in ARIB STD-B10.
- static short PARTIALTS_TIME
    - The constant indicates the value of the partial transport stream time descriptor tag specified in ARIB STD-B10.
- static short REFERENCE
    - The constant indicates the value of the reference descriptor tag specified in ARIB STD-B10.
- static short SATELLITE_DELIVERY_SYSTEM
    - The constant indicates the tag value for the satellite delivery system descriptor specified in ARIB STD-B10.
- static short SERIES
    - The constant indicates the tag value for the series descriptor specified in ARIB STD-B10.
- static short SERVICE
    - The constant indicates the tag value for the service descriptor specified in ARIB STD-B10.
- static short SERVICE_LIST
    - The constant indicates the tag value for the service list descriptor specified in ARIB STD-B10.
- static short SHORT_EVENT
    - The constant indicates the tag value for the short form event descriptor specified in ARIB STD-B10.
- static short SHORT_NODE_INFORMATION
    - The constant indicates the tag value for the short form node information descriptor specified in ARIB STD-B10.
- static short SI_PARAMETER
    - The constant indicates the tag value for the SI transmission parameter descriptor specified in ARIB STD-B10.
- static short SI_PRIME_TS
    - The constant indicates the tag value for the SI prime TS descriptor specified in ARIB STD-B10.

© ABNT 2010 - All rights reserved

- static short STC_REFERENCE
    - The constant indicates the tag value for the STC reference descriptor specified in ARIB STD-B10.
- static short STREAM_IDENTIFIER
    - The constant indicates the tag value for the stream identification descriptor specified in ARIB STD-B10.
- static short STUFFING
    - The constant indicates the tag value for the stuffing descriptor specified in ARIB STD-B10.
- static short SYSTEM_MANAGEMENT
    - The constant indicates the tag value for the system management descriptor specified in ARIB STD-B10.
- static short TARGET_AREA
    - The constant indicates the tag value for the target area descriptor specified in ARIB STD-B10.
- static short TERRESTRIAL_DELIVERY_SYSTEM
    - The constant indicates the tag value for the terrestrial delivery system descriptor specified in ARIB STD-B10.
- static short TIME_SHIFTED_EVENT
    - The constant indicates the tag value for the time shifted event descriptor specified in ARIB STD-B10.
- static short TIME_SHIFTED_SERVICE
    - The constant indicates the tag value for the time shifted service descriptor specified in ARIB STD-B10.
- static short TS_INFORMATION
    - The constant indicates the tag value for the TS information descriptor specified in ARIB STD-B10.
- static short VIDEO_DECODE_CONTROL
    - The constant indicates the tag value for the video decode control descriptor specified in ARIB STD-B10.

B.2.2.2 PMTElementaryStream interface

The PMTElementaryStream interface indicates the elementary stream for the service (channel). The PMT is present in each service to describe the elementary streams of the service. This means that the object set up with the interface indicates one of these elementary streams. Each object set up with the PMTElementaryStream interface is identified by a combination of original_network_id, transport_stream_id, service_id and component_tag (or elementary_PID).

© ABNT 2010 - All rights reserved

The public methods for the PMTElementaryStream class are:

- SBTVDLocator getSBTVDLocator ()
    - Recovers the SBTVDLocator that identifies the elementary stream.
- int getComponentTag ()
    - Recovers the component tag.
- short getElementaryPID ()
    - Recovers the elementary_PID.
- int getOriginalNetworkID ()
    - Recovers the original network ID.
- int getServiceID ()
    - Recovers the service ID.
- byte getStreamType ()
    - Recovers the stream type ID for the elementary stream.
- int getTransportStreamID ()
    - Recovers the transport stream ID.

B.2.2.3 PMTService interface

The PMTService interface indicates the specific service to be transmitted by the transport stream. The information is recovered from the PMT. Each object set up with the PMTElementaryStream interface is identified by a combination of original_network_id, transport_stream_id and service_id.

The public methods for the PMTService class are:

- SBTVDLocator getSBTVDLocator()
    - Recovers the SBTVDLocator that identifies this service.
- int getOriginalNetworkID ()
    - Recovers the original network ID.
- int getPcrPid()
    - Recovers the PCR's PID.
- int getServiceID()
    - Recovers the service ID.

© ABNT 2010 - All rights reserved

- int getTransportStreamID()
    - Recovers the transport stream ID.
- SIRequest retrievePMTElementaryStreams(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] somePMTDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - Recovers the information relevant to the elementary streams that comprise this service from the PMT.

B.2.2.4 PMTStreamType interface

The PMTStreamType interface defines the constants that correspond to the various stream types.

The PMTStreamType class' public static constants are:

- static byte DSMCC_DATA_CAROUSEL
    - The constant indicates the data carousel stream type defined by ISO/IEC 13818-1.
- static byte INDEPENDENT_PES
    - The constant indicates the independent PES stream type defined by ISO/IEC 13818-1.
- static byte MPEG1_AUDIO
    - The constant indicates the MPEG1 audio stream type defined by ISO/IEC 13818-1.
- static byte MPEG1_VIDEO
    - The constant indicates the MPEG1 video stream type defined by ISO/IEC 13818-1.
- static byte MPEG2_AAC_AUDIO
    - The constant indicates the MPEG2AAC audio stream type defined by ISO/IEC 13818-1.
- static byte MPEG2_AUDIO
    - The constant indicates the MPEG2 audio stream type defined by ISO/IEC 13818-1.
- static byte MPEG2_VIDEO
    - The constant indicates the MPEG2 video stream type defined by ISO/IEC 13818-1.
- static byte MPEG4_VIDEO
    - The constant indicates the MPEG4 video stream type defined by ISO/IEC 13818-1.
- static byte MPEG4_AVC_VIDEO
    - The constant indicates the H.264/MPEG-4 AVC video stream type defined by ISO/IEC 13818-1.

© ABNT 2010 - All rights reserved

B.2.2.5 SIBouquet interface

The SIBouquet interface indicates the sub-table of the Bouquet Association Table's (BAT), which describes a specific bouquet (with the SITransportStreamBAT). Each object that sets up the SIBouquet interface is identified by the bouquet_id identifier. This interface extends br.org.sbtvd.si.SIInformation. The public methods for the SIBouquet class are:

- int getBouquetID()
  - Recover bouquet ID.
- short[] getDescriptorTags()
  - This method defines additional semantics for the SIInformation#getDescriptorTags.
- java.lang.String getName()
  - This method returns the name of the bouquet to be described in the bouquet descriptor.
- SBTVDLocator[] getSIServiceLocators()
  - This method recovers the SBTVDLocators list to identify the service that belongs to the service.
- SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener)
  - This method defines additional semantics for the first SIInformation#retrieveDescriptors prototype.
- SIRequest retrieveDescriptors(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
  - This method defines additional semantics for the second SIInformation#retrieveDescriptors prototype.
- SIRequest retrieveSIBouquetTransportStreams(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
  - This method provides significant information for the transport stream to which the bouquet belongs.

B.2.2.6 SI Broadcaster interface

The interface indicates the specific provider in the service. The information returned by the methods is acquired from BIT. Each object that implements the SI Broadcaster interface is identified by the broadcaster_id.

The public methods for the SI Broadcaster class are:

- int getBroadcasterID ()
  - Returns the provider's ID.

© ABNT 2010 - All rights reserved

- boolean getBroadcastViewProperty ()
    - Returns the broadcaster's display property value.
- java.lang.String getName ()
    - Returns the name of the provider to be described in the provider descriptor.
- int getOriginalNetworkID()
    - Returns the original network ID.
- SBTVDLocator[] getSIServiceLocators()
    - This method recovers the SBTVDLocators list to identify the service that belongs to the provider.
- SIRequest retrieveOriginalNetworkDescriptors(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method recovers all descriptors transmitted in the first BIT loop.
- SIRequest retrieveOriginalNetworkDescriptors(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method recovers the set of descriptors transmitted in the first BIT loop.

B.2.2.7 SIEvent interface

The SIEvent interface indicates a specific program in the service. Each object that implements the SIEvent interface is identified by a combination of original_network_id, transport_stream_id, service_id and event_id. If the method's return value is acquired in the simple manner from the event descriptor and more than one descriptor is present, the following algorithm shall be used. In case the language returned by javax.tv.service.SIManager#getPreferredLanguage is used in the simple event descriptor, the value is returned from the descriptor. Otherwise, it depends on the set up status that shall be used besides the available simple event descriptors. This interface extends br.org.sbtvd.si.SIInformation.

The public methods for the SIEvent class are:

- SBTVDLocator getSBTVDLocator()
    - Recover the SBTVDLocator that identifies the program.
- java.lang.String[] getAudioComponentDescriptions()
    - This method returns the description in elementary audio stream text relevant to the program.
- java.lang.String[] getComponentDescriptions()
    - This method returns the description in elementary stream text relevant to the program.
- byte[] getContentNibbles()
    - This method returns the program genre.

© ABNT 2010 - All rights reserved

- java.lang.String[] getDataContentDescriptions()
    - This method returns the description in text relevant to the data broadcasting program.
- long getDuration()
    - This method recovers program duration.
- int getEventID()
    - This method recovers event ID.
- SIExEventInformation[] getExEventInformations()
    - This method returns detailed information relevant to the program.
- boolean getFreeCAMode ()
    - This method recovers the program shuffle value.
- byte[] getLevel1ContentNibbles()
    - This method returns the first step of program content classification.
- java.lang.String getName()
    - This method returns the name of the program.
- int getOriginalNetworkID()
    - This method recovers the original network ID.
- byte getRunningStatus()
    - This method recovers program execution status.
- java.lang.String getSeriesName()
    - This method returns the name of the relevant series to the program.
- int getServiceID()
    - This method recovers the service ID.
- java.lang.String getShortDescription()
    - This method returns the program description.
- java.util.Date getStartTime()
    - This method recovers the time the program started.
- int getTransportStreamID()
    - This method recovers the transport stream ID.

© ABNT 2010 - All rights reserved

- byte[] getUserNibbles()
    - This method returns the genre relevant to the program.
- SIRequest retrieveSIService(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method recovers the SIService that indicates the service. The service is the one to which the program indicated by SIEvent belongs.

B.2.2.8 SIInformation interface

The SIInformation interface is a collection of functions, which are common to SIBouquet, SIBroadcaster, SINetwork, SITransportStream, SIService, PMTService, SIEvent, SITime and PMTElementaryStream.

The SIInformation class public static constants are:

- static short FROM_CACHE_ONLY
    - The constant is used for the acquisition mode parameter of the acquisition methods.
- static short FROM_CACHE_OR_STREAM
    - The constant is used for the acquisition mode parameter of the acquisition methods.
- static short FROM_STREAM_ONLY
    - The constant is used for the acquisition mode parameter of the acquisition methods.

The public methods for the SIInformation class are:

- boolean fromActual()
    - If the information contained in the object that implements this interface was selected from the “actual” table or the table that does not distinguish “actual” or “not actual”, “true” is returned.
- com.sun.dtv.transport.TransportStream getDataSource()
    - This method returns the com.sun.dtv.transport.TransportStream object selected from the information contained in the object that implements this interface.
- short[] getDescriptorTags()
    - This method recovers the tag values for all descriptors that are part of the object’s current version.
- SIDatabase getSIDatabase()
    - This method returns the hierarchical structure root to which the object that implements this interface belongs.
- java.util.Date getUpdateTime()
    - This method returns the last day and hour the information included in the object that implements this interface was updated.

© ABNT 2010 - All rights reserved

- SIRequest retrieveDescriptors (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method returns all descriptors in the order they were sent.
- SIRequest retrieveDescriptors (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method recovers some descriptors.

B.2.2.9 SIIterator interface

The object that implements the SIIterator interface can access the content through the collection of SI objects. In order to maintain collection consistency, some accesses to the stream are not initiated depending on access to the content.

The public method for the SIIterator class is:

- int numberOfRemainingObjects()
    - The number of objects maintained in the iterator.

B.2.2.10 SIMonitoringListener interface

The SImonitoringListener interface is implemented by the application class, in order to receive the SI object monitoring changes.

The public method for the SIMonitoringListener class is:

- void postMonitoringEvent(SIMonitoringEvent anEvent)
    - This method is called by the SI's API to inform the listener of the event.

B.2.2.11 SIMonitoringType interface

The SIMonitoringType interface defines the constants that correspond to each type of SI information in the SIMonitoringEvent.

The SIMonitoringType class' public static constants are:

- static byte BOUQUET
    - Constant of the SIInformation object that indicates the bouquet.
- static byte BROADCASTER
    - Constant of the SIInformation object that indicates the provider.
- static byte NETWORK
    - Constant of the SIInformation object that indicates the network.
- static byte PMT_SERVICE
    - Constant of the SIInformation object that indicates the PMT service.

© ABNT 2010 - All rights reserved

- static byte PRESENT_FOLLOWING_EVENT
    - Constant of the SIInformation object that indicates the EIT [Present/Following].
- static byte SCHEDULED_EVENT
    - Constant of the SIInformation object that indicates the EIT [Schedule].
- static byte SERVICE
    - Constant of the SIInformation object that indicates the service.

B.2.2.12 SINetwork interface

The SINetwork interface indicates the Network Information Table (NIT) sub-table that describes a specific network (with the SITransportStreamNIT). Each object that implements the SINetwork interface is identified by the network_id.

The public methods for the SINetwork class are:

- short[] getDescriptorTags()
    - This method defines additional semantics for the SIInformation#getDescriptorTags method.
- java.lang.String getName()
    - This method returns the name of the network described in the Network Name Descriptor.
- int getNetworkID()
    - Recovers the network ID from this network.
- SIRequest retrieveDescriptors(shortretrieveMode,java.lang.ObjectappData, SIRetrievalListener listener) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method defines additional semantics for the first prototype of the SIInformation#retrieve Descriptors method.
- SIRequest retrieveDescriptors(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method defines additional semantics for the second prototype of the SIInformation#retri eveDescriptors method.
- SIRequest retrieveSITransportStreams(shortretrieveMode,java.lang.ObjectappData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd. si.SIIllegalArgumentException
    - This method recovers information about the transport stream to be transmitted over the network.

B.2.2.13 SIRetrievalListener interface

The SIRetrievalListener interface shall be implemented to receive an SI event.

© ABNT 2010 - All rights reserved

The public method for the SIRetrievalListener class is:

- void postRetrievalEvent(SIRetrievalEvent event)
    - This method is called from the implemented SI API in order to notify conclusion of the listener's request.

B.2.2.14 SIRunningStatus interface

The SIRunningStatus interface defines the constant that corresponds to the execution status value for the service and for the event.

The SIRunningStatus class' public static constants are:

- static NOT_RUNNING
    - constant byte, as defined in ARIB STD-B10, indicates the status is "not running".
- static PAUSING
    - constant byte, as defined in ARIB STD-B10, indicates the status is executing "pausing".
- static RUNNING
    - constant byte, as defined in ARIB STD-B10, indicates the status is "running".
- static STARTS_IN_A_FEW_SECONDS
    - constant byte, as defined in ARIB STD-B10, indicates the status is "ready to start in a few seconds".
- static UNDEFINED
    - constant byte, as defined in ARIB STD-B10, indicates the status is "undefined".

B.2.2.15 SIService interface

The SIService interface indicates a specific service that is transmitted by one of the transport streams. The information obtained through this interface's method is acquired from the SDT. Each object set up with the SIService interface is identified by a combination of the following ID:

Original network ID, Transport stream ID, Service ID

The public methods for the SIService class are:

- SBTVDLocator getSBTVDLocator()
    - This method acquires the SBTVDLocator to identify this service.
- boolean getEITPresentFollowingFlag()
    - This method returns the EIT_present_following_flag value.
- boolean getEITScheduleFlag()
    - This method returns the EIT_schedule_flag value.

© ABNT 2010 - All rights reserved

- int getEITUserDefinedFlag()
    - This method returns the EIT_user_defined_flags value.
- boolean getFreeCAMode()
    - This method returns the free_CA_mode value.
- java.lang.String getName()
    - This method returns the name that indicates the service included in the service descriptor.
- int getOriginalNetworkID()
    - This method recovers the original network ID.
- java.lang.String getProviderName()
    - This method returns the name of the service provider included in the service descriptor.
- byte getRunningStatus()
    - This method acquires the execution status for this service.
- int getServiceID()
    - This method acquires the ID service.
- short getSIServiceType()
    - This method acquires the type of service.
- int getTransportStreamID()
    - This method acquires the transport stream ID.
- SIRequest retrieveFollowingSIEvent(shortretrieveMode,java.lang.ObjectappData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the next EIT program [Current / next].
- SIRequest retrievePMTService(shortretrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This PMTService method acquires the relevant information for this service.
- SIRequest retrievePresentSIEvent(shortretrieveMode,java.lang.ObjectappData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the current EIT program [Current / next].

© ABNT 2010 - All rights reserved

- SIRequest retrieveScheduledSIEvents(shortretrieveMode,java.lang.ObjectappData, SIRetrievalListener listener, short[] someDescriptorTags, java.util.Date startTime, java.util.Date endTime) throws br.org.sbtvd.si.SIIllegalArgumentException, br.org.sbtvd.si.SIInvalidPeriodException
    - This method acquires the relevant information for the program projected in the designated EIT period [Calendar].

B.2.2.16 SIServiceType interface

This API is responsible for access to information related to the definition of ServiceType.

The SIServiceType class' public static constants are:

- static BOOKMARK_LIST
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “data list marker”.
- static DATA
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “data service”.
- static DATA_EXCLUSIVE_FOR_ACCUMULATION
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “data service exclusively for accumulation”.
- static DATA_FOR_ACCUMULATION_IN_ADVANCE
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “data service exclusively for in-advance accumulation”.
- static DIGITAL_AUDIO
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is “digital audio”.
- static DIGITAL_TELEVISION
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is “digital TV”.
- static ENGINEERING_DOWNLOAD
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is “download engineering”.
- static PROMOTION_DATA
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “data promotion”.
- static PROMOTION_SOUND
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “sound promotion”.

© ABNT 2010 - All rights reserved

- static PROMOTION_VIDEO
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “video promotion”.
- static SPECIAL_AUDIO
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “special audio”.
- static SPECIAL_DATA
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “special data”.
- static SPECIAL_VIDEO
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is a “special video”.
- static UNKNOWN
    - Constant, short type, as defined in ARIB STD-B10, indicates the service type is an “unknown”.

B.2.2.17 SITime interface

The SITime interface provides access to Time and Date Table (TDT) information. If the object indicates TDT, the retrieveDescriptors and getDescriptorTags methods behave as specified when there is no description, because TDT does not have descriptors.

This interface indicate time and date obtained from the table (TDT). If the object indicates TDT, retrieveDescriptors and getDescriptorTags behavior is similar to when there is no descriptor (because the TDT Descriptor is not found).

The public method for the SITime class is:

- java.util.Date getTime()
    - This method acquires the encrypted time in the TDT or the TOT.

B.2.2.18 SITransportStream interface

The SITransportStream interface is the base interface used to indicate relevant information for the transport stream.

The method that recovers the transport stream in the SIDatabase class and the SINetwork interface returns the object set up with the SITransportStreamNIT interface that refers to the NIT. The method that recovers the transport stream in the SIBouquet interface returns the object set up with the SITransportStreamBAT interface that refers to the BAT.

The public methods for the SITransportStream class are:

- SBTVDLocator getSBTVDLocator()
    - This method acquires the SBTVDLocator that identifies the transport stream.

© ABNT 2010 - All rights reserved

- int getOriginalNetworkID()
    - This method acquires the original network ID.
- int getTransportStreamID()
    - This method acquires the transport stream ID.
- SIRequest retrieveSIServices(short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for the service to be transmitted by the stream.

B.2.2.19 SITransportStreamBAT interface

The SITransportStreamBAT interface indicates information from the transport stream recovered from BAT. All of the methods that access descriptors return information from descriptors acquired from BAT. The method that recovers the transport stream in the SIBouquet returns an object that implements this interface.

The public method for the SITransportStreamBAT class is:

- int getBouquetID()
    - The method acquires the ID from the bouquet to which this transport stream belongs.

B.2.2.20 SITransportStreamNIT interface

The SITransportStreamNIT interface indicates information from the transport stream acquired from NIT. All of the methods that access descriptors return information from descriptors acquired from NIT. The method that recovers the transport stream in the SIDatabase or SINetwork returns an object that implements this interface.

The public method for the SITransportStreamNIT class is:

- int getNetworkID()
    - This method acquires the ID from the network to which this transport stream belongs.

B.2.2.21 Descriptor class

The Descriptor class indicates sub-table descriptors.

The public methods for the Descriptor class are:

- byte getByteAt(int index) throws java.lang.IndexOutOfBoundsException
    - This method obtains the value of one byte from the descriptor data section.
- byte[] getContent()
    - This method acquires a copy of the descriptor data section (bytes that are after the byte that indicates descriptor size).

© ABNT 2010 - All rights reserved

- Short getContentLength()
    - This method returns the length of the data section indicated in the “descriptor length” field.
- short getTag()
    - This method acquires the tag from the descriptor.

B.2.2.22 SIDatabase class

The SIDatabase class indicates the hierarchical structure root for SI information. There is one SIDatabase for each network interface. Thus, there is only one SIDatabase, if there is only one network interface.

The SIDatabase class' public static constants are:

- static int RETRIEVE_ALL_INFORMATIONS
- static int RETRIEVE_CURRENT_SELECTED

The public methods for the SIDatabase class are:

- void addBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of bouquet information.
- void addBroadcasterMonitoringListener(SIMonitoringListener listener, int broadcasterId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of broadcaster information.
- void addEventPresentFollowingMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of EIT information [Current / Next].
- void addEventScheduleMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId, java.util.Date startTime, java.util.Date endTime) throws br.org.sbtvd.si.SIIllegalArgumentException, br.org.sbtvd.si. SIInvalidPeriodException
    - This method initializes the accompaniment of EIT information [schedule].
- void addNetworkMonitoringListener(SIMonitoringListener listener, int networkId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of network information.
- void addPMTServiceMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of PMT information relevant to the service.

© ABNT 2010 - All rights reserved

- void addServiceMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method initializes the accompaniment of SDT information relevant to the service.
- static SIDatabase[] getSIDatabase()
    - This method returns the SIDatabase object (for each network interface).
- void removeBouquetMonitoringListener(SIMonitoringListener listener, int bouquetId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the listener registry from the bouquet information monitoring event.
- void removeBroadcasterMonitoringListener(SIMonitoringListener listener,int broadcasterId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the listener registry from the broadcaster information monitoring event.
- void removeEventPresentFollowingMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the listener registry from the EIT information monitoring event [current/next].
- void removeEventScheduleMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - Corresponding to the scheduled total, this method eliminates the registry of the EIT monitoring event [schedule].
- void removeNetworkMonitoringListener(SIMonitoringListener listener, int networkId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the registry of the network monitoring event.
- void removePMTServiceMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the listener registry from the monitoring event for PMT information relevant to the service.
- void removeServiceMonitoringListener(SIMonitoringListener listener, int originalNetworkId, int transportStreamId) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method removes the listener registry from monitoring for information relevant to the service.
- SIRequest retrieveActualSINetwork(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for the current network.

© ABNT 2010 - All rights reserved

- SIRequest retrieveActualSIServices(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for this service.
- SIRequest retrieveActualSITransportStreaM (short retrieveMode, java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags). throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for the stream.
- SIRequestretrievePMTElementaryStreams(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, SBTVDLocator sbtvdLocator, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires information about PMT's elementary stream relevant to the service component of this SIDatabase stream.
- SIRequest retrievePMTElementaryStreams(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener,int serviceId,int componentTag, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires information about PMT's elementary stream relevant to the service component of this SIDatabase stream.
- SIRequest retrievePMTService(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, SBTVDLocator sbtvdLocator, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires PMT information relevant to this service.
- SIRequest retrievePMTServices(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, int serviceId, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the PMT information relevant to the service from this SIDatabase's transport stream.
- SIRequest retrieveSIBouquets(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the stream.
- SIRequest retrieveSIBroadcaster(short retrieveMode,java.lang.ObjectappData, SIRetrievalListener listener, intoriginalNetworkId, int broadcasterId, short [] some DescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the broadcaster.
- SIRequest retrieveSIBroadcasters(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, int originalNetworkId, short [] some DescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the broadcaster specified by originalNetworkId.

© ABNT 2010 - All rights reserved

- SIRequest retrieveSINetworks(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, int networkId, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the information relevant to the network.
- SIRequest retrieveSIService(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, SBTVDLocator sbtvdLocator, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for this service.
- SIRequest retrieveSIServices(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener,int originalNetworkId,int transportStreamId, int serviceId, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires the relevant information for this service.
- SIRequest retrieveSITimeFromTDT(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires information about time from the Time Date Table (TDT).
- SIRequest retrieveSITimeFromTOT(short retrieveMode,java.lang.Object appData, SIRetrievalListener listener, short[] someDescriptorTags) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method acquires information about time from the Time Offset Table (TOT).

B.2.2.23 SIExEventInformation class

The SIExEventInformation interface indicates the items of the description and the names of the details for a specific program. The information is acquired from the event descriptors in extended format.

The public methods for the SIExEventInformation class are:

- java.lang.String getDescription()
    - This method acquires the description of the item.
- java.lang.String getName()
    - This method acquires the name of the item.

B.2.2.24 SILackOfResourcesEvent class

The SILackOfResourcesEvent class event is notified when the resources necessary for acquiring requested data in a requisition are not available in the SI. This class extends br.org.sbtvd.si.SIRetrievalEvent

The public method for the SILackOfResourcesEvent class is:

- SILackOfResourcesEvent(java.lang.Object appData, SIRequest request)
    - The constructor of this event.

© ABNT 2010 - All rights reserved

B.2.2.25 SIMonitoringEvent class

The object of the SIMonitoringEvent class is transmitted so the listener object notifies the application of the change in monitored information. This class extends java.util.EventObject.

The public methods for the SIMonitoringEvent class are:

- SIMonitoringEvent(SIDatabase source,byte objectType,int networkId,int bouquetId, int originalNetworkId, int transportStreamId, int broadcasterId, int serviceId, java.util.Date startTime, java.util.Date endTime)
    - Constructor of event object
- int getBouquetID()
    - This method returns the bouquet ID
- int getBroadcasterID()
    - This method returns the broadcaster ID for the broadcaster.
- java.util.Date getEndTime()
    - This method returns the end of programming when event information is modified.
- int getNetworkID()
    - This method returns the network ID for the network.
- int getOriginalNetworkID()
    - This method returns the original network ID for the SIInformation object.
- int getServiceID()
    - This method returns the service ID for the SIInformation object.
- byte getSIInformationType()
    - This method acquires the SIInformation type in the change of information.
- java.lang.Object getSource()
    - This method acquires the SIDatabase instance to be sent to the event.
- java.util.Date getStartTime()
    - This method returns the beginning of programming when event information is modified.
- int getTransportStreamID()
    - This method returns the transport stream ID for the SIInformation object.

© ABNT 2010 - All rights reserved

B.2.2.26 SINotInCacheEvent class

When the request for SI acquisition in the FROM_CACHE_ONLY mode is executed and the data requested does not exist in the cache, this event is notified as a reply. This class extends br.org.sbtvd.si.SIRetrievalEvent.

The public method for the SINotInCacheEvent class is:

SINotInCacheEvent(java.lang.Object appData, SIRequest request)

- Constructor of the event

B.2.2.27 SIObjectNotInTableEvent class

The SIObjectNotInTableEvent class event is notified when the SI table with the information about the location of the requested object is recovered but does not contain the referred to object. This class extends br.org.sbtvd.si.SIRetrievalEvent.

The public method for the SIObjectNotInTableEvent class is:

SIObjectNotInTableEvent(java.lang.Object appData, SIRequest request)

- Constructor of the event

B.2.2.28 SIRequest class

The instance of the SIRequest class object indicates the application acquisition request. The application can cancel the request using this object.

The public methods for the SIRequest class are:

- boolean cancelRequest()
  - This method cancels the acquisition request.
- boolean isAvailableInCache()
  - This method returns the availability of information if it is returned from the stream or cache.

B.2.2.29 SIRequestCancelledEvent class

The SIRequestCancelledEvent class event is launched as a response when a requisition is canceled using the SIRequest.cancelRequest method. This class extends br.org.sbtvd.si.SIRetrievalEvent.

The public method for the SIRequestCancelledEvent class is:

SIRequestCancelledEvent(java.lang.Object appData, SIRequest request)

- Constructor

B.2.2.30 SIRetrievalEvent class

The SIRetrievalEvent class is a basic class for the SI acquisition request conclusion event. Only one event is returned for an SI acquisition request. This class extends java.util.EventObject.

© ABNT 2010 - All rights reserved

The public methods for the SIRetrievalEvent class are:

- SIRetrievalEvent(java.lang.Object appData, SIRequest request)
    - Constructor of the event
- java.lang.Object getAppData()
    - This method returns application data that go through the acquisition method.
- java.lang.Object getSource()
    - This method returns a SIRequest object for the event of origin.

B.2.2.31 SISuccessfulRetrieveEvent class

The SISuccessfulRetrieveEvent class event is sent as a response when the request is terminated normally. The result can be acquired by using the getResult method. This class extends br.org.sbtvd.si.SIRetrievalEvent.

The public methods for the SISuccessfulRetrieveEvent class are:

- SISuccessfulRetrieveEvent(java.lang.Object appData, SIRequest request, SIIterator result)
    - Constructor
- SIIterator getResult()
    - This method returns the SIIterator object that includes requested data.

B.2.2.32 SITableNotFoundEvent class

The SITableFoundEvent class event is sent as a response when the SI table that shall contain the requested information has not been found. One of the reasons may be the fact it is not being transmitted in the stream connected to the SI database. This class extends br.org.sbtvd.si.SIRetrievalEvent class.

The public method for the SITableNotFoundEvent class is:

SITableNotFoundEvent(java.lang.Object appData, SIRequest request)

- Constructor

B.2.2.33 SITableUpdatedEvent class

The SITableUpdateEvent class event is launched as a response when the table, which transmits the information about the SI requisition's target object is updated and the descriptor information in compliance with the old object is not available. In this case, the application shall initially update the SIInformation object. Then the information about the descriptor shall be requested again. This class extends br.org.sbtvd.si.SIRetrievalEvent.

The public method for the SITableUpdatedEvent class is:

- SITableUpdatedEvent(java.lang.Object appData, SIRequest request)
    - Standard Constructor

© ABNT 2010 - All rights reserved

B.2.2.34 SIUtil class

The SIUtil class includes a utility function relevant to the SI.

The public method for the SIUtil class is:

- static java.lang.String convertSIStringToJavaString(byte[] sbtvdSIText, int offset, int length) throws br.org.sbtvd.si.SIIllegalArgumentException
    - This method converts the encrypted text string to the Java object string based on ARIB STD-B10:2008 Part 2, Appendix A. This method inherited from java.lang.Object class.

B.2.2.35 SIException() class

The SIException() class is based on the SI exception hierarchy. This class extends java.lang.Exception.

The public method for the SIException() is:

- SIException()
    - Standard exception constructor.
- SIException(String message)
    - Constructor with parameter (String) to indicate the exception reason.

B.2.2.36 SIIllegalArgumentException() class

The SIIllegalArgumentException() class is launched when more than one improper argument is passed on (for example, out-of-space numerical values). This class extends br.sbtvd.si.SIException.

The public method for the SIIllegalArgumentException() class is:

- SIIllegalArgumentException()
    - Standard exception constructor.
- SIIllegalArgumentException (String message)
    - Constructor with parameter (String) to indicate the exception reason.

B.2.2.37 SIInvalidPeriodException class

The SIInvalidPeriodException class occurs when the specified time extension is improper, for example, initialization time is further delayed. This class extends br.org.sbtvd.si.SIException.

The public methods for the SIInvalidPeriodException class are:

- SIInvalidPeriodException()
    - Standard exception constructor.
- SIInvalidPeriodException(java.lang.String reason)
    - Exception constructor that has a reason to be specified.

© ABNT 2010 - All rights reserved

Annex C
(normative)

API extension specification for tuning – Package br.org.sbtvd.net.tuning

C.1 ChannelManager class

The ChannelManager class specifies an object responsible for the zapping activity (changing) of channels through the network interface (land, cable, satellite, IPTV) that exists in the digital television receiver.

The public methods are:

- static ChannelManager getInstance ()
  - Returns an object (instance) ChannelManager.
- int getNumberofChannels ()
  - Returns the number of channels found in all network interfaces available in the system.
- Channel [] getChannels ()
  - Returns the number of channels found in all network interfaces available in the system.
- void tuneChannel (Channel ch, com.sun.dtv.tuner.TunerListener lis) throws com.sun.dtv.tuner.TuningException
  - Asynchronically tunes in the channel provided by the whole parameter num. This method launches an exception of the com.sun.dtv.tuner.TuningException type in the case of tuning failure.
- void tuneNextChannel (com.sun.dtv.tuner.TunerListener lis) throws com.sun.dtv.tuner.TuningException
  - Asynchronically tunes in the next channel on the table TransportStream generated during the sweep. This method launches an exception of the com.sun.dtv.tuner.TuningException type in the case of tuning failure.
- void tunePreviousChannel (com.sun.dtv.tuner.TunerListener lis) throws com.sun.dtv.tuner.TuningException
  - Asynchronously tunes in the previous channel on the table TransportStream generated during the sweep. This method launches an exception of the com.sun.dtv.tuner.TuningException type in the case of tuning failure.

© ABNT 2010 - All rights reserved

C.2 Channel class

The Channel class represents an object that contains channel data detected during the sweep. For example, from this class' information, it is possible to tune in a channel from its virtual number (remote control key id).

The public methods are:

- com.sun.dtv.transport.TransportStream getTransportStream ()
    - Returns the object that contains the channel.
- String getNetworkName ()
    - Returns the description of the network in which the channel is now.
- String getITransportStreamName ()
    - Returns the description of the transport stream in which the channel is now.
- int getRemoteControlKeyId()
    - Returns the channel’s virtual number.

© ABNT 2010 - All rights reserved

Annex D
(normative)

NCL Bridge API Specification

D.1 General considerations

The br.org.sbtvd.bridge and br.org.sbtvd.bridge.ncl packages have a set of available classes for the bridge between applications written in NCL and Java languages in a Ginga environment. The functions available in the classes described in D.2.1 permit development of Ginga-J applications, including Ginga-NCL applications, as well as Ginga-NCL applications development, including Java Xlets.

This NCL Ginga-J bridge API makes it possible to present and manipulate a NCL document in a Java application, through NCLPlayer class, preserving the original document throughout the entire exhibition process. The classes that bring such functionalities together are provided by the br.org.sbtvd.bridge package.

A NCL document is also capable of including Ginga-J Xlets as one of its media nodes (element <media>). An element <media> with Java code can define anchors (through elements <area>), and attributes (through elements <property>). The transitions applied to Xlet will invoke the methods of the javax.microedition.xlet.Xlet interface (see PBP 1.1:2008), representing the transitions of the machine states. The transitions applied to anchors shall generate NCLAchorEvent class events that encapsulate the transition and the anchor identifier in question. D.2.2 presents the classes that are responsible for Ginga-NCL communication with the Ginga-J environment when an NCL document includes a Ginga-J application. These classes comprise the br.org.sbtvd.bridge.ncl package.

Complementary information can be obtained in ABNT NBR 15606-2:2007, 10.3.4.3 and 11.2.

D.2 NCL bridge API

D.2.1 br.org.sbtvd.net.tuning package

D.2.1.1 NCLPlayer class

The NCLPlayer class is a class that represents a display for an NCL document, a graphic component that extends java.awt.Component. Input events (keys, for example) will be handled by the NCL display (the events are repassed by the Ginga-J environment to the Ginga-NCL) while the NCLPlayer has the interaction focus, among the graphic components being used in the Xlet in question.

The NCLPlayer class' public static constants are:

- static int PLAYING
    - Identification for the document in execution.
- static int PAUSED
    - Identification for the paused document.

© ABNT 2010 - All rights reserved

- static int STOPPED
    - Identification for the stopped document.

The public methods for the NCLPlayer class are:

- NCLPlayer(java.net.URL documentURL)
    - Constructor method for the NCLPlayer, which receives a java.net.URL class instance as a parameter for document locator.
- voidaddNCLPlayerEventListener(NCLPlayerEventListener listener, long nclPlayerEventPlayerMask)
    - This method registers an NCLPlayerEventListener to receive all NCLPlayerEvents distributed by the machine associated to the node linked to the long eventMask value supplied.
- void removeNCLPlayerEventListener(NCLPlayerEventListener listener)
    - Removes an NCLPlayerEventListener from the distributed NCLPlayerEvents reception class.
- NCLPlayerEventListener[] getANCLPlayerEventListeners()
    - Returns a list of all NCLPlayerEventListeners registered in this NCLPlayer. Observes that the listener objects added several times appear only once on the returned list.
- NCLPlayerEventListener[] getANCLPlayerEventListeners(long nclPlayerEventMask)
    - Returns a list of all NCLPlayerEventListeners registered in this NCLPlayer that listens to all types of events indicated in the long eventMask value. The listener objects added several times appear only once on the returned list.
- java.net.URL getDocumentURL()
    - Returns a class object java.net.URL that is the NCL document locator being manipulated by the NCLPlayer object in question.
- java.lang.String getPropertyValue(java.lang.String propertyId)
    - Returns a String instance with the property value defined by the String propertyIdparameter.
- int getStatus()
    - Returns an integer that represents the status of the NCLPlayer object (PLAYING – in execution, PAUSED – paused, STOPPED – stopped).
- void setDocument(java.net.URL documentURL)
    - Defines the NCL document to be manipulated by the NCLPlayer, receiving an object from the java.net.URL class as document identifier. The new execution status for the NCLPlayer shall be defined as stopped (STOPPED).
- boolean startDocument(java.lang.String interfaceId)
    - It begins reproduction of an NCL document starting the presentation at the document interface specified by the String interfaceId instance. Returns true if successful, and false if otherwise.

© ABNT 2010 - All rights reserved

- boolean stopDocument()
    - Stops the presentation of an NCL document. All document events in execution shall obligatorily be stopped. Returns true if successful, and false if otherwise.
- boolean pauseDocument()
    - Pauses the presentation of an NCL document. All document events in execution shall obligatorily be paused. Returns true if successful, and false if otherwise.
- boolean resumeDocument()
    - Resumes presentation of an NCL document. All document events previously paused by the pauseDocument() method shall obligatorily be resumed. Returns true if successful, and false if otherwise.
- NCLEdit getNCLEdit()
    - Returns an NCLEdit class instance that offers NCL document editing functionalities in exhibition time.

D.2.1.2 NCLPlayerEvent class

The NCLPlayerEvent class is the root event for all NCL events generated by a NCL encapsulated application by an instance of the class in question . Event masks defined in this class are used to specify to which types of events an NCLPlayerEventListener shall listen. Additional information can be obtained from ABNT NBR 15606-2:2007, 10.3.4.3.

The NCLPlayerEvent class' public static constants are:

- static int PRESENTATION_START = 1;
    - Event mask for presentation events (presentation type) whose action (action field) was the start of reproduction (start) of a node or anchor.
- static int PRESENTATION_STOP = 2;
    - Event mask for presentation events (presentation type) whose action (action field) was the end of reproduction (stop) of a node or anchor.
- static int PRESENTATION_ABORT = 4;
    - Event mask for presentation events (presentation type) whose action (action field) was the aborting of reproduction (abort) of a node or anchor.
- static int PRESENTATION_PAUSE = 8
    - Event mask for presentation events (presentation type) whose action (action field) was the pausing of reproduction (pause) of a node or anchor.
- static int PRESENTATION_RESUME = 16;
    - Event mask for presentation events (presentation type) whose action (action field) was the resumption of reproduction (resume) of a node or anchor.

© ABNT 2010 - All rights reserved

- static int ATTRIBUTION_SET= 32;
    - Event mask for attribution events (attribution type) whose action (action field) was the definition (set) of a parameter for a node or anchor.
- protected int id
    - Identification of the event.

The public methods for the NCLPlayerEvent class are:

- NCLEvent(Object source, int id, String value)
- Constructor method for NCLEvent, which receives as parameter a reference (source) of the object that originated the event, an integral (id), which identifies the event and an identifier (value) of the node or anchor related to the event.int getID()
    - Returns the type of event.
- String getValue()
    - Returns the identifier of the node or anchor related to the event.

D.2.1.3 NCLPlayerEventListener interface

The Listener interface, which shall be implemented by whoever wants to receive notification of events distributed to NCLEvent class elements objects. It extends the java.util.EventListener interface.

The application interested in monitoring NCL events of an NCLPlayer implements this interface registering as the NCLPlayer using the NCLPlayer.addNCLPlayerEventListener() method. When an event is distributed in the NCLPlayer, the eventDispatched method of this object is executed.

The public method for the NCLPlayerEventListener interface is:

- void NCLPlayerEventDispatched(NCLPlayerEvent event)
    - Method executed when an event is distributed in the NCLPlayer.

D.2.1.4 NCLGingaSettingsNodes class

The NCLGingaSettingsNodes class is a class that represents an NCL node, whose attributes are global variables defined by the document's author or environment variables that can be manipulated by NCL document processing. The complete list of these environment variables is shown in ABNT NBR 15606-2.

The public methods for the NCLGingaSettingsNodes class are:

- NCLGingaSettingsNodes(java.lang.String nodeId)
    - Constructor method for NCLGingaSettingsNodes that receives a single identifying String as a parameter.

© ABNT 2010 - All rights reserved

- String getValue(String value)
    - Returns the variable value according to the description of the environment variable entered as a String. The complete list of environment variables available can be seen in ABNT NBR 15606-2:2007, Table 12.

D.2.1.5 NCLEdit class

The NCLEdit class offers methods to edit a NCL document, which when encapsulated in a NCLPlayer class object; instantiates associated NCLEdit class objects (method getNCLEdit). Editing commands from the NCLEdit instance only alter NCL document presentation (represented by NCLPlayer object) – the original document is preserved throughout the editing process, as specified for the NCL editing commands in ABNT NBR 15606-2:2007.

The public methods for the NCLEdit class are:

- boolean addRegion(java.lang.String regionBaseId, java.lang.String regionId, java.lang.String regionStr)
    - Adds an element <region> in the NCL document, like a member of the region base identified by the String regionBaseId, like a child element identified by the String regionId and defined in the String regionStr. Returns true if successful, and false if otherwise.
- boolean removeRegion(java.lang.String regionId)
    - Removes the element <region> identified by the NCL document's regionId String. Returns true if successful, and false if otherwise.
- boolean addRegionBase(java.lang.String regionBaseStr)
    - Adds the element <regionBase> described in the String regionBaseStr to the element <head> in the NCL document. Returns true if successful, and false if otherwise.
- boolean removeRegionBase(java.lang.String regionBaseId)
    - Removes the element <regionBase> identified by the NCL document's regionBaseId String. Returns true if successful, and false if otherwise.
- boolean addRule(java.lang.String ruleStr)
    - Adds an element <rule> described in the String ruleStr as part of the NCL document <ruleBase> element. Returns true if successful, and false if otherwise.
- boolean removeRule(java.lang.String ruleId)
    - Removes the element <rule> identified in the String ruleId of the NCL document <ruleBase> element. Returns true if successful, and false if otherwise.
- boolean addRuleBase(java.lang.String ruleBaseStr)
    - Adds an element <ruleBase> described in the String ruleBaseStr as part of the NCL document <head> element. Returns true if successful, and false if otherwise.

© ABNT 2010 - All rights reserved

- boolean removeRuleBase(java.lang.String ruleBaseId)
    - Removes the element <ruleBase> identified in the String ruleBaseId of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean addConnector(java.lang.String connectorStr)
    - Adds an element <connector> described in the String connectorStr as part of the NCL document <connectorBase> element. Returns true if successful, and false if otherwise.
- boolean removeConnector(java.lang.String connectorId)
    - Removes the element <connector> identified in the String connectorId of the NCL document <connectorBase> element. Returns true if successful, and false if otherwise.
- boolean addConnectorBase(java.lang.String connectorBaseStr)
    - Adds an element <connectorBase> described in the String connectorBaseStr as part of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean removeConnectorBase(java.lang.String connectorBaseId)
    - Removes the element <connectorBase> identified in the String connectorBaseId of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean addDescriptor(java.lang.String descriptorStr)
    - Adds an element <descriptor> described in the String descriptorStr as a part of the NCL document <descriptorBase> element. Returns true if successful, and false if otherwise.
- boolean removeDescriptor(java.lang.String descriptorId)
    - Removes the element <descriptor> identified in the String descriptorId of the NCL document <descriptorBase> element. Returns true if successful, and false if otherwise.
- boolean addDescriptorSwitch(java.lang.String descriptorSwitchStr)
    - Adds an element <descriptorSwitch> described in the String descriptorSwitchStr as a part of the NCL document <descriptorBase> element. Returns true if successful, and false if otherwise.
- boolean removeDescriptorSwitch(java.lang.String descriptorSwitchId)
    - Removes the element <descriptorSwitch> identified in the String descriptorSwitchId of the NCL document <descriptorBase> element. Returns true if successful, and false if otherwise.
- boolean addDescriptorBase(java.lang.String descriptorBaseStr)
    - Adds an element <descriptorBase> described in the String descriptorStr as a part of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean removeDescriptorBase(java.lang.String descriptorBaseId)
    - Removes the element <descriptorBase> identified in the String descriptorBaseId of the NCL document <head> element. Returns true if successful, and false if otherwise.

© ABNT 2010 - All rights reserved

- boolean addTransition(java.lang.String transitionStr)
    - Adds an element <transition> described in the String transitionStr as a part of the NCL document <transitionBase> element. Returns true if successful, and false if otherwise.
- boolean removeTransition(java.lang.String transitionId)
    - Removes the element <transition> identified in the String transitionId of the NCL document <transitionBase> element. Returns true if successful, and false if otherwise.
- boolean addTransitionBase(java.lang.String transitionBaseStr)
    - Adds an element <transitionBase> described in the String transitionBaseStr as a part of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean removeTransitionBase(java.lang.String transitionBaseId)
    - Removes the element <transitionBase> identified in the String transitionBaseId of the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean addImportBase(java.lang.String docBaseId, java.lang.String importBaseStr)
    - Adds an NCL base element identified in the String docBaseId (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> or <connectorBase>) to the definition of the element <importBase> contained in the String importBaseStr in the NCL document. Returns true if successful, and false if otherwise.
- boolean removeImportBase(java.lang.String docBaseId, java.lang.String importBaseId)
    - Removes the element <importBase> identified in the String importBaseId for an NCL base element identified in the String docBaseId (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> or <connectorBase>) of the NCL document. Returns true if successful, and false if otherwise.
- boolean addImportedDocumentBase(java.lang.String importedDocumentBaseStr)
    - Adds the definition of the element <importedDocumentBase> contained in the String importedDocumentBaseStr to the NCL document <head> element. Returns true if successful, and false if otherwise.
- boolean removeImportedDocumentBase(java.lang.String importedDocumentBaseId)
    - Removes the element <importedDocumentBase> identified by the String importedDocumentBaseId from the NCL document <head> element.
- boolean addImportNCL (java.lang.String importNCLStr)
    - Adds the definition of the element <importNCL> contained in the String importNCLStr to the NCL document <importedDocumentBase> element. Returns true if successful, and false if otherwise. Returns true if successful, and false if otherwise.

© ABNT 2010 - All rights reserved

- boolean removeImportNCL(java.lang.String importNCLId)
    - Removes the element <importNCL> identified by the String importNCLId from the NCL document <importedDocumentBase> element. Returns true if successful, and false if otherwise.
- boolean addNode(java.lang.String compositeId, java.lang.String nodeStr)
    - Adds the definition of an NCL node (<media>, <context> or <switch>) contained in the String nodeStr to an NCL composition node identified in the String compositeId (<body>, <context> or <switch>). Returns true if successful, and false if otherwise.
- boolean removeNode(java.lang.String compositeId, java.lang.String nodeId)
    - Removes the definition of an NCL node (<media>, <context> or <switch>) identified in the String nodeId from an NCL composition node identified in the String compositeId (<body>, <context> or <switch>). Returns true if successful, and false if otherwise.
- boolean addInterface(java.lang.String nodeId, java.lang.String interfaceStr)
    - Adds an NCL interface (element <port>, <area>, <property> or <switchPort>) described in the String interfaceStr to a node (element <media>, <body>, <context> or <switch>) identified by the NCL document nodeId String. Returns true if successful, and false if otherwise.
- boolean removeInterface(java.lang.String nodeId, java.lang.String interfaceId)
    - Removes an NCL interface (element <port>, <area>, <property> or <switchPort>) described in the String interfaceStr from a node (element <media>, <body>, <context> or <switch>) identified by the NCL document nodeId String. Returns true if successful, and false if otherwise.
- boolean addLink(java.lang.String compositeId, java.lang.String linkStr)
    - Adds the definition of an NCL <link> element contained in the linkStr String to an NCL composition node identified in the compositeId (<body>, <context> or <switch>) String. Returns true if successful, and false if otherwise.
- boolean removeLink(java.lang.String compositeId, java.lang.String linkId)
    - Removes the definition of an NCL (<media>, <context> or <switch>) element identified in the String linkId from an NCL composition node identified in the compositeId (<body>, <context> or <switch>) String. Returns true if successful, and false if otherwise.
- boolean setPropertyValue(java.lang.String propertyId, java.land.String value)
    - Attributes the value of the String value to a property identified by propertyId. The instance value for String propertyId shall obligatorily identify an attribute name for an element <property> or an element id attribute <switchPort>. The <property> or <switchPort> shall obligatorily belong to an NCL document (element <body>, <context>, <switch> or <media>) node. Returns true if successful, and false if otherwise.

© ABNT 2010 - All rights reserved

D.2.2 br.org.sbtvd.net.bridge.ncl package

D.2.2.1 NodeManager class

The NodeManager static class has all the methods for NCLEventListeners registration in a way that a Xlet Ginga-J, associated to a media node (element <media>) can receive events (encapsulated in NCLEvent class instances) of the Ginga-NCL environment. Additional information can be obtained from ABNT NBR 15606-2:2007, 10.3.4.3 and 11.2.

The public methods for the NodeManager class are:

- static void addNCLEventListener(NCLEventListener listener)
    - This method registers a NCLEventListener to receive all the NCLEvent instances distributed by the Ginga-NCL environment to the Xlet associated to a media node (element <media>) in a Ginga-NCL. Application.
- static void removeNCLEventListener(NCLEventListener listener)
    - Removes a NCLEventListener previously registered.
- static NCLEventListener[] getNCLEventListeners()
    - Returns a list of all NCLEventListener instances registered with the Node Manager. The listener objects added various times appear only once in the returned list.

D.2.2.2 NCLEvent class

The NCLEvent class extends java.util.EventObject and is the root event class for all events generated by the NCL formatter that manipulates a document including a Xlet Ginga-J. Event masks defined in this class are used to specify which types of events a NCLEventListener shall listen. Additional information can be obtained from ABNT NBR 15606-2:2007, 10.3.4.3 and 11.2.

The NCLEvent class' public static constants are:

- static int PRESENTATION_START
    - Event mask for presentation events (presentation type) whose action (action field) was the start of the reproduction (start) of an anchor defined for the media node (element <media>) that includes Xlet Ginga-J.
- static int PRESENTATION_STOP
    - Event mask for presentation events (presentation type) whose action (action field) was the end of the reproduction (stop) of an anchor defined for the media node (element <media>) that includes Xlet Ginga-J.
- static int PRESENTATION_ABORT
    - Event mask for presentation events (presentation type) whose action (action field) was the abortion of the reproduction (abort) of an anchor defined for the media node (element <media>) that includes Xlet Ginga-J.

© ABNT 2010 - All rights reserved

- static int PRESENTATION_PAUSE
    - Event mask for presentation events (presentation type) whose action (action field) was the pausing of the reproduction (pause) of an anchor defined for the media node (element <media>) that includes Xlet Ginga-J.
- static int PRESENTATION_RESUME
    - Event mask for presentation events (presentation type) whose action (action field) was the resumption of the reproduction (resume) of an anchor defined for the media node (element <media>) that includes Xlet Ginga-J.
- static int ATTRIBUTION_SET
    - Event mask for attribution events (attribution type) whose action (action field) was the definition (set) of a parameter defined for the media node (element <media>) that includes Xlet Ginga-J.
- protected int id
    - Identification of the event.

The public methods for the NCLEvent class are:

- NCLEvent(Object source, int id, String value)
    - Constructor method for NCLEvent, which receives as parameter a reference (source) of the object that originated the event, an integral (id), which identifies the event and an identifier (value) of the node or anchor related to the event.
- int getID()
    - Returns the type of event.
- String getValue()
    - Returns the identifier of the node or anchor related to the event.

D.2.2.3 NCLEventListener interface

The Listener interface shall be implemented by whoever wants to receive notification of events distributed by the NCL formatter manipulating an NCL document that includes an Xlet Ginga-J, which are objects that are NCLEvent class elements. It extends the java.util.EventListener interface

The application you want to monitor the events, generated by the NCL formatter shall implement this interface.

The public method for the NCLEventListener interface is:

- void NCLPlayerEventDispatched(NCLEvent event)
    - Method executed when an NCLEvent event is generated by the NCL formatter.

© ABNT 2010 - All rights reserved

Annex E
(normative)

API specification for graphic plane support - br.org.sbtvd.ui package

E.1 ColorCoding class

The ColorCoding class holds constants to number the different coding models possible for each plane. The possible values correspond to those returned in com.sun.dtv.ui.Plane.getColorCodingModel().

The public static constants found in this class are:

- public static final int ARGB8888 = 1
    - indicates that the color model in the plane is ARGB8888.
- public static final int YUV442
    - indicates that the color model in the plane is YUV442.
- public static final int YUV444 = 3
    - indicates that the color model in the plane is YUV444.
- public static final int ONE_BPP = 4
    - indicates that the color model in the plane is one bit per pixel.

E.2 StillPicture class

The StillPicture class extends the com.sun.dtv.lwuit.Component class. It is the means through which JPEG images are added to the static image plane. This class shall be declared with the final modifier.

The public constructors are the following:

- StillPicture(String path)
    - Constructs a StillPicture object. The passed parameter path shall correspond to the location of a JPEG image in the application file system.

This class' instances do not support focus or animation functionalities.

The following methods inherited from com.sun.dtv.lwuit.Component shall not be invoked directly by the applications:

- void paint(Graphics g)
- void paintBackgrounds(Graphics g)
- void paintComponent(Graphics g)
- void paintComponent(Graphics g, boolean background)

© ABNT 2010 - All rights reserved

E.3 SwitchArea class

The SwitchArea class extends the com.sun.dtv.lwuit.Component class. This class shall be declared with the final modifier.

It is a component that defines a rectangular area for the video/image selection plane. Each rectangular area added through the com.sun.dtv.lwuit.Component#addComponent method() corresponds to an area in which the static image plane will appear over the video plane or vice-versa depending on the component's (com.sun.dtv.lwuit.plaf.Style) style color.

This class' instances do not support focus or animation functionalities.

The following methods inherited from com.sun.dtv.lwuit.Component shall not be invoked directly by the applications:

- void paint(Graphics g)
- void paintBackgrounds(Graphics g)
- void paintComponent(Graphics g)
- void paintComponent(Graphics g, boolean background)

Through the com.sun.dtv.lwuit.plaf.Style associated to each component, it is possible to define whether the video will be displayed on the Still Picture Plane or vice-versa. The instances of com.sun.dtv.lwuit.plaf.Style associated with this component may only contain solid colors (java.awt.Color). The color black, java.awt.Color. BLACK, represents that the video shall be displayed over the Still Picture Plane. By using any other color, the Still Picture Plane content will be displayed in the front of the video.

© ABNT 2010 - All rights reserved

Bibliography

[1] *SOUZA FILHO, Guido Lemos de; LEITE, Luiz Eduardo Cunha; BATISTA, Carlos Eduardo Coelho Freire. Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. In: _____ Journal of the Brazilian Computer Society. No. 4, Vol. 13. p.47-56. ISSN: 0104-6500. Porto Alegre, RS, 2007.*

[2] *SOARES, Luiz Fernando Gomes; RODRIGUES, Rogério Ferreira; MORENO, Márcio Ferreira. Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. In: _____ Journal of the Brazilian Computer Society. No. 4, Vol. 13. p.37-46. ISSN: 0104-6500. Porto Alegre, RS, 2007.*

[3] *Sun Microsystems, Java Digital Television (DTV) API:2008, < http://java.sun.com/javame/technology/javatv/index.jsp>*

[4] *Sun Microsystems, Java TV API:2007, <http://java.sun.com/products/javatv/>*

[5] *Sun Microsystems, Java Media Framework API (JMF), <http://java.sun.com/products/java-media/jmf/index.jsp>*

© ABNT 2010 - All rights reserved