NORMA BRASILEIRA

ABNT NBR 15606-8

Primeira edição
14.10.2011

Válida a partir de
14.11.2011

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 8: Ginga-J — Diretrizes operacionais para a ABNT NBR 15606-4

Digital terrestrial television – Data coding and transmission specification for digital broadcasting
Part 8: Ginga-J – Operations guidleines for the ABNT NBR 15606-4

ICS 35.040; 33.060.20

ISBN 978-85-07-03056-0

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referência
ABNT NBR 15606-8:2011
47 páginas

ABNT
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

Sumário

Página

Prefácio

A Associação Brasileira de Normas Técnicas (ABNT) é o Foro Nacional de Normalização. As Normas Brasileiras, cujo conteúdo é de responsabilidade dos Comitês Brasileiros (ABNT/CB), dos Organismos de Normalização Setorial (ABNT/ONS) e das Comissões de Estudo Especiais (ABNT/CEE), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidade, laboratório e outros).

Os Documentos Técnicos ABNT são elaborados conforme as regras da Diretiva ABNT, Parte 2.

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para a possibilidade de que alguns dos elementos deste documento podem ser objeto de direito de patente. A ABNT não deve ser considerada responsável pela identificação de quaisquer direitos de patentes.

Esta Norma é baseada nos trabalhos do Fórum do Sistema Brasileiro de Televisão Digital Terrestre, conforme estabelecido no Decreto Presidencial nº 5.820, de 29.06.2006.

A ABNT NBR 15606, sob o título geral "*Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital*", tem previsão de conter as seguintes partes:

— Parte 1: Codificação de dados;

— Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações;

— Parte 3: Especificação de transmissão de dados;

— Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais;

— Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações;

— Parte 6: Java DTV 1.3;

— Parte 7: Ginga-NCL – Diretrizes operacionais para as ABNT NBR 15606-2 e ABNT NBR 15606-5;

— Parte 8: Ginga-J - Diretrizes operacionais para a ABNT NBR 15606-4;

— Parte 9: Diretrizes operacionais para a ABNT NBR 15606-1.

O desenvolvimento desta parte da ABNT NBR 15606 foi regido por um acordo que estabeleceu a não utilização de componentes cuja propriedade intelectual estivesse sujeita ao pagamento de *royalties* e à gratuidade das tecnologias desenvolvidas.

Esta parte da ABNT NBR 15606 inclui termos utilizados pela linguagem Java para a descrição de pacotes, classes, métodos, interfaces e variáveis. Estes termos estão grafados em fonte *Arial 10 Itálico*. O emprego desta fonte visa destacar claramente os termos que devem ser usados em todas as implementações desta parte da ABNT NBR 15606 exatamente como aparecem, por serem termos técnicos significativos para tais implementações.

De forma análoga, para os trechos de código fonte em linguagens de programação (Java, XML etc.), emprega-se fonte `Courier New com 9 pts`. O uso deste formato objetiva a diferenciação dos exemplos de código-fonte que podem ser implementados seguindo as regras desta parte da ABNT NBR 15606.

Excepcionalmente, as seções que descrevem pacotes e classes da linguagem, assim como as diversas seções de índices e detalhes, nomes de classes e pacotes, assim como listas de nomes de classes, métodos, exceções, campos etc., são representados em **Arial Negrito** usando formatos padronizados da linguagem Java, a qual

consiste em agregar diversas palavras, sem espaços e iniciando cada uma delas com letra maiúscula, exceto a primeira, que inicia com letra minúscula; no caso de constantes Java, estas são representadas como palavras separadas por '_' e com todas as letras maiúsculas.

O Escopo desta Norma Brasileira em inglês é o seguinte:

Scope

This part of ABNT NBR 15606 details and explains the requirements for the procedural part of the middleware for the Brazilian digital terrestrial television system (SBTVD), providing the operational guidelines for an implementation in accordance with ABNT NBR 15606-4.

Introdução

A especificação Ginga-J é composta por um conjunto de interfaces de programação de aplicativos (API – *Application Programming Interface*) projetadas para suprir todas as funcionalidades necessárias para a implementação de aplicativos para televisão digital, desde a manipulação de dados multimídia até protocolos de acesso.

A especificação Ginga se aplica aos receptores para sistemas de transmissão terrestre de televisão (*over-the-air*). Ginga é destinado a cobrir uma série completa de implementações, incluindo os receptores-decodificadores integrados (IRD), aparelhos de televisão integrados, computadores multimídia e *clusters* locais de aparelhos conectados via redes domésticas (*Home Area Networks* - HAN).

Esta parte da ABNT NBR 15606 é destinada aos desenvolvedores de receptores compatíveis com o sistema brasileiro de televisão digital terrestre (SBTVD) e aos desenvolvedores de aplicativos que utilizam a funcionalidade e API Ginga.

Esta parte da ABNT NBR 15606 tem como objetivo detalhar e explicar os requisitos para a parte procedural do *middleware* para o sistema brasileiro de televisão digital terrestre (SBTVD), fornecendo as diretrizes operacionais para uma implementação de acordo com a ABNT NBR 15606-4.

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 8: Ginga-J - Diretrizes operacionais para a ABNT NBR 15606-4

1 Escopo

Esta parte da ABNT NBR 15606 detalha e explica os requisitos para a parte procedural do *middleware* para o sistema brasileiro de televisão digital terrestre (SBTVD), fornecendo as diretrizes operacionais para uma implementação de acordo com a ABNT NBR 15606-4.

2 Referências normativas

Os documentos relacionados a seguir são indispensáveis à aplicação desta Norma. Para referências datadas, aplicam-se somente as edições citadas. Para referências não datadas, aplicam-se as edições mais recentes do referido documento (incluindo emendas).

ABNT NBR 15603-2, *Televisão digital terrestre – Multiplexação e serviços de informação (SI) – Parte 2: Estrutura de dados e definições da informação básica de SI*

ABNT NBR 15606-1, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 1: Codificação de dados*

ABNT NBR 15606-2:2011, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações*

ABNT NBR 15606-3, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 3: Especificação de transmissão de dados*

ABNT NBR 15606-4:2010, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Ginga-J - Ambiente para a execução de aplicações procedurais*

ABNT NBR 15606-6:2010, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 6: Java DTV 1.3.*

ABNT NBR 15608-3:2011, *Televisão digital terrestre – Guia de operação – Parte 3: Multiplexação e serviço de informação (SI) – Guia para implementação da ABNT NBR 15603:2007*

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

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

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

PBP 1.1:2008, *Personal Basis Profile 1.1 (JSR 217), disponível em http://jcp.org/en/jsr/detail?id=217*

3 Termos e definições

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os seguintes termos e definições das ABNT NBR 15606-1, ABNT NBR 15606-2, ABNT NBR 15606-3, ABNT NBR 15606-4 e ABNT NBR 15606-6 e os seguintes.

3.1
observador
listener
padrão de desenho de arquitetura de *software*

3.2
thread
menor unidade de um fluxo de execução que pode ser escalonada por um sistema operacional

3.3
diálogo modal
janela dependente de um formulário-pai que obriga o usuário a interagir com ela antes de poder retornar à interação com o formulário-pai

3.4
diálogo *modeless*
ao contrario do diálogo modal, o diálogo *modeless* é uma diálogo dependente de um formulário-pai que permite que o usuário a interaja com o formulário-pai mesmo sem ter concluído a interação com o diálogo

3.5
callback
referência a uma porção de código executável (função), passada como argumento para uma outra porção de código

NOTA Isto permite que, posteriormente, a porção de código passada como argumento seja invocada ante o acontecimento de algum evento.

3.6
z-order
mecanismo para definir a ordem na qual objetos gráficos de duas dimensões são superpostos em um plano gráfico de duas dimensões

3.7
foco
indica qual componente gráfico da interface de usuário pode receber os eventos de entrada de teclas.

3.8
***clipping* de vídeo**
funcionalidade que permite definir a região do conteúdo de vídeo a ser exibida na tela

3.9
deprecated
estado atribuído a API e funcionalidades de *software* quando o seu uso deve ser evitado

NOTA Normalmente este estado reflete que tais API ou funcionalidades podem ser descontinuadas em futuras versões.

4 Símbolos e abreviaturas

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os símbolos e abreviaturas das ABNT NBR 15606-1, ABNT NBR 15606-2, ABNT NBR 15606-3, ABNT NBR 15606-4 e ABNT NBR 15606-6 e os seguintes.

AIT *Application Information Table*

AWT *Abstract Window Toolkit*

EDT *Event Dispatch Thread*

EIT *Event Information Table*

FP *Foundation Profile*

HTTP *Hypertext Transfer Protocol*

IP *Internet Protocol*

JAR *Java Archive*

JMF *Java Media Framework*

LWUIT *LightWeight User Interface Toolkit*

NCL *Nested Context Language*

NIT *Network Information Table*

PAT *Program Association Table*

PBP *Personal Basis Profile*

PMT *Program Map Table*

SDT *Service Descriptor Table*

TCP *Transmission Control Protocol*

TTF *True Type Font*

UDP *User Datagram Protocol*

URL *Universal Resource Locator*

5 Arquitetura do *middleware* Ginga

Os requisitos da ABNT NBR 15606-4:2010, Seção 5, devem ser considerados.

6 Formato do conteúdo

Os requisitos da ABNT NBR 15606-4:2010, Seção 6, devem ser considerados.

7 Modelo de aplicação Ginga-J

7.1 Modelo de aplicação

7.1.1 Ciclo de vida da aplicação

A API *Application Management and Lifecycle Control* encontra-se descrita nas seções sobre modelo de aplicação (ver ABNT NBR 15606-4) e aplicações Java DTV (ver ABNT NBR 15606-6).

O gerenciador de aplicações Ginga-J cria e passa para as aplicações Ginga-J instâncias de *XletContext*, implementando as seguintes interfaces:

— *javax.tv.xlet.XletContext*

— *javax.microedition.xlet.XletContext*

Este requisito é importante para atingir compatibilidade mínima entre as especificações JAVATV 1.1:2008 e PBP 1.1:2008 adotadas pelo padrão Ginga-J.

initXlet e startXlet

Por padrão, o gerenciador de aplicações Ginga-J espera por 15 000 ms antes de cancelar uma chamada de métodos *initXlet()* e *startXlet()* de uma aplicação Ginga-J, ou seja, os métodos *initXlet()* e *startXlet()* das aplicações Ginga-J devem ser processados antes do limite-padrão de tempo; caso contrário, pode ocorrer um comportamento inesperado.

pauseXlet

As aplicações Ginga-J devem considerar o fato de que o gerenciador de aplicações pode solicitar que elas suspendam sua execução chamando *pauseXlet()* a qualquer momento. Isso pode acontecer caso a plataforma precise de alguns recursos escassos de volta em seu controle. Quando isso ocorrer, as aplicações devem se comportar o mais adequadamente possível:

— liberando recursos (em especial gráficos, apresentadores de mídia, filtros de seção, controle do sintonizador, manipuladores de arquivos etc.);

— minimizando as exigências de processamento, cancelando *threads* ou colocando-as para dormir;

— evitando qualquer cálculo ou rotinas que possam exigir muitos recursos computacionais;

— evitando solicitações de repintura da tela e operações gráficas em geral;

— evitando qualquer operação JMF diferente de interrupção e liberando apresentadores de monomídias;

— evitando quaisquer alterações na exibição do conteúdo do serviço corrente (operações como início/interrupção/alteração de conteúdos do serviço, alteração de volume, escalonamento de vídeo são altamente desencorajadas e podem ser ignoradas pelo receptor após o gerente de aplicações ter solicitado que a *Xlet* seja suspensa).

É uma decisão de implementação do gerenciador de aplicações suspender as *threads* do *Xlet* após uma chamada a *pauseXlet*(). Neste caso, as *threads* devem ser reiniciadas antes de o método *startXlet*() ser invocado novamente, a fim de retomar a execução da *Xlet*.

Suspender os eventos de notificação (de qualquer natureza - ou seja, teclas, mídias, filtro de seção, etc.) em aplicações Ginga-J em estado suspenso é uma decisão de implementação do gerenciador de aplicações. Nesse caso, tais eventos podem ser descartados. As aplicações devem ter conhecimento dessa condição e, quando retomadas, elas devem verificar quaisquer alterações no estado do receptor que possam ter ocorrido enquanto estava suspensa.

As aplicações no estado *PAUSED* são consideradas como em execução.

destroyXlet

Nas chamadas a *destroyXlet()* as aplicações Ginga-J devem liberar todos seus recursos alocados e cancelar suas *threads* no período de tempo mais curto possível. O gerenciador de aplicações Ginga-J pode, por si só, cancelar de forma incondicional as *threads* da *Xlet* e liberar todos os recursos, caso uma chamada ao método *destroyXlet()* não retorne em menos de 1 000 ms.

O gerenciador de aplicações Ginga-J é responsável por liberar qualquer recurso ou cancelar qualquer *thread* que possa estar ativa após o retorno de *destroyXlet()*. O gerenciador é também responsável por restaurar qualquer estado de exibição de serviço que possa ter sido alterado pela *Xlet* finalizada:

— a reprodução de componentes de vídeo/áudio/closed caption deve ser restaurada, caso tenha sido interrompida pela *Xlet*;

— o posicionamento e o escalonamento do vídeo devem ser restaurados para tela cheia, caso tenham sido alterados pela *Xlet*;

— as configurações de áudio (como controle de volume e idioma) devem ser restauradas, caso tenham sido alteradas pela *Xlet*.

7.1.2 Inicialização de aplicações

Os requisitos da ABNT NBR 15606-4:2010, 7.1.2, devem ser considerados.

7.1.3 Finalização de aplicações

Os requisitos da ABNT NBR 15606-4:2010, 7.1.3, devem ser considerados.

7.1.4 Suporte a múltiplas aplicações Ginga-J

O ambiente de execução Ginga-J permite a execução de aplicações múltiplas em modo concorrente, conforme especificado na ABNT NBR 15606-4:2010, 7.1.4.

De acordo com *Java DTV Application Runtime* (ver ABNT NBR 15606-6), um receptor em conformidade com o Ginga-J deve suportar a execução de no mínimo duas aplicações ao mesmo tempo. Em vez disso, o receptor deve suportar a execução de apenas uma aplicação Ginga-J ao mesmo tempo em que um documento Ginga-NCL.

Em termos práticos, o suporte mínimo de múltiplas aplicações é de duas aplicações executadas em modo concorrente (podendo ser uma aplicação Ginga-NCL e uma Ginga-J ou duas aplicações Ginga-J). *Xlets* embutidas em documentos NCL ou documentos NCL embutidos em *Xlets* não podem ser considerados.

Caso o número máximo de aplicações em execução suportado pelo receptor seja atingido, uma solicitação para execução de uma nova aplicação (ver ABNT NBR 15606-4:2010, 7.1.2) pode ser ignorada pelo receptor sem informar erro.

7.1.5 Recursos disponíveis a aplicações Ginga-J

7.1.5.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 7.1.5, devem ser considerados.

Os recursos disponíveis para o ambiente Ginga-J devem ser compartilhados entre todas as aplicações em execução. Aplicações Ginga-NCL que estiverem sendo executadas devem ser também consideradas, haja visto que em muitos casos os recursos de plataforma são compartilhados entre estes dois ambientes de execução.

A plataforma é responsável por liberar todos os recursos escassos anteriormente reservados, caso a aplicação Ginga-J não os libere corretamente durante seu procedimento de destruição.

A fim de prevenir que as aplicações Ginga-J bloqueiem a reserva de recursos, ao efetuar a chamada do método *ScarceResourceListener#releaseForced(),* uma implementação Ginga-J espera no mínimo 1 000 ms antes de tomar o recurso em seu controle.

7.1.5.2 Tela e gráficos

Ver 8.4.2.

7.1.5.3 Teclas

Ver 8.4.4.

7.1.5.4 Filtros de seção

O receptor deve reservar no mínimo quatro filtros de seção para aplicações Ginga. Esses filtros de seções podem ser utilizados para filtrar:

— carrossel de objetos *DSMCC* (o carrossel de objetos *DSMCC* principal e os carrosséis de objetos *DSMCC* adicionais da aplicação montados utilizando a API *com.sun.dtv.broadcast.BroadcastFileSystem*). Um filtro de seção da plataforma deve ser considerado para cada instância diferente de um *stream* de carrossel de objetos DSMCC que estiver sendo filtrado;

— *DSMCC Stream Event* (utilizando as API *com.sun.dtv.broadcast.BroadcastStream* e *.sun.dtv.broadcast.event.**). Um filtro de seção da plataforma deve ser considerado para cada instância diferente de *DSMCC StreamEvent* que estiver sendo filtrado;

— filtro de seção básico utilizando a API *com.sun.dtv.filtering*. Um filtro de seção da plataforma deve ser considerado para cada instância de *com.sun.dtv.filtering.DataSectionFilter*.

Em caso de uma aplicação de radiodifusão, um dos filtros de seção da plataforma já é alocado para o carrossel de objetos *DSMCC* que a contém, portanto três filtros de seção adicionais ficam disponíveis para serem utilizados:

— na execução de uma segunda aplicação em um carrossel de objetos *DSMCC* diferente;

— para montar um sistema de arquivo de carrossel de objetos *DSMCC* adicional;

— para recepção e análise de *stream events*; ou

— para filtragem de seções em geral.

O ambiente de execução Ginga-NCL pode ainda utilizar filtros de seção de plataforma enquanto executa documentos NCL (em geral, um para o carrossel de objetos *DSMCC* e, eventualmente, um para *stream events*). Nesse caso, o número de filtros de seção disponível para aplicações em execução concorrente Ginga-J estaria reduzido.

7.1.5.5 Apresentadores de mídia

Ver 8.6. O número de apresentadores de mídia disponíveis para aplicações Ginga é limitado. A implementação JMF provê apresentadores solicitados às aplicações enquanto estiverem disponíveis.

As instâncias de apresentadores de mídia não podem ser compartilhadas entre diferentes aplicações.

7.1.6 Controlando aplicações

Os requisitos da ABNT NBR 15606-4:2010, 7.1.6, devem ser considerados.

7.1.7 Comunicação entre aplicações

Os requisitos da ABNT NBR 15606-4:2010, 7.1.7, devem ser considerados.

7.1.8 Propriedades de ambiente

Os requisitos da ABNT NBR 15606-4:2010, 7.1.8, devem ser considerados.

7.1.9 Códigos de controle de aplicação

Os requisitos da ABNT NBR 15606-4:2010, 7.1.9, devem ser considerados.

7.2 Cache e armazenamento de aplicação

Os requisitos da ABNT NBR 15606-4:2010, 7.2, devem ser considerados.

Essa funcionalidade pode não estar presente, haja visto que um dado receptor pode não ter memória suficiente disponível para armazenamento. Nesse caso, as aplicações sinalizadas com o código de controle STORE devem ser tratadas como se estivessem sinalizadas com o código de controle PRESENT. O mesmo deve ser considerado para aplicações sinalizadas com o código de controle UNBOUND (conforme estabelecido na ABNT NBR 15606-4:2010, 7.1.9).

7.3 Transmissão de aplicações

7.3.1 Regras de sinalização

Os requisitos da ABNT NBR 15606-4:2010, 7.3.1, devem ser considerados.

7.3.2 Empacotamento de aplicações

Os requisitos da ABNT NBR 15606-42010, 7.3.2, devem ser considerados.

A fim de minimizar a sobrecarga de processamento e acelerar o acesso a arquivos, é altamente recomendado não utilizar pacotes JAR, já que isso exige o acesso sequencial aos arquivos neles contidos.

Diretório-base

Emissoras podem definir um diretório-base onde classes de Java (*bytecodes*) podem ser encontradas. Com isso, o diretório base pode ser passado para o gerenciador de aplicações Ginga-J utilizando o descritor AIT *ginga_application_location_descriptor* (ver ABNT NBR 15606-3), onde o campo *base_directory* representa o diretório onde as classes de aplicações podem ser encontradas.

Classpaths adicionais

Assim como aplicações Java em modo *stand-alone* podem definir *classpaths* adicionais utilizando o parâmetro de comando *-classpath* (ver JVM:1997), *classpaths* adicionais podem ser transmitidos ao gerenciador de aplicações durante a criação de um *Xlet.*

Dessa forma, *classpaths* adicionais podem ser transmitidos ao gerenciador de aplicações Java utilizando-se o descritor AIT*ginga_application_location_descriptor* (ver ABNT NBR 15606-3), onde o campo

classpath_extension representa um conjunto de arquivos JAR adicionais (ver JAR:2009) e diretórios (separados por vírgulas), onde as classes da aplicação podem ser encontradas.

O uso dos arquivos JAR é restrito a caminhos acessíveis através do canal interativo, utilizando o protocolo HTTP.

Carregando classes

O gerenciador de aplicações Ginga-J deve procurar por classes na seguinte ordem:

— classes de sistema (CDC, PBP, FP e Ginga-J);

— diretório-base;

— caminhos de classes adicionais, na ordem listada.

O escopo dessas definições de caminhos de classes é exclusivo da aplicação.

Classes de aplicações e nomeação de pacotes

Os *namespaces* dos pacotes definidos pelo padrão Ginga-J (*br.org.sbtvd.**), JavaDTV (*com.sun.dtv.**), *java.**, *javax.**, *sun.** e *com.sun.** são reservados e não podem ser utilizados pelas aplicações Ginga-J.

7.3.3 Autenticação de aplicação e concessão de permissões

A fim de garantir retrointeroperabilidade com futuras definições de segurança no referente à assinatura das aplicações e às permissões que devem ser outorgadas, as seguintes regras devem ser consideradas:

— para aplicações assinadas:

  — as aplicações podem ser transmitidas com informações de assinatura, permissões e certificados. Essas informações são empacotadas como metainformações;

  — os receptores sem suporte de autenticação podem ignorar as metainformações de assinatura, permissões e certificados. A aplicação não é autenticada, porém é executada com permissões conforme padrão definido pela implementação;

  — os receptores com suporte de autenticação interpretam as metainformações de assinatura, permissões e certificados, e tentam autenticar a aplicação. Se a autenticação tiver êxito, a aplicação é executada com as permissões solicitadas. Caso contrário, a aplicação é executada com permissões similares às concedidas às aplicações não assinadas;

— para aplicações não assinadas:

  — as aplicações são transmitidas sem quaisquer metainformações de assinatura, permissões e certificados;

  — os receptores sem suporte de autenticação podem executar a aplicação com permissões conforme padrão definido pela implementação;

  — os receptores com suporte à autenticação executam a aplicação com permissões concedidas a aplicações não assinadas.

As aplicações Ginga podem solicitar permissões para uso de recursos escassos disponíveis na plataforma, utilizando procedimentos definidos em “Política de Segurança Recomendada Para Java DTV” (ver ABNT NBR 15606-6). No entanto, cabe ao receptor a decisão final de outorgar permissão à aplicação Ginga para uso do recurso em questão.

7.3.4 Transmitindo a mesma aplicação em diferentes serviços

Os requisitos da ABNT NBR 15606-4:2010, 7.3.4, devem ser considerados. Os seguintes esclarecimentos devem ser considerados, presumindo que três serviços estejam transmitindo a mesma aplicação de acordo com a Tabela 1:

— transição do serviço A para o serviço B: a aplicação deve continuar a execução sem ser destruída;

— transição do serviço A para o serviço C: a aplicação deve continuar a execução sem ser destruída;

— transição do serviço C para o serviço A: a aplicação deve ser destruída e reiniciada;

— transição do serviço C para o serviço B: a aplicação deve ser destruída.

Tabela 1 — Cenário de transição de aplicação

Serviço/parâmetros de transmissão	Serviço A	Serviço B	Serviço C
ID de Aplicação (<org_id:app_id>)	50:01	50:01	50:01
Protocolo de transporte	*Object carousel* 0x01	*Object carousel* 0x01	*Object carousel* 0x01
Flag service bound	0	1	1
Código de controle de aplicação	AUTOSTART	PRESENT	AUTOSTART

7.3.5 *Download* de aplicação através do canal interativo

Os requisitos da ABNT NBR 15606-4:2010, 7.3.5, devem ser considerados.

Pacotes JAR com compressão não podem ser utilizados, já que seu suporte é opcional.

8 Plataforma Ginga-J

8.1 Plataforma Java

Os requisitos da ABNT NBR 15606-4:2010, 8.1, devem ser considerados.

8.2 Considerações básicas para a plataforma

8.2.1 Ambiente de execução

Os requisitos da ABNT NBR 15606-4:2010, 8.2.1, devem ser considerados.

8.2.2 Hierarquia de pacotes e classes

Os requisitos da ABNT NBR 15606-4:2010, 8.2.2, devem ser considerados.

8.2.3 Notificação de eventos

Os requisitos da ABNT NBR 15606-4:2010, 8.2.3, devem ser considerados.

Como uma diretriz geral, as aplicações devem cancelar o registro de todos os observadores (*listeners*) registrados quando tais aplicações forem destruídas (por exemplo, na implementação do método *destroyXlet*()).

8.2.4 Codificação de texto

Os requisitos da ABNT NBR 15606-4:2010, 8.2.4, devem ser considerados.

8.2.5 Ciclo de vida da aplicação

Os requisitos da ABNT NBR 15606-4:2010, 8.2.5, devem ser considerados.

8.3 Infraestrutura comum

Os requisitos da ABNT NBR 15606-4:2010, 8.3, devem ser considerados.

8.4 Apresentação de gráficos e tratamento de eventos

8.4.1 LWUIT

Os requisitos da ABNT NBR 15606-4:2010, 8.4.1, devem ser considerados.

8.4.2 Interface gráfica de usuário

8.4.2.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 8.4.2, devem ser considerados.

A fim de harmonizar a apresentação gráfica das aplicações Ginga-J executadas em diferentes plataformas, o comportamento esperado com relação às API gráficas AWT e LWUIT é descrito em 8.4.2.2 a 8.4.2.10.

8.4.2.2 Procedimentos de pintura do LWUIT

Durante uma chamada ao método *com.sun.dtv.lwuit.Form#show()*, se o formulário não for previamente adicionado à instância *com.sun.dtv.ui.DTVContainer* associada ao plano de gráficos, ele é implicitamente adicionado pelo *middleware*, e o plano supracitado deve ser, também em forma implícita, reservado para a aplicação.

Como é usual com componentes AWT, as chamadas ao método *java.awt.Component#repaint()* podem ser colapsadas e atendidas por grupos. O algoritmo de colapso de solicitações de redesenho, para acelerar os processos de atualização da tela, depende da configuração da plataforma. Assim, é permitido que a implementação LWUIT chame o método *paint()* em seus componentes menos vezes do que o número de solicitações *repaint()*.

As aplicações Ginga devem considerar que o colapso para as solicitações de redesenho é sempre plausível de acontecer, em especial quando se utilizam animações LWUIT. Mesmo que o método *animate()* seja chamado diversas vezes para o mesmo componente e este tenha sempre retornado *true*, o método *paint()* pode ser chamado apenas uma vez. Nesse caso, o último estado ou o quadro do componente animado é atualizado na tela.

Como orientação geral para o desenvolvimento de aplicações, as atualizações frequentes da tela cheia (mais de uma atualização de tela cheia por segundo) devem ser evitadas.

8.4.2.3 Comportamento de classe *com.sun.dtv.lwuit.MediaComponent* do LWUIT

A instância do tipo *javax.media.Player* passada como argumento no construtor da classe *com.sun.dtv.lwuit.MediaComponent* deve estar no estado *Realized*. Caso contrário, um *javax.media.NotRealizedError* é lançado.

As instâncias *com.sun.dtv.lwuit.MediaComponent* devem ser renderizadas sobre todos os conteúdos gráficos visíveis no formulário.

8.4.2.4 Utilizando componentes legados AWT na hierarquia LWUIT

As aplicações Ginga-J podem inserir instâncias da classe *java.awt.Component* na hierarquia LWUIT, utilizando a classe *com.sun.dtv.lwuit.AWTComponent*. Sendo uma classe utilitária de encapsulamento e uma subclasse de *com.sun.dtv.lwuit.Component*, o *com.sun.dtv.lwuit.AWTComponent* deve suportar a sobreposição com componentes LWUIT.

As solicitações de redesenho (por exemplo, chamadas ao método *repaint()*) feitas diretamente no componente AWT encapsulado têm o mesmo comportamento que as solicitações de componentes LWUIT.

Qualquer instância de *com.sun.dtv.lwuit.AWTComponent* pode ser registrada como uma animação LWUIT utilizando-se *com.sun.dtv.lwuit.Form#registerAnimation()*. Da mesma forma, qualquer outra instância *com.sun.dtv.lwuit.Component* pode ainda ser registrada como uma animação LWUIT.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
Form form = new Form("Ginga Form");

java.awt.Component legacyComponent = new java.awt.Component(){
    int count=0;
    public void paint(Graphics g) {
        g.drawString("Ginga", count++, 0);
    }
};
legacyComponent.setBounds(0,0,100,100);
legacyComponent.setVisible(true);
legacyComponent.validate();

AWTComponent awtComponent = new AWTComponent(legacyComponent);

form.addComponent(awtComponent);

form.show();

form.registerAnimation(awtComponent);
```

8.4.2.5 Componentes AWT x componentes LWUIT

As aplicações Ginga-J devem evitar utilizar componentes AWT e componentes gráficos LWUIT ao mesmo tempo, porque não há garantia de um comportamento esperado exato nessa situação.

Se necessário, as aplicações Ginga-J que utilizam a *framework* LWUIT devem utilizar a classe *com.sun.dtv.lwuit.AWTComponent*. Caso contrário, as aplicações Ginga-J com essas exigências devem ser projetadas para utilizar apenas AWT.

8.4.2.6 Fontes LWUIT

Os métodos de criação de fontes *com.sun.dtv.lwuit.Font#createSystemFont(int, int, int)* e *com.sun.dtv.lwuit.Font#createSystemFont(String, int, int)* recebem como terceiro argumento um valor inteiro utilizado para configurar o tamanho da fonte do sistema. Esse argumento pode ser preenchido com uma das constantes *SIZE_** definidas na classe *com.sun.dtv.lwuit.Font*.

Essas constantes devem ser estabelecidas com representações inteiras, válidas e reais, uma vez que as aplicações Ginga-J podem também utilizar valores inteiros arbitrários no terceiro parâmetro, além das constantes *SIZE_SMALL*, *SIZE_MEDIUM* e *SIZE_LARGE*.

Como uma sugestão de implementação, as constantes *com.sun.dtv.lwuit.Font SIZE_** podem ser definidas com os valores inteiros a seguir:

— `public static final int SIZE_LARGE = 24;`

— `public static final int SIZE_MEDIUM = 20;`

— `public static final int SIZE_SMALL = 16.`

8.4.2.7 Formulários LWUIT

Algumas convenções devem ser definidas com relação ao seguinte trecho da ABNT NBR 15606-6:2010, 24.15.1, sobre a classe *com.sun.dtv.lwuit.Form*:

> *"componente de nível mais alto que serve como a entidade mais visível no UI (diretamente acoplado no DTVContainer, esse Container manuseia os menus e título enquanto coloca conteúdo entre eles."*

Só é suportada a execução e exibição de apenas uma instância de *com.sun.dtv.lwuit.Form* por vez.

Após uma instância de *com.sun.dtv.lwuit.Form* ser apresentada (chamando o método *com.sun.dtv.lwuit.Form#show()*), se outra instância de *com.sun.dtv.lwuit.Form* na mesma aplicação invocar *show()*, a instância do *com.sun.dtv.lwuit.Form* atualmente exibida não é mais visível.

Uma solicitação *com.sun.dtv.lwuit.Form#show()* torna vísivel implicitamente a instância do *com.sun.dtv.ui.DTVContainer* associada ao plano gráfico antes de apresentar o *com.sun.dtv.lwuit.Form*.

Após chamar *com.sun.dtv.lwuit.Form#show()*, não é necessário chamar *com.sun.dtv.lwuit.Form#repaint()*. Por padrão, as instâncias de *com.sun.dtv.lwuit.Form* possuem a mesma resolução da instância do *com.sun.dtv.ui.DTVContainer* associada ao plano gráfico (por exemplo, 1280x720). Isso significa que chamar *com.sun.dtv.lwuit.*Form#repaint() resulta na repintura da tela inteira. As aplicações Ginga não podem fazer isso para evitar problemas de desempenho durante a atualização de tela.

Os componentes LWUIT não são repintados implicitamente após uma chamada de seus métodos *setVisible()*.

Diálogos

Um caso especial de *Forms do LWUIT* são os diálogos. Diálogos são por padrão de tipo *modal*.

As aplicações devem ter cuidado especial quando utilizarem diálogos *modal*, já que em algumas situações podem comprometer a integridade e estabilidade do sistemas se não forem utilizados corretamente. Devido aos diálogos *modal* bloquearem a *thread* que solicita a sua exibição por um longo tempo, as aplicações não podem invocar tais diálogos desde:

— a EDT;

— as *threads* que invocam os métodos da interface *javax.microedition.xlet.Xlet*; ou

— quaisquer outra função de *callback* ou método de interface de *listener* registrado com quaisquer classe do sistema.

Ao cometer este erro podem ser ocasionados resultados ou efeitos não esperados ou indesejados pelas aplicações. Por este motivo, é fortemente recomendado que o uso de diálogos *modal* seja feito com extremo cuidado quando não evitado. Adicionalmente, é recomendado que as implementações tratem os diálogos *modal* como *modeless*, a fim de garantir a robustez do sistema.

8.4.2.8 Instalação de fontes

A instalação de fontes é somente possível utilizando-se a classe *com.sun.dtv.ui.DownloadableFont* estabelecida na ABNT NBR 15606-6.

As aplicações Ginga-J devem fornecer arquivos de fonte com um nome da família de fonte válido aos métodos de instalação definidos na classe *com.sun.dtv.ui.DownloadableFont*. Um arquivo de fonte com um nome de família de fonte nulo ou vazio não pode ser instalado.

O nome da família de fontes funciona como o nome lógico da fonte. Ele pode ser usado por aplicações Ginga-J para criar instâncias de fonte, conforme a seguir:

— utilizando o método estático *com.sun.dtv.lwuit.Font#getFont(String)* para criar instâncias de fonte LWUIT;

— utilizando o construtor *java.awt.Font#Font(String, int, int)*, passando o nome da fonte supracitado no primeiro parâmetro, a fim de criar uma instância de Fonte AWT; ou

— utilizando o método estático *java.awt.Font#getFont(String)* para criar instâncias de fonte AWT.

As aplicações Ginga-J são incentivadas a usar apenas o formato de fonte *True Type Format*(TTF).

Depois de destruir uma aplicação Ginga-J, o gerenciador de aplicações do Ginga-J desinstala automaticamente todas as fontes instaladas pela aplicação.

8.4.2.9 Animações LWUIT

Todas as clases derivadas da classe *com.sun.dtv.lwuit.Component* devem implementer as interfaces *com.sun.dtv.lwuit.Animation* e *com.sun.dtv.ui.Animated*.

— *com.sun.dtv.ui.Animated*: esta interface permite controlar a animação (estado: iniciada/detida; *delay*; modo de repetição) – o estado inicial é detido.

— *com.sun.dtv.lwuit.Animation*: é a interface que permite ao *com.sun.dtv.lwuit.Form* gerenciar a visualização do componente animado (a EDT pode verificar a necessidade de atualizar a animação utilizando o método *animate()* desta interface e invocar ao método *paint()* para desenhar o quadro).

As aplicações devem utilizar estas interfaces na seguinte ordem:

1) o componente animado deve ser adicionado a uma instância ativa de *com.sun.dtv.lwuit.Form*;

2) a interface *com.sun.dtv.lwuit.animations.Animation* do componente animado deve ser registrada na instância do *com.sun.dtv.lwuit.Form* ativa, utilizando o método *com.sun.dtv.lwuit.Form#registerAnimation()*. A partir deste momento, a EDT começará a verificar a necessidade de atualizar a animação do componente (invocando o método do componente *com.sun.dtv.lwuit.animations.Animation#animate()*) e a invocar o método *com.sun.dtv.lwuit.animations.Animation#paint()* sucessivamente para criar o efeito de animação;

3) a animação deve ser iniciada pela aplicação, utilizando o método *com.sun.dtv.ui.Animated#start()* implementado pelo componente animado – o método *com.sun.dtv.lwuit.animations.Animation#animate()* somente deve retornar *true* quando a animação estiver iniciada.

Os componentes *com.sun.dtv.lwuit.List* e *com.sun.dtv.lwuit.ComboBox* do LWUIT implementam a interface *com.sun.dtv.ui.VisualOnlyComponent*. Apesar disto, a funcionalidade do método herdado *setAnimatedContent()* e o comportamento das interfaces *com.sun.dtv.lwuit.animations.Animation* e *com.sun.dtv.ui.Animated* não são consistentes com resultado esperado para estes dois componentes. Por este motivo, é fortemente recomendado que as aplicações não utilizem o mecanismo de animações do LWUIT neste tipo de objetos. Alternativamente, as aplicações podem utilizar a classe associada *com.sun.dtv.lwuit.list.DefaultListCellRenderer* para animar estes componentes.

8.4.2.10 Transições LWUIT

As transições LWUIT, definidas pelas classes *com.sun.dtv.lwuit.animations.Transitions* e *com.sun.dtv.lwuit.animations.CommonTransitions* devem ser completamente suportadas.

Em alguns casos extremos, como transições que abarcam a tela inteira, pode acontecer que a plataforma não tenha recursos suficientes (por exemplo, devido a restrições de processador ou memória) para exibir o efeito animado de transição de forma completa. Em tais casos, é esperado que a plataforma, no mínimo, exiba, o primeiro e o último quadro do efeito de transição. Nestas situações a melhor prática para as aplicações é evitar o uso de transições que ocupem a tela inteira.

8.4.3 Tratamento dos planos da tela

8.4.3.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.1, devem ser considerados.

8.4.3.2 Plano de texto e plano gráfico

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.2, devem ser considerados.

8.4.3.3 Plano de seleção de imagens estáticas e vídeo

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.3, devem ser considerados.

8.4.3.4 Plano de imagens estáticas

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.4, devem ser considerados.

O uso da classe *br.org.sbtvd.ui.StillPicture* é o único modo de adicionar imagens ao plano de imagens estáticas. As aplicações Ginga não podem chamar o método *displayImage()* na instância da classe *com.sun.dtv.ui.PlaneSetup* associada ao plano de imagens estáticas. Neste caso, as chamadas a este método não podem ter qualquer efeito.

8.4.3.5 Plano de vídeo

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.5, devem ser considerados.

8.4.3.6 Composição dos planos da tela

Os requisitos da ABNT NBR 15606-4:2010, 8.4.3.6, devem ser considerados.

8.4.3.7 Considerações adicionais

As condições a seguir definem o comportamento esperado para as instâncias dos planos de tela e como devem ser exibidas as suas respectivas instâncias da classe *com.sun.dtv.ui.DTVContainer*:

Plano gráfico:

— o tamanho inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associado ao plano gráfico é a tela cheia;

— a cor de fundo inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano gráfico é completamente transparente - ARGB (0, 0, 0, 0);

— as solicitações de redimensionamento chamando o método *com.sun.dtv.ui.DTVContainer#resize(DTVContainerPattern)* da instância de *DTVContainer* associada ao plano de gráficos devem ser tratadas adequadamente. A instância de *com.sun.dtv.ui.DTVContainerPattern* retornada define a configuração mais adequada ao parâmetro *preferences* passado. A preferência *DTV_CONTAINER_PLANE_SETUP* deve ser ignorada no redimensionamento. A preferência *DTV_CONTAINER_DIMENSION* deve ser uma instância de *com.sun.dtv.lwuit.geom.Dimension.* A preferência *DTV_CONTAINER_LOCATION* deve ser uma instância de *com.sun.dtv.lwuit.geom.Point.* As aplicações não podem utilizar qualquer outro método da classe *com.sun.dtv.ui.DTVContainer*, herdado de *com.sun.dtv.lwuit.Component*, que possa alterar seus limites, pois seus resultados são indefinidos. Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
Plane graphicPlane=null;

Plane[] planes = Screen.getCurrentScreen().getAllPlanes();

for(int i = 0; i < planes.length;i++){

    if ( planes[i].getID() == "GraphicPlane") {
        graphicPlane=planes[i];
        break;
    }
}

DTVContainer graphicDTVContainer=DTVContainer.getDTVContainer(graphicPlane);

DTVContainerPattern pattern = new DTVContainerPattern();

pattern.setPreference(DTVContainerPattern.DTV_CONTAINER_DIMENSION,
                  new Dimension(100,100), PlaneSetupPattern.REQUIRED);
pattern.setPreference(DTVContainerPattern.DTV_CONTAINER_LOCATION,
                  new Point(0,0), PlaneSetupPattern.REQUIRED);

pattern = graphicDTVContainer.resize(pattern);
// The Location will be the closest possible value to 0,0.
// The Dimension will be the closest possible value to 100,100.
```

As aplicações Ginga-J podem ainda redimensionar a exibição do plano de gráficos, chamando o método *java.awt.Container#setBounds()* na instância de *java.awt.Container* raiz do Xlet retornada pelo método *javax.microedition.xlet.XletContext#getContainer()*. As novas localização e dimensão são as que melhor se ajustarem aos valores indicados. Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
try {
    java.awt.Container rootContainer = context.getContainer();
} catch (UnavailableContainerException e) {
    e.printStackTrace();
}
rootContainer.setBounds(0, 0, 100, 100);

// The Location will be the closest possible value to 0,0.
// The Dimension will be the closest possible value to 100,100.
```

Plano de imagens estáticas:

— o tamanho inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas é a tela cheia;

— a cor de fundo inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas é preta - ARGB (255, 0, 0, 0);

— as solicitações de redimensionamento chamando o método *com.sun.dtv.ui.DTVContainer#resize()* da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas devem ser ignoradas.

Plano de seleção de imagens estáticas e vídeo:

— o tamanho inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de seleção de imagens estáticas e vídeo é a tela cheia;

— a cor de fundo inicial da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de seleção de imagens estáticas e vídeo é preta - ARGB (255, 0, 0, 0). Isto implica que o plano de vídeo esteja integralmente selecionado por padrão;

— as solicitações de redimensionamento chamando o método com.sun.dtv.ui.DTVContainer*#resize()* da instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de seleção de imagens estáticas e vídeo devem ser ignoradas.

Plano de vídeo:

— não pode haver nenhuma instância de *com.sun.dtv.ui.DTVContainer* associada ao plano de vídeo.

8.4.3.8 Reserva dos planos da tela

Considerando que o Ginga-J é um ambiente de múltiplas aplicações, é possível ter duas ou mais aplicações Ginga-J solicitando acesso simultâneo aos planos da tela.

Plano de imagens estáticas e plano de seleção de imagens estáticas e vídeo

As seguintes definições devem ser consideradas pelas aplicações Ginga-J ao utilizar uma instância *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas ou ao plano de seleção de imagens estáticas e vídeo:

— é opcional reservar o plano relacionado à instância de *com.sun.dtv.ui.DTVContainer*, caso haja somente uma aplicação Ginga-J em execução;

— a visibilidade dos componentes adicionados nesses containers só é garantida após a reserva do respectivo plano. Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
Plane stillPlane=null;

   Plane[] planes = Screen.getCurrentScreen().getAllPlanes();

   for(int i = 0; i < planes.length; i++){

       if ( planes[i].getID() == "StillPlane") {
           stillPlane=planes[i];
           break;
       }
   }

   DTVContainer stillDTVContainer=DTVContainer.getDTVContainer(stillPlane);

   ScarceResourceListener listener= new ScarceResourceListener() {
       public void released(ScarceResource resource) {}
       public boolean releaseRequested(ScarceResource resource) {return false;}
       public void releaseForced(ScarceResource resource) {};
   };
   stillPlane.reserve(true, 1000, listener);

   stillDTVContainer.addComponent(new StillPicture("background.jpg"));
```

— após ser liberado, os componentes visíveis previamente adicionados a um *com.sun.dtv.ui.DTVContainer* associado ao plano de imagens estáticas ou ao plano de seleção de imagens estáticas o vídeo não são mais visíveis;

— as aplicações Ginga-J devem estar cientes de que outras aplicações podem solicitar a reserva de plano a qualquer momento.

Plano gráfico

As seguintes diretrizes devem ser consideradas pelas aplicações Ginga-J ao utilizar a instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de gráficos:

— é opcional reservar o *com.sun.dtv.ui.DTVContainer*;

— se uma aplicação Ginga-J tentar exibir qualquer componente no plano gráfico sem requerer a instância da classe *com.sun.dtv.ui.DTVContainer* associada a este plano (utilizando LWUIT ou AWT), uma instância da classe *com.sun.dtv.ui.DTVContainer* deve ser implicitamente adicionada na hierarquia, sendo o *container* de nível superior;

— cada aplicação Ginga-J tem uma instância diferente e única da classe *com.sun.dtv.ui.DTVContainer* para o plano gráfico. Sempre que mais de uma aplicação estiver presente, as diferentes instâncias da classe *com.sun.dtv.ui.DTVContainer*s são sobrepostas uma sobre a outra no plano gráfico. A ordem de composição inicial é indefinida, porém as aplicações podem ser trazidas para a frente chamando o método *setToFront()*;

— a plataforma é responsável pela composição das múltiplas instâncias da classe *com.sun.dtv.ui.DTVContainer* relacionadas ao plano gráfico (de acordo com as regras compostas estabelecidas na ABNT NBR 15606-4);

— a quantidade de memória de vídeo disponível na plataforma limita a renderização de múltiplas instâncias da classe *com.sun.dtv.ui.DTVContainers* no plano gráfico. As aplicações Ginga-J devem estar cientes dessa limitação.

8.4.3.9 Gerenciamento de foco e eventos de entrada para os planos da tela

Algumas considerações devem ser definidas com relação ao seguinte trecho da documentação de classe *com.sun.dtv.ui.DTVContainer* da ABNT NBR 15606-6:2010, 50.12.1:

> *"Sendo um Component, um DTVContainer é capaz de receber eventos de input e manipulá-los com o uso de handlers de evento anexos. A visibilidade de um DTVContainer não é garantia de que também receberá eventos. Por isso, o aplicativo associado deve garantir que o DTVContainer receba foco de input. O DTVContainer então passará o foco de input para um dos componentes contidos de uma maneira dependente da implementação. Se o aplicativo inteiro ganhar ou perder o foco de input (e ao mesmo tempo também o DTVContainer associado), isso deve ser indicado pelo sistema enviando um evento WINDOW_ACTIVATED ou WINDOW_DEACTIVATED ao DTVContainer, respectivamente. "*

A instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas não possui nenhum suporte de foco. Assim, uma chamada ao seu método *com.sun.dtv.ui.DTVContainer#requestFocus()* não pode provocar qualquer alteração visual.

A instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de seleção de imagens estáticas e vídeo não possui suporte de foco. Assim, uma chamada ao seu método *com.sun.dtv.ui.DTVContainer#requestFocus()* não pode resultar em qualquer alteração visual.

Uma aplicação Ginga deve chamar o método *com.sun.dtv.ui.DTVContainer#requestFocus()* na instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de gráficos quando desejar obter foco. Além disso, esse método faz com que todo o foco seja fornecido à aplicação Ginga-J do chamador. Em um cenário de múltiplas aplicações, o último Componente detentor do foco (independentemente de ser um *com.sun.dtv.lwuit.Component* ou um java.awt.Component) antes da chamada acima, perde o foco para eventos de entrada.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
Plane graphicPlane=null;
Plane[] planes = Screen.getCurrentScreen().getAllPlanes();

for(int i = 0; i < planes.length;i++){

    if ( planes[i].getID() == "GraphicPlane") {
        graphicPlane=planes[i];
        break;
    }
}

DTVContainer graphicDTVContainer=DTVContainer.getDTVContainer(graphicPlane);
graphicDTVContainer.requestFocus();
```

8.4.3.10 Suporte para *z-order* de planos

Algumas considerações devem ser definidas com relação ao seguinte excerto da documentação de método *com.sun.dtv.ui.DTVContainer#setToFront()* da ABNT NBR 15606-6:2010, 50.12.3:

> *"Traz o DTVContainer para a frente"*

Se a instância de *com.sun.dtv.ui.DTVContainer* não for visível, a chamada para esse método não tem efeito.

A instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de imagens estáticas não possui suporte para *z-order*. Assim, as chamadas a seu método *setToFront()* não podem provocar qualquer alteração visual.

A instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de seleção de imagens estáticas e vídeo não possui suporte para *z-order*. Assim, as chamadas a seu método *setToFront()* não pode provocar qualquer alteração visual.

Uma aplicação Ginga-J que desejar ter sua instância da classe *com.sun.dtv.ui.DTVContainer* associada ao plano de gráficos exibida na frente das instâncias da classe *com.sun.dtv.ui.DTVContainer* petencentes a outras aplicações deve utilizar o método *setToFront*(). Chamadas sucessivas a este método não podem causar qualquer efeito visual adicional.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
Plane graphicPlane=null;
Plane[] planes = Screen.getCurrentScreen().getAllPlanes();

for(int i = 0; i < planes.length;i++){

    if ( planes[i].getID() == "GraphicPlane") {
        graphicPlane=planes[i];
        break;
    }
}

DTVContainer graphicDTVContainer=DTVContainer.getDTVContainer(graphicPlane);
graphicDTVContainer.setToFront();
```

8.4.4 Tratamento de eventos do usuário

8.4.4.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 8.4.4, devem ser considerados.

Além disso, a *thread* LWUIT EDT é sincronizada à *thread* de eventos do AWT, de forma a garantir a integração adequada entre ambos os *frameworks* (LWUIT e AWT).

8.4.4.2 Reserva de evento de teclas (controle remoto)

A fim de evitar problemas em um cenário de múltiplas aplicações, as aplicações Ginga-J devem registrar-se explicitamente com a API de reserva de eventos de teclas (ver *com.sun.dtv.ui.event.UserInputEventManager* da ABNT NBR 15606-6), a fim de receber eventos sobre instâncias dos componentes LWUIT *com.sun.dtv.lwuit.TextField* e *com.sun.dtv.lwuit.TextArea*. Depois de receber os eventos reservados, o tratador de evento devem transmiti-los à respectiva instância da classe *com.sun.dtv.lwuit.Form* ou *com.sun.dtv.lwuit.Textfield* ou *com.sun.dtv.lwuit.TextArea* (utilizando o método *keyPressed()*). Desta forma, as instâncias das classes *com.sun.dtv.lwuit.TextField* e *com.sun.dtv.lwuit.TextArea* são capazes de receber eventos ainda quando múltiplas aplicações estejam simultaneamente em execução e uma aplicação não focada tenha reservado os eventos de teclas.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
UserInputEventManager manager;
RemoteControlEvent evt;
Screen currentScreen = Screen.getCurrentScreen();

RemoteControl remoteControl = currentScreen.getRemoteControl();
evt = (RemoteControlEvent) remoteControl.getInitiatedEvent(RemoteControlEvent.VK_1);
manager = UserInputEventManager.getUserInputEventManager(currentScreen);

UserInputEventListener listener = new UserInputEventListener(){
    public void userInputEventReceived(UserInputEvent event) {
        // handle event
    }
}

inputManager.addUserInputEventListener( listener, evt);
```

Como uma diretriz geral para o desenvolvimento de aplicações Ginga-J, estas devem liberar todos os eventos de teclas reservados antes de concluir a sua destruição (por exemplo, chamando *com.sun.dtv.ui.event.KeyEvent#release()* na implementação do método *javax.microedition.xlet.Xlet#destroyXlet()*).

8.5 Informações de serviço e seleção de serviço

Os requisitos da ABNT NBR 15606-4:2010, 8.4.5.1 e 8.4.5.2, devem ser considerados.

Além disso, ver 9.6, 9.7, 9.8, 9.9, 9.10, 10.14 e 10.17.

8.6 Apresentação de mídia

8.6.1 Regra geral

Toda apresentação de conteúdos de mídias é feita através da *API Java Media Framework* (JMF).

8.6.2 Visão geral

O método *javax.media.Manager#getSystemTimeBase()* sempre retorna uma instância de *javax.media.TimeBase* com base no tempo da tabela TOT (ver ABNT NBR 15603). Essa é o *TimeBase* padrão.

Somente instâncias de *javax.media.protocol.DataSource* criadas por *javax.media.Manager* são suportadas.

Como uma diretriz de desenvolvimento de aplicação geral, as aplicações Ginga-J devem liberar os recursos utilizados por seus apresentadores JMF durante sua destruição (por exemplo, chamando o método *java.media.Player#deallocate()* na implementação do método *destroyXlet()*).

8.6.3 Suporte a apresentador de mídia de componente de vídeo associado a um serviço

As aplicações Ginga-J podem utilizar um apresentador de mídia JMF para fins de controle de exibição de componente de vídeo de um serviço.

Comportamento geral

As condições a seguir definem o comportamento esperado de uma instância de apresentador JMF criado para controlar a apresentação de um componente de vídeo associado a um serviço:

— As aplicações Ginga-J são capazes de criar instâncias de apresentadores apenas para os componentes que estão localizados no serviço corrente.

— As instâncias de apresentadores são criadas no estado *Unrealized*.

— Instâncias de apresentadores controlam a exibição do componente de vídeo associado a um serviço quando estiverem no estado *Realized*.

— Apenas uma instância de apresentador associado ao componente de vídeo de um serviço pode estar no estado *Realized* por vez.

— Uma vez iniciado, interromper o apresentador JMF associado ao componente de vídeo de um serviço que interrompe (ou oculta) de fato a exibição do componente de vídeo.

— As aplicações Ginga-J podem parar e fechar uma instância de apresentador e iniciar uma outra instância associada a um outro componente de vídeo.

— Quando a aplicação Ginga-J obtém o controle do apresentador de vídeo, ela é responsável pela exibição correta do serviço. A plataforma evita controlar a exibição de vídeo até o término da aplicação (isto é, enquanto a aplicação estiver no estado STARTED), ainda que a aplicação feche o apresentador. A plataforma pode assumir de volta o controle da exibição do vídeo quando a aplicação estiver no estado PAUSED.

— Somente após o término da aplicação Ginga-J, a exibição de serviço é restaurada (reprodução de vídeo, *clipping* e dimensionamento) pela plataforma.

— O método *javax.media.Player#getVisualComponent()* retorna uma instância de um *java.awt.Component* que representa o vídeo como um componente a ser inserido na hierarquia AWT.

— O método *javax.media.Player#getGainControl()* retorna *null*.

— O método *javax.media.Player#getControlPanelComponent()* retorna *null*.

— Instâncias de apresentadores associadas a componentes de vídeo de um serviço implementam a interface *javax.tv.service.selection.ServiceMediaHandler.*

— Gerenciando outras instâncias de controladores (*javax.media.Controller*): controladores de conteúdo de serviço podem apenas gerenciar outros controladores de conteúdo de serviço (isto é, um apresentador associado a um componente de serviço de vídeo pode gerenciar somente um apresentador associado a um componente de serviço de áudio).

— A duração retornada pelo método *javax.media.Duration#getDuration()* é sempre *DURATION_UNKNOWN.*

— O método *javax.tv.service.selection.ServiceContext#getServiceContentHandlers()* retorna um vetor de *javax.tv.service.selection.ServiceContentHandler* para cada um dos componentes do serviço corrente. No caso de um componente de vídeo em execução no serviço, a instância de *javax.tv.service.selection.ServiceContentHandler* deve ainda implementar *javax.tv.service.selection.ServiceMediaHandler* e deve ser uma instância de *javax.media.Player*. Nesse caso, esse apresentador retornado está no estado *Started*.

— O método *com.sun.dtv.ui.PlaneSetup#getVideoController()*, quando chamado de um *com.sun.dtv.ui.PlaneSetup* associado ao plano de vídeo, deve retornar a instância *javax.media.Player* correspondente ao componente de vídeo do serviço corrente ou *null*. No caso das instâncias *com.sun.dtv.ui.PlaneSetup* associadas aos planos de tela diferentes do plano de vídeo, esse método deve sempre retornar *null*.

Origem de dados

As seguintes definições devem ser consideradas ao tratar de instâncias da classe *javax.midia.protocol.DataSource* para controle da apresentação de componente de vídeo associado a um serviço:

— O localizador abaixo referencia um componente de vídeo de um serviço e pode ser usado para criar uma instância da classe *javax.midia.protocol.DataSource* para este tipo de apresentador de mídia:

dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<event_id>]/<component_tag>

— As aplicações Ginga-J podem utilizar a classe *javax.tv.service.navigation.ServiceComponent*, a fim de obter facilmente essas URL no tempo de execução.

— Uma instância de *javax.midia.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Positionable*.

— Uma instância de *javax.midia.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Seekable*. As aplicações Ginga-J podem utilizar um apresentador de mídia JMF para fins de controle da apresentação do componente de vídeo de um serviço.

— Uma instância de *javax.midia.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Rateconfigureable*.

Posicionamento, escalonamento e *clipping* de vídeo

O suporte a escalonamento e *clipping* de vídeo deve estar de acordo com ABNT NBR 15606-1. Outros fatores de escala, não obrigatórios, podem ser suportados através de uma implementação de *middleware* específica. Sempre que um fator de escala (não obrigatório) é solicitado pelas aplicações, no caso de não ser suportado, a implementação do *middleware* pode aproximá-lo do fator de escala obrigatório mais próximo.

O vídeo pode ser posicionado e escalonado usando um dos seguintes mecanismos:

— O controle JMF *javax.media.AWTVideoSizeControl*.

— Alterando as coordenadas da instância *java.awt.Component* retornada pelo método *javax.media.Player#getVisualComponent()* do apresentador JMF.

— Utilizando a classe LWUIT *com.sun.dtv.lwuit.MediaComponent*.

Ao utilizar *java.awt.Component* retornado por *javax.media.Player#getVisualComponent()*, o vídeo é renderizado sobre todos os conteúdos gráficos. É uma decisão de implementação, se outros componentes gráficos adicionados sobre este *java.awt.Component* puderem ser renderizados.

O controle JMF *com.sun.dtv.media.format.ArbitraryVideoScalingControl* retorna os intervalos definidos pelos fatores de escala vertical e horizontal mínima e máxima, suportados pela plataforma. Pelo menos os fatores de escala mínima e máxima estabelecidos na ABNT NBR 15606-1 devem ser suportados.

Media Clock e *TimeBase*

As seguintes restrições aplicam-se às definições em JAVATV 1.1:2008 para documentação da interface *javax.media.Clock*:

— O *TimeBase* padrão é estabelecido de acordo com o tempo transmitido na tabela TOT (ver ABNT NBR 15603-2) corrente.

— A taxa de reprodução é sempre 1.0.

— O tempo inicial da mídia é 0.

— O tempo de início do *TimeBase* é 0.

— O tempo da mídia não pode ser configurado (atribuído). Chamadas ao método *javax.media.Clock#setMediaTime()* são ignoradas.

— Ao tentar configurar (atribuir) um *TimeBase* diferente do *TimeBase* padrão utilizando *javax.media.Clock#setTimeBase()*, um *javax.media.IncompatibleTimeBaseException* é lançado.

Controles disponíveis

Os seguintes controles JMF devem estar disponíveis para o apresentador de mídia associado a um componente de vídeo de um serviço:

— *com.sun.dtv.media.FreezeResumeControl*

— *com.sun.dtv.media.format.ArbitraryVideoScalingControl*

— *com.sun.dtv.media.format.ClippingControl*

— *javax.tv.media.AWTVideoSizeControl*

O método *javax.media.Control#getControlComponent()* implementado por todos os controles acima sempre retorna *null*.

8.6.4 Suporte a apresentador de mídia de componente de áudio associado a um serviço

As aplicações Ginga-J podem utilizar um apresentador de mídia JMF para fins de controle de exibição de componente de áudio de um serviço.

Comportamento geral

As condições a seguir definem o comportamento esperado de uma instância de apresentador JMF criado para controlar a apresentação de um componente de áudio associado a um serviço:

— As aplicações Ginga-J devem ser capazes de criar instâncias de apresentadores apenas para os componentes que estão localizados no serviço corrente.

— As instâncias de apresentadores são criadas no estado *Unrealized*.

— As instâncias de apresentadores controlam a execução do componente de áudio associado a um serviço quando estiverem no estado *Realized*.

— Apenas uma instância de apresentador associado ao componente de áudio de um serviço pode estar no estado *Realized* por vez.

— Uma vez iniciado, interromper o apresentador JMF associado ao componente de áudio de um serviço que interrompe de fato a reprodução do componente de áudio.

— As aplicações Ginga-J podem parar e fechar uma instância de apresentador e iniciar outra instância associada a um outro componente de áudio.

— Quando a aplicação Ginga-J obtém o controle do apresentador de áudio, ela é responsável pela exibição correta do serviço. A plataforma evita controlar a reprodução de áudio até o término da aplicação (isto é, enquanto a aplicação estiver no estado STARTED), ainda que a aplicação feche o apresentador.

A plataforma pode assumir de volta o controle da reprodução de áudio quando a aplicação estiver no estado PAUSED.

— Somente após o término da aplicação Ginga-J, a exibição de serviço é restaurada (reprodução de áudio e nível de volume) pela plataforma.

— O método *javax.media.Player#getVisualComponent()* retorna *null*.

— O método *javax.media.Player#getGainControl()* retorna um objeto implementando o controle JMF *javax.media.GainControl.*

— O método *javax.media.Player#getControlPanelComponent()* retorna *null*.

— Instâncias de apresentadores associados a componentes de áudio de um serviço implementam a interface *javax.tv.service.selection.ServiceMediaHandler*.

— Gerenciar outras instâncias de controladores (*javax.media.Controller*): os controladores de conteúdo de serviço podem apenas gerenciar outros controladores de conteúdo de serviço (isto é, um apresentador associado a um componente de serviço de áudio pode gerenciar somente um apresentador associado a um componente de serviço de vídeo).

— O valor retornado pelo método *javax.media.Duration#getDuration()* sempre é *DURATION_UNKNOWN.*

— O método *javax.tv.service.selection.ServiceContext#getServiceContentHandlers()* retorna um vetor de *javax.tv.service.selection.ServiceContentHandler* para cada um dos componentes do serviço corrente. No caso de um componente de áudio em execução no serviço, a instância de *javax.tv.service.selection.ServiceContentHandler* deve ainda implementar *javax.tv.service.selection.ServiceMediaHandler* e deve ser uma instância de *javax.media.Player*. Nesse caso, esse apresentador retornado está no estado *Started*.

Origem de dados

As seguintes definições devem ser consideradas ao tratar de instâncias da classe *javax.media.protocol.DataSource* para o controle da apresentação de componente de áudio associado a um serviço:

— O localizador a seguir faz referência a um componente de áudio de um serviço e pode ser usado para criar uma instância de *javax.media.protocol.DataSource* para este tipo de apresentador de mídia:

dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<event_id>]/<component_tag>

— As aplicações Ginga-J podem utilizar a classe *javax.tv.service.navigation.ServiceComponent*, a fim de obter facilmente essas URL em tempo de execução.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Positionable*.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Seekable*.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo não pode implementar a interface *javax.media.protocol.Rateconfigureable*.

Controle do volume de áudio

As aplicações Ginga-J podem controlar o volume de áudio utilizando a instância de *javax.media.GainControl* retornada pelo método *javax.media.Player#getGainControl()* ou ainda utilizando o método

javax.media.Player#getControls() . A faixa de ganho efetivo varia do nível 0.0 (mudo) até o nível 1.0. Por padrão, o nível de ganho é configurado para 1.0. O nível de ganho 1.0 corresponde ao ajuste de volume atual no receptor.

Em escala dB, o ganho de 0 dB é equivalente ao nível de ganho 1.0. A escala dB efetiva está na faixa de – 20 dB a 0 dB. Valores de ganho inferiores a - 20 dB são considerados nível 0.0.

Media Clock* e *TimeBase

As seguintes restrições aplicam-se às definições em JAVATV 1.1:2008 para documentação da interface *javax.media.Clock*:

— O *TimeBase* é estabelecido de acordo com o tempo transmitido na tabela TOT (ver ABNT NBR 15603-2) corrente.

— A taxa de reprodução é sempre 1.0.

— O tempo inicial da mídia é 0.

— O tempo de início do *TimeBase* é 0.

— O tempo da mídia não pode ser configurado (atribuído). Chamadas ao método *javax.media.Clock#setMediaTime()* são ignoradas.

— Ao tentar configurar (atribuir) um *TimeBase* diferente do *TimeBase* padrão utilizando *javax.media.Clock#setTimeBase()*, um *javax.media.IncompatibleTimeBaseException* é lançado.

Controles disponíveis

Os seguintes controles JMF devem estar disponíveis para o apresentador de mídia associado a um componente de áudio de serviço:

— *javax.media.GainControl*

— *com.sun.dtv.media.audio.AudioControl*

— *com.sun.dtv.media.language.LanguageControl*

O método *javax.media.Control#getControlComponent()*, implementado por todos os controles anteriores, sempre retorna *null*.

8.6.5 Suporte a apresentador de mídia de monomídias de vídeo (não suportado no Perfil A)

Uma aplicação Ginga-J pode utilizar um apresentador de mídia para reprodução de arquivos de monomídias de vídeo de acordo com ABNT NBR 15606-1.

Esse recurso não está disponível em receptores que implementam o Perfil A da especificação Ginga. Por essa razão, as aplicações devem ser preparadas, caso o suporte para tal recurso não esteja disponível (para mais detalhes, ver 8.11).

Comportamento geral

As condições a seguir definem o comportamento esperado de uma instância de apresentador JMF criado para controlar a exibição de monomídias de vídeo:

— Apenas uma instância de apresentador pode estar no estado *Realized* por vez.

— As aplicações Ginga-J podem parar e fechar uma instância de apresentador e iniciar uma outra instância associada a um arquivo de monomídia de vídeo diferente.

— O método *javax.media.Player#getVisualComponent()* retorna uma instância de um *java.awt.Component* que representa o vídeo como um componente a ser inserido na hierarquia AWT.

— O método *javax.media.Player#getGainControl()* retorna *null*.

— O método *javax.media.Player#getControlPanelComponent()* retorna *null*.

— Gerenciando outras instâncias de controlador (*javax.media.Controller*): os controladores de monomídias podem apenas gerenciar outros controladores de monomídias (isto é, um apresentador de uma monomídia de vídeo pode somente gerenciar um apresentador de monomídia de áudio).

— O valor retornado pelo método *javax.media.Duration#getDuration()* pode retornar *DURATION_UNKNOWN* se a duração de mídia não puder ser determinada pelo receptor.

Origem de dados

Os seguintes localizadores ou URL fazem referência a um arquivo de monomídia de vídeo no carrossel de objetos DSMCC e podem ser utilizados para criar uma instância da classe *javax.media.protocol.DataSource* (ou *MediaLocator* ou *Player*) para esse tipo de apresentador de mídia:

— *dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<event_id>]/<DSMCC_component_tag>/<object_carousel_id>/[{<file_directory>/}]<media_file>*

— *file:///[{<file_directory>/}]<media_file>*

As aplicações Ginga-J podem utilizar a classe *javax.tv.service.navigation.ServiceComponent*, a fim de obter facilmente a URL de *DSMCC_component_tag* em tempo de execução.

Somente arquivos contidos em um carrossel de objetos DSM-CC montado podem ser utilizados.

Além disso, o seguinte tipo de URL pode ser usado para reprodução de arquivos de monomídia persistidos previamente no receptor:

— *file:///<root>/<organization_id>/<application_id>/[{<file_directory>/}]<media_file>*

Onde o valor *<root>* pode ser obtido adequadamente do sistema *com.sun.dtv.persistent.root* (ver ABNT NBR 15606-6).

As seguintes definições devem ser consideradas para instâncias da classe *javax.media.protocol.DataSource* associadas a apresentadores de monomídias de vídeo retornados pela plataforma:

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.Positionable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

— Uma instância *javax.media.protocol.DataSource* para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.Seekable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.RateConfigureable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

Posicionamento, escalonamento e *clipping* de vídeo

O suporte a escalonamento e *clipping* de vídeo deve estar de acordo com a ABNT NBR 15606-1. Outros fatores de escala, não obrigatórios, podem ser suportados através de uma implementação de *middleware* específica. Sempre que um fator de escala (não obrigatório) é solicitado pelas aplicações, no caso de não ser suportado, a implementação do *middleware* pode aproximá-lo ao fator de escala obrigatório mais próximo.

O vídeo pode ser posicionado e escalonado usando os seguintes mecanismos:

— Utilizando o controle JMF *javax.media.AWTVideoSizeControl*.

— Alterando as coordenadas da instância *java.awt.Component* retornada pelo método *getVisualComponent()* do apresentador JMF.

— Utilizando a classe LWUIT *com.sun.dtv.lwuit.MediaComponent*.

Ao utilizar a instância de *java.awt.Component* retornado por *javax.media.Player#getVisualComponent()*, o vídeo deve ser renderizado sobre todos os conteúdos gráficos. É uma decisão de implementação se outros componentes gráficos adicionados sobre este *java.awt.Component* puderem ser renderizados.

O controle JMF *com.sun.dtv.media.format.ArbitraryVideoScalingControl* retorna os intervalos definidos pelos fatores de escala vertical e horizontal mínima e máxima, suportados pela plataforma. Pelo menos os fatores de escala mínima e máxima estabelecidos na ABNT NBR 15606-1 devem ser suportados.

Media Clock* e *TimeBase

As seguintes restrições aplicam-se às definições em JAVATV 1.1:2008 para documentação de interface:

— O *TimeBase* padrão é o tempo transmitido na tabela TOT (ver ABNT NBR 15603-2) corrente.

— A taxa de reprodução é de 1.0 por padrão. O suporte a diferentes taxas é opcional.

— O tempo inicial da mídia é 0 por padrão. O suporte ao início da mídia em diferentes pontos desde o início é opcional.

— O tempo de início de *TimeBase* é o tempo transmitido na TOT quando o apresentador inicia.

— O suporte à configuração (atribuição) do tempo de mídia é opcional. Se não suportadas, as chamadas para *TimeBase* padrão é opcional. Se não suportado, ao tentar configurar (atribuir) um *TimeBase* diferente do *TimeBase* padrão utilizando *javax.media.Clock#setTimeBase()*, uma exceção do tipo *javax.media.IncompatibleTimeBaseException* é lançada.

Controles disponíveis

Os seguintes controles JMF devem estar disponíveis para o apresentador de mídia associado a uma monomídia de vídeo:

— *com.sun.dtv.media.FreezeResumeControl*

— *com.sun.dtv.media.format.ArbitraryVideoScalingControl*

— *com.sun.dtv.media.format.ClippingControl*

— *javax.tv.media.AWTVideoSizeControl*

— *javax.tv.media.CachingControl* (opcional e apenas para arquivos mpeg no DSM-CC)

— *com.sun.dtv.media.MediaTimePositionControl* (opcional e apenas em caso de arquivos mpeg no DSM-CC).

O método *javax.media.Control#getControlComponent()* implementado por todos os controles acima sempre retorna *null*.

8.6.6 Suporte a apresentador de mídia de monomídias de áudio

Uma aplicação Ginga-J pode utilizar um apresentador de mídia para reprodução de arquivos de monomídias de áudio de acordo com a ABNT NBR 15606-1.

Comportamento geral

As condições a seguir definem o comportamento esperado de uma instância de apresentador JMF criado para controlar a exibição de monomídias de áudio:

— Apenas uma instância de apresentador pode estar no estado *Realized* por vez.

— As aplicações Ginga-J podem parar e fechar uma instância de apresentador e iniciar outra instância de apresentador associada a um outro arquivo de monomídia de áudio.

— O método *javax.media.Player#getVisualComponent()* retorna *null*.

— O método *javax.media.Player#getGainControl()* retornará um objeto implementando o controle JMF *javax.media.GainControl*.

— O método *javax.media.Player#getControlPanelComponent()* retorna *null*.

— Gerenciando outras instâncias de controladores (*javax.media.Controller*): os controladores de monomídias podem apenas gerenciar outros controladores de monomídias (isto é, um apresentador de uma monomídia de áudio pode somente gerenciar um apresentador de monomídia de vídeo).

— O valor retornado pelo método *javax.media.Duration#getDuration()* pode retornar *DURATION_UNKNOWN* se a duração de mídia não puder ser determinada pelo receptor.

Origem de dados

Os seguintes localizadores e URL fazem referência a um arquivo de monomídia de áudio no carrossel de objetos DSMCC e podem ser utilizados para criar uma instância da classe *javax.media.protocol.DataSource* (ou *MediaLocator* ou *Player*) para esse tipo de apresentador de mídia:

— *dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<event_id>]/<DSMCC_component_tag>/<object_carousel_id>/[{<file_directory>/}]<media_file>*

— *file:///[{<file_directory>/}]<media_file>*

Uma aplicação Ginga-J pode utilizar o *javax.tv.service.navigation.ServiceComponent*, a fim de obter facilmente o URL de *DSMCC_component_tag* em tempo de execução.

Somente conteúdo de arquivo em um carrossel de objetos DSMCC ativo pode ser utilizado.

Além disso, o seguinte tipo de URL pode ser usado para reprodução de arquivos de monomídia mantidos previamente no armazenamento persistente:

— *file:///<root>/<organization_id>/<application_id>/[{<file_directory>/}]<media_file>*

Onde o valor *<root>* pode ser obtido adequadamente do sistema *com.sun.dtv.persistent.root* (ver ABNT NBR 15606-6).

As seguintes definições devem ser consideradas para instâncias da classe *javax.media.protocol.DataSource* associadas a apresentadores de monomídias de áudio:

— Uma instância de *javax.media.protocol.*DataSource para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.Positionable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.Seekable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

— Uma instância de *javax.media.protocol.DataSource* para esse tipo de conteúdo pode implementar a interface *javax.media.protocol.RateConfigureable*. As aplicações Ginga-J podem verificar se a instância de *javax.media.protocol.DataSource* retornada implementa essa interface.

Controle do volume do áudio

As aplicações Ginga-J podem controlar o volume de áudio utilizando a instância de *javax.media.GainControl* retornada pelo método javax.media.Player*#getGainControl()* ou ainda utilizando o método *javax.media.Player#getControls().* A faixa de ganho efetivo varia do nível 0.0 (mudo) até o nível 1.0. Por padrão, o nível de ganho é 1.0. O nível de ganho 1.0 corresponde ao ajuste de volume atual no receptor.

Em escala dB, o ganho de 0 dB é equivalente ao Nível de ganho 1.0. A escala dB efetiva está na faixa de – 20 dB a 0 dB. Valores de ganho inferiores a – 20 dB são considerados nível 0.0.

Media Clock* e *TimeBase

As seguintes restrições aplicam-se às definições em JAVATV 1.1:2008 para documentação de interface *javax.media.Clock*:

— O tempo *TimeBase* é o tempo da tabela TOT (ver ABNT NBR 15603-2).

— A taxa de reprodução é de 1.0 por padrão. O suporte a diferentes taxas é opcional.

— O tempo inicial da mídia é 0 por padrão. O suporte ao início da mídia em diferentes pontos desde o início é opcional.

— O tempo de início de *TimeBase* é o tempo transmitido na tabela TOT quando o apresentador inicia.

— O suporte à configuração (atribuição) do tempo de mídia é opcional. Se não suportadas, as chamadas para *javax.media.Clock#setMediaTime()* são ignoradas.

— Suporte a *TimeBases* diferentes do *TimeBase* padrão é opcional. Se não suportado, ao tentar configurar (atribuir) um *TimeBase* diferente do *TimeBase* padrão utilizando *javax.media.Clock#setTimeBase()*, uma exceção do tipo *javax.media.IncompatibleTimeBaseException* é lançada.

Controles disponíveis

Os seguintes controles JMF devem estar disponíveis para o apresentador de mídia associado a uma monomídia de áudio:

— *javax.media.GainControl*

— *javax.tv.media.CachingControl* (opcional e apenas para arquivos *MPEG* no *DSM-CC*)

— *com.sun.dtv.media.MediaTimePositionControl* (opcional e apenas para arquivos *MPEG* no *DSM-CC*)

O método *javax.media.Control#getControlComponent()*, implementado por todos os controles acima, sempre retorna *null*.

8.6.7 Suporte a apresentador de mídia de MNG

O tipo de mídia de MNG é apenas suportado usando a *framework* JMF e, portanto, deve ser tratado por um apresentador de mídia comum. Por outro lado, as aplicações Ginga-J que utilizam a *framework* LWUIT podem utilizar a classe *com.sun.dtv.lwuit.MediaComponent* para apresentar arquivos de monomídia de MNG.

Comportamento geral

As condições a seguir definem o comportamento esperado de uma instância de apresentador de monomídias de MNG:

— O método *javax.media.Player#getVisualComponent()* retorna uma instância da classe *java.awt.Component*, que representa o vídeo como um componente a ser inserido na hierarquia AWT.

— O método *javax.media.Player#getGainControl()* retorna *null*.

— O método *javax.media.Player#getControlPanelComponent()* retorna *null*.

— Gerenciando outras instâncias de controladores (*java.media.Controller*): apresentadores desse tipo não podem ser controlados ou controlar outros controladores.

— O valor retornado pelo método *javax.media.Duration#getDuration()* pode retornar *DURATION_UNKNOWN*, se a duração de mídia não puder ser determinada pelo receptor.

Origem de dados

Os seguintes localizadores ou URL fazem referência a um arquivo MNG no carrossel de objetos DSM-CC e podem ser utilizados para criar uma instância de *javax.media.protocol.DataSource* (ou *MediaLocator* ou *Player*) para esse tipo de apresentador de mídia:

— *dtv://<original_network_id>.<transport_stream_id>.<service_id>[;<event_id>]/<DSMCC_component_tag>/<object_carousel_id>/[{<file_directory>/}]<mng_file>*

— *file:///[{<file_directory>/}]<mng_file>*

As aplicações Ginga-J podem utilizar a classe *javax.tv.service.navigation.ServiceComponent*, a fim de obter facilmente a URL de *DSMCC_component_tag* em tempo de execução.

Somente conteúdo de arquivo presente em um carrossel de objetos DSM-CC pode ser utilizado.

Além disso, o seguinte tipo de URL pode ser usado para reprodução de arquivos MNG mantidos previamente no armazenamento persistente:

— *file:///<root>/<organization_id>/<application_id>/[{<file_directory>/}]<mng_file>*

Onde o valor *<root>* pode ser obtido adequadamente do sistema *com.sun.dtv.persistent.root* (ver ABNT NBR 15606-6).

Posicionamento, escalonamento e *clipping*

O *clipping* não é suportado nos apresentadores de monomídias de MNG.

Os apresentadores de MNG podem ser posicionados e escalonados usando os seguintes mecanismos:

— Utilizando o controle JMF interno *javax.media.AWTVideoSizeControl*.

— Alterando os limites da instância *java.awt.Component* retornada pelo método *getVisualComponent()* do apresentador JMF.

— Utilizando a classe LWUIT *com.sun.dtv.lwuit.MediaComponent* .

Ao utilizar o *java.awt.Component* retornado por *javax.media.Player#getVisualComponent()*, o MNG deve ser renderizado sobre todos os conteúdos gráficos. Outros componentes de gráficos adicionados sobre esse *java.awt.Component* podem não ser renderizados pela plataforma.

Media Clock* e *TimeBase

As seguintes restrições aplicam-se às definições em JAVATV 1.1:2008 para documentação de interface *javax.media.Clock*:

— O tempo *TimeBase* é o tempo transmitido na TOT corrente.

— A taxa de reprodução é de 1.0 por padrão.

— O tempo de início da mídia é 0.

— O tempo de início de *TimeBase* é o tempo transmitido na TOT quando o apresentador inicia.

— O suporte a configuração (atribuição) do tempo de mídia é opcional. Se não suportadas, as chamadas para *javax.media.Clock#setMediaTime()* são ignoradas.

— Suporte a *TimeBases* diferentes do *TimeBase* padrão é opcional. Ao tentar configurar (atribuir) um *TimeBase* diferente do *TimeBase* padrão utilizando *javax.media.Clock#setTimeBase()*, um *javax.media.IncompatibleTimeBaseException* é lançado.

Controles disponíveis

Os seguintes controles JMF estão disponíveis para o apresentador de mídia associado a arquivos de monomídia de MNG:

— *javax.tv.media.AWTVideoSizeControl*

O método *javax.media.Control#getControlComponent()*, implementado por todos os controles anteriores, sempre retorna *null*.

É apresentado a seguir um exemplo de código de apresentação de um apresentador MNG em um formulário LWUIT:

```
try {
    player = Manager.createPlayer(new MediaLocator("sample.mng"));
} catch (Exception e) {
    // no player available
}
if (player.getState() < Controller.Realized) {
    player.realize();
}

frm = new Form();
MediaComponent mediaComponent = new MediaComponent( player);
mediaComponent.setPreferredSize(new Dimension(100, 100));
frm.addComponent( mediaComponent);
frm.show();

// starting...
try {
    mediaComponent.startMedia();
} catch (MediaException e) {
    // it is not possible to start this media
}
```

A seguir é apresentado um exemplo de código de uso de um apresentador de MNG inserido em um *java.awt.Container*:

```
try {
    player = Manager.createPlayer(new MediaLocator( "sample.mng"));
} catch (Exception e) {
    // no player available
}

player.start();

container.add(player.getVisualComponent());
container.setVisible(true);
container.validate();
container.repaint();
```

8.7 Acesso a dados

8.7.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 8.7.1, devem ser considerados.

8.7.2 Acesso a arquivos

Os requisitos da ABNT NBR 15606-4:2010, 8.7.2, devem ser considerados.

As aplicações Ginga-J podem acessar arquivos utilizando URL com o protocolo *file* (ou seja file:///) de acordo com as configurações do *hardware* da plataforma.

8.7.3 Protocolo de transporte no canal de difusão

Os requisitos da ABNT NBR 15606-4:2010, 8.7.3, devem ser considerados.

8.7.4 Acesso ao sistema de arquivos persistente

Os requisitos da ABNT NBR 15606-4:2010, 8.7.4, devem ser considerados.

O acesso ao espaço de armazenamento persistente pode estar disponível para aplicações Ginga-J de acordo com as configurações de *hardware* da plataforma. Utilizando a propriedade do sistema *com.sun.dtv.persistent.root* (ver–ABNT NBR 15606-6), as aplicações podem definir onde podem armazenar seus dados. Neste caso, o caminho onde a plataforma armazena dados relacionados a aplicações Ginga-J é construído da seguinte forma:

— *<persistent_root_path>/<organization_id>/<application_id>/*

Onde *<persistent_root_path>* é o valor retornado pela propriedade *com.sun.dtv.persistent.root* (por exemplo, *file:///ginga)*, *<organization_id>* é o identificador único da organização e *<application_id>* é o identificador único da aplicação indicados no *application_identifier_descriptor* da tabela AIT.

Todas as aplicações Ginga-J com o mesmo <organization_id> devem ter permissão de acesso de leitura/escrita ao diretório *<persistent_root_path>/<organization_id>/*. Dessa forma, as aplicações Ginga-J fornecidas pela mesma organização podem compartilhar dados usando esse diretório.

De acordo com ABNT NBR 15606-1, o espaço de armazenamento persistente disponível a ambas as máquinas de execução e apresentação (Ginga-J e Ginga-NCL) no *middleware* Ginga é de 47 KB. Como orientação geral para o desenvolvimento de aplicações, os autores de aplicações devem adotar um comportamento conservador, armazenando pequenas quantidades de dados nesta área.

Os requisitos da ABNT NBR 15606-4:2010, 8.7.4, requerem que os dados persistidos sejam mantidos no receptor pelo tempo em que o serviço de TV digital estiver sintonizado no receptor, a aplicação for sinalizada nesse serviço e seu *application_control_code* for diferente de *KILL* ou *DESTROY*. Por essa razão, é aceitável aos receptores que não têm capacidades reais de armazenamento persistente emular o armazenamento persistente na memória RAM (ou seja, utilizando um sistema de arquivos temporal).

8.7.5 Acesso a propriedades do sistema

Os requisitos da ABNT NBR 15606-4:2010, 8.7.5, devem ser considerados.

8.7.6 Suporte de IP sobre canal de interatividade

Os requisitos da ABNT NBR 15606-4:2010, 8.7.6, devem ser considerados.

Atualmente, nenhuma norma publicada estabelece as tecnologias disponíveis para dispositivos de canal interativo e as propriedades que devem ser utilizadas com cada um deles, a fim de estabelecer conexões. Por essa razão, caso o dispositivo de canal interativo exija que se estabeleça uma conexão de nível inferior à camada de transporte (camadas de rede, enlace ou física) antes de estabelecer uma conexão TCP ou UDP, esse fato é ocultado da aplicação Ginga e totalmente tratado pela plataforma. As aplicações Ginga podem utilizar o canal interativo sem considerar quaisquer requisitos de conexão de nível inferior e sem verificar a tecnologia utilizada pelo dispositivo de canal interativo subjacente.

O pacote *com.sun.dtv.net* estabelecido na ABNT NBR 15606-6 é destinado a permitir às aplicações o acesso detalhado e controle do dipostivido de canal interativo. Como atualmente não há nenhuma norma que estabeleça as tecnologias necessárias, padronizando os parâmetros de configuração e o comportamento dos diferentes dispositivos de canal interativo que podem ser usados em um receptor, as aplicações não podem confiar nas funcionalidades fornecidas por tal pacote.

8.7.7 Filtragem de seção

Os requisitos da ABNT NBR 15606-4:2010, 8.7.7, devem ser considerados.

Antes de iniciar a filtragem de seções usando a API de MPEG-2 *Section Filtering* estabelecida na ABNT NBR 15606-6, as aplicações Ginga-J devem configurar o valor do identificador de pacote (*PID)* definido adequadamente, utilizando o método *com.sun.dtv.filtering.DataSectionFilter#setPID(int)*. Opcionalmente, as aplicações Ginga podem configurar o identificador de tabela (*table_id*) desejado para filtragem usando o método *com.sun.dtv.filtering.DataSectionFilter#setTableId (int)*.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
try{
    Tuner [] t = Tuner.getInstances();
    DataSectionFilterCollection collection;
    collection = new DataSectionFilterCollection(t[0].getCurrentTransportStream(), 2);

    c.reserve(true, 10000, new ScarceResourceListener() {
        public boolean releaseRequested(ScarceResource resource) {
                return true;
        }
        public void releaseForced(ScarceResource resource) {
                // save state
        }
        public void released(ScarceResource resource) {
                // resource was taken
        }
    });

    CircularFilter filter = collection.newCircularFilter(1);
    circularFilter.setPid(78); //  AIT pid
    //circularFilter.setTableId(116); // (optional) AIT table id
    circularFilter.addSectionFilterListener( new FilterEventListener() {

        public void dataSectionFilterUpdate(DataSectionFilterEvent dataSectionFilterEvent) {
            if (dataSectionFilterEvent.getSource() instanceof CircularFilter) {
                CircularFilter circularFilter;
                circularFilter = (CircularFilter) dataSectionFilterEvent.getSource();
                if (circularFilter.getSections() != null)
                for (int i = 0; i < circularFilter.getSections().length; i++) {
                    DataSection dataSectionOrig = circularFilter.getSections()[i];
                    DataSection dataSectionClone = (DataSection) dataSectionOrig.clone();
                    assertEquals(dataSectionOrig, dataSectionClone);
                }
            }
        }
    });

    collection.connect(this);
    circularFilter.startFiltering(this);

} catch(Exception e) {
    testFailed("Exception: "+e);
}
```

Como uma diretriz geral para o desenvolvimento de aplicações, as aplicações Ginga-J devem liberar, na sua destruição, todas as coleções de filtro de seção reservadas (por exemplo, chamando *com.sun.dtv.filtering.DataSectionFilterCollection#release()* na implementação do método *javax.microedition.xlet.Xlet#destroyXlet()*). Além disso, ao finalizar uma aplicação, o gerenciador de aplicações Ginga-J deve restaurar as alterações feitas pela aplicação nos estados dos filtros de seção.

8.8 Gerenciador de aplicação

Os requisitos da ABNT NBR 15606-4:2010, 8.8, devem ser considerados. Mais esclarecimentos sobre este tópico podem ser encontrados em 7.1, 7.3 e 10.2.

8.9 Sintonização

A sintonização de serviços em diferentes *transport streams* (canais de frequência diferentes) não é permitida para aplicações Ginga.

8.10 Ponte NCL

8.10.1 Considerações gerais

Os requisitos da ABNT NBR 15606-4:2010, 8.10 devem ser considerados.

8.10.2 Ponte Java-NCL

De acordo com o estabelecido na ABNT NBR 15606-2, uma *Xlet* Ginga-J pode ser embutida em um documento NCL. As aplicações Ginga-NCL devem apresentar uma *Xlet* embutida por vez.

Comportamento gráfico

As aplicações Java embutidas em documentos NCL devem comportar-se graficamente como se estivessem em execução em modo *stand-alone*. A área da tela definida por uma região NCL se referindo a uma *Xlet* embutida em um documento NCL é totalmente reservada para a aplicação Java embutida e nenhum desenho deve ser feito pelo formatador NCL, mas apenas pela própria JVM. Assim, as aplicações devem ter cuidado para que não ocorra nenhuma sobreposição com os nós de mídia NCL. Recomenda-se que a *Xlet* embutida seja colocada em uma área transparente do documento NCL, a fim de evitar mostrar qualquer conteúdo sobreposto.

A visibilidade das aplicações Ginga-J embutidas é controlada pelo documento NCL. Quaisquer chamadas ao método *setVisible(boolean)* na instância do DTVContainer do Ginga-J associada ao plano gráfico são ignoradas.

Eventos de teclas

A gestão de foco nos *Xlets* Ginga-J é independente da gestão de foco tratada pelo Formatador NCL. Em um evento *key-pressed*, se a tecla não for reservada pela aplicação Java (ver API TV Specific UI functionality event reservation da ABNT NBR 15606-6), o mesmo evento de teclas pode ser entregue para ambas as aplicações.

Eventos de ciclo de vida

Quando a *Xlet* em execução for um apresentador embutido em um documento NCL, caso termine a execução espontaneamente (por exemplo, por uma chamada a *javax.microedition.xlet.XletContext#notifyDestroyed()*), o resultado deve ser uma transição "stop" do nó de mídia correspondente. Portanto, o documento NCL pode capturar o término espontâneo de uma *Xlet* da mesma forma que pode capturar o "término natural" de qualquer outro tipo de mídia, ou seja, incluindo *links* com uma função "onEnd", vinculado à mídia Xlet.

8.10.3 Ponte Lua-Java

As seguintes orientações definem o comportamento esperado da ponte Lua-Java:

Tipos

Não existe qualquer conversão implícita das classes: *java.lang.Integer*, *java.lang.Short*, *java.lang.Float*, *java.lang.Double*, *java.lang.Long*, *java.lang.Byte* a um número Lua, nem de *java.lang.Character* a uma *string* Lua.

Arrays

Os *arrays* no Lua iniciam com o índice 1 e no Java com o índice 0. Cada linguagem utiliza sua própria convenção, efetuando a conversão quando necessário.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
lua_array = { 'a','b','c' }
java_array = Arrays.asList( lua_array )

print( lua_array[1] ) -- will print the character 'a'
print( java_vector:get(0) ) -- will print the character 'a'
print( java_vector:toArray()[1] ) -- will print the character 'a'
```

As aplicações Ginga devem incluir o campo 'type' em *arrays* de Lua, de modo que ela possa ser usada ao converter um *array* de Lua para um *array* Java. Se o campo 'type' não for fixado, o tipo *java.lang.Object* é assumido por padrão.

Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
integerArray = { 1, 2, 3, type = ginga.java.lang.Integer } -- Integer array

intArray = { 1, 2, 3, type = ginga.java.lang.Integer.TYPE } -- int array

objectArray = { 1, 2, 3 } -- Object array
```

Classe e implementação de objeto em Lua

Algumas convenções são estabelecidas com relação ao seguinte trecho da ABNT NBR 15606-2:2011, 10.4.5:

> *"Toda vez que o método newInstance fornecido por uma classe representada em Lua é chamado, ele retorna uma nova instância (objeto) dessa classe. O objeto retornado é uma tabela Lua que tem todos os membros de instância especificados por sua classe (campos e métodos públicos)."*

Lua não tem o conceito de classes, embora estas possam ser implementadas por meio de metaprogramação. Assim, para fins de esclarecimento público sobre a forma como os métodos públicos devem ser chamados por *scripts* Lua, os métodos de classe são implementados como funções que recebem a instância da classe como seu primeiro parâmetro. Os métodos estáticos não recebem a instância da classe.

NOTA Para outros detalhes, consultar *Programming in Lua.*

Para mais detalhes, consultar o código-fonte de exemplo abaixo:

```
SIManager = ginga.javax.tv.service.SIManager

si = SIManager.createInstance() -- static method call
preferredLanguge = si:getPreferredLanguage() -- method call
```

8.11 Propriedades da plataforma

8.11.1 Propriedades de detalhes do sistema

Os requisitos da ABNT NBR 15606-4:2010, 8.11.1, devem ser considerados.

Descoberta de perfil

As aplicações Ginga podem consultar a versão e o perfil da especificação Ginga-J (ver ABNT NBR 15606-4) que está disponível na implementação Ginga do receptor, a fim de identificar quais funcionalidades são implementadas e estão disponíveis. Isso pode ser feito verificando a propriedade do sistema *system.gingaj_profile* (ver ABNT NBR 15606-4) em tempo de execução. Para mais detalhes, consultar o código-fonte do exemplo a seguir:

```
String profile = System.getProperty("system.gingaj_profile").trim().toUpperCase();
if (profile.equals("A")) {
    // profile A found!
}
else if (profile.equals("B")) {
    // profile B found!
}
else if (profile.equals("C")) {
    // profile C found!
}
else {
    // profile could not be found!
   }
```

Da mesma forma, as aplicações Ginga podem descobrir qual versão Ginga está presente no receptor, utilizando ambas as propriedades *"system.gingaj_version"* e *"system.gingancl_version"*. Ambas as propriedades devem retornar o valor "1.0.0".

A propriedade *"br.org.ginga.system.version"* não pode ser utilizada e deve retornar o valor *null*.

8.11.2 Propriedades de usuário

Os requisitos da ABNT NBR 15606-4:2010, 8.11.2, devem ser considerados.

As aplicações Ginga-J podem armazenar propriedades customizadas no sistema a qualquer momento usando a API *User Preferences*, ver ABNT NBR 15606-6. No entanto, é opcional para a plataforma manter essas propriedades após desligar o receptor.

O espaço mínimo disponível para armazenamento para preferências do usuário é 47 KB.

8.12 Canal de interatividade

Os requisitos da ABNT NBR 15606-4:2010, 8.11.2, devem ser considerados. Além disso, o estabelecido em 8.7.6 e 10.10 deve ser considerado.

8.13 Lista de pacotes em Ginga-J

Devem ser considerados todos os requisitos da ABNT NBR 15606-4:2010, 8.13 e todas as suas subseções.

8.14 Localizadores

Todas as definições de protocolos definidos na classe *com.sun.dtv.locator.URLLocator* (ver ABNT NBR 15606-6) devem ser consideradas. Ademais, as aplicações Ginga-J devem usar o seguinte formato de URL para criar um localizador de arquivos transmitidos em carrosséis objetos DSMCC:

— *dtv://<original_network_id>.<transport_stream_id>.<service_id>/<component_tag>/<carousel_id>/<carousel_file.ext>*

onde:

— *<original_network_id>:* campo *original_network_id* do *transportstream* informado na tabela NIT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *<transport_stream_id>:* campo *transport_stream_id* do *transportstream* informado na tabela NIT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *<service_id>:* campo *program_id* do serviço informado na tabela PAT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *<component_tag>:* identificador do *elementary stream* informado no *stream_identifier_descriptor* (ver ABNT NBR 15603-2) em formato hexadecimal;

— *<carousel_id>:* identificador do carrousel de objetos DSMCC informado na tabela DII (ver ABNT NBR 15606-3) em formato hexadecimal;

— *<carousel_file.ext>:* caminho completo ao arquivo transmitido no carrousel de objetos DSMCC (exemplo: dir1/dir2/arquivo.txt).

9 API JavaTV 1.1

9.1 Consideração geral

As orientações operacionais para as API referentes à especificação JavaTV são estabelecidas em 9.2 a 9.12.

9.2 Pacote *javax.media, javax.media.protocol e javax.tv.media*

Ver 8.6.

9.3 Pacote *javax.tv.graphics*

O pacote *javax.tv.graphics* deve ser considerado *deprecated* e não pode ser utilizado pelas aplicações Ginga-J.

Pelo fato de a classe *javax.tv.graphics.TVContainer* estar *deprecated*, as aplicações Ginga-J devem utilizar o método *javax.microedition.xle.XletContextt#getContainer()* para obter a referência à instância da classe *java.awt.Container* raiz do *Xlet*.

Para mais detalhes sobre o comportamento gráfico das aplicações Ginga-J, ver 8.4.2.

9.4 Pacote *javax.tv.locator*

Ver 8.14.

Os localizadores transmitidos para o método *javax.tv.locator.Locator#createLocator(String)* devem estar no esquema de algum dos seguintes protocolos suportados:

— *dtv*

— *file*

— *http*

— *https*

No caso do protocolo *dtv*, as seguintes considerações aplicam-se às partes que compõem este tipo de URL:

— *original_network_id:* campo *original_network_id* do *transportstream* informado na tabela NIT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *transport_stream_id:* campo *transport_stream_id* do *transportstream* informado na tabela NIT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *service_id:* campo *program_id* do serviço informado na tabela PAT (ver ABNT NBR 15603-2) em formato hexadecimal;

— *component_tag:* identificador do *elementary stream* informado no *stream_identifier_descriptor* (ver ABNT NBR 15603-2) em formato hexadecimal.

9.5 Pacote *javax.tv.net*

Este pacote não é destinado a ser utilizado por aplicações Ginga-J (de acordo com ABNT NBR 15606-1:2010, Tabela 2, o encapsulamento de datagramas IP não está incluso entre os protocolos de canal de transmissão suportados em Ginga).

9.6 Pacote *javax.tv.service*

9.6.1 Considerações gerais

As informações de serviço disponíveis para as aplicações são restritas ao *transport stream* sintonizado no momento.

As classes deste pacote refletem os dados de informações de serviço sinalizados no *transport stream*. A relação entre os descritores de tabelas estabelecidas na ABNT NBR 15608-3 e as interfaces, classes e métodos do pacote *javax.tv.service* é estabelecida em 9.6.2 a 9.6.4.

9.6.2 Interface *javax.tv.service.RatingDimension*

A Tabela 2 mostra o mapeamento entre os valores de retorno dos métodos da interface *javax.tv.service.RatingDimension* e os campos *parental rating descriptor* presentes na tabela EIT (ver ABNT NBR 15608-3).

Tabela 2 — Mapeamento de *javax.tv.service.RatingDimension*

Método	Campo de descritor
String getDimensionName()	*rating*
String[] getRatingLevelDescription(short ratingLevel)	*rating*

Valores de classificação indicativa no padrão SBTVD

Os valores retornados pelos métodos da interface *javax.tv.service.RatingDimension* devem refletir os requisitos da ABNT NBR 15608-3:2011, 8.3.11:

— O método *getDimensionName()* deve retornar o string *"SBTVD age based rating".*

— O método *getNumberOfLevels()* deve retornar o valor inteiro 7.

— A ABNT NBR 15608-3 estabelece outros oito níveis. Todos eles estão reservados para uso futuro e devem ser ignorados.

— O método *getRatingLevelDescription(short ratingLevel)* deve retornar um vetor com dois elementos. O valor retornado deve estar de acordo com o parâmetro *ratingLevel*. A Tabela 3 mostra o valor retornado para cada um dos níveis de *raiting*.

Tabela 3 — Níveis de *Rating*

ratingLevel	Valor retornado
0	{"0", "Sem classificação etária."}
1	{"L", "Livre para todos os públicos."}
2	{"10", "Não recomendado para menores de 10 anos."}
3	{"12", "Não recomendado para menores de 12 anos."}
4	{"14", "Não recomendado para menores de 14 anos."}
5	{"16", "Não recomendado para menores de 16 anos."}
6	{"18", "Não recomendado para menores de 18 anos."}
7-15	{"Reservado", "Reservado para uso futuro."}

9.6.3 Interface *javax.tv.service.Service*

A Tabela 4 mostra o mapeamento entre os valores de retorno dos métodos da interface *javax.tv.service.Service* e os campos presentes na tabela EIT (ver ABNT NBR 15608-3).

Tabela 4 — Métodos da interface *javax.tv.service.Service*

Método de interface	*Section/Descriptor*	Campo(s)
String getName()	*service descriptor*	*service_name*
Locator getLocator()	*event_information_section*	*original_network_id, transport_stream_id and service_id*

9.6.4 Interface *javax.tv.service.SIElement*

A Tabela 5 mostra o mapeamento entre os métodos da interface *javax.tv.service.SIElement* e os campos presentes na tabela EIT (ver ABNT NBR 15608-3).

Tabela 5 — Métodos da interface *javax.tv.service.SIElement*

Método	Campo(s)
Locator getLocator()	*original_network_id, transport_stream_id and service_id*

9.7 Pacote *javax.tv.service.guide*

9.7.1 Considerações gerais

As informações de evento de programa transmitidas nas tabelas EIT (*current/next* e *schedule* para o *transport stream* atual) (ver ABNT NBR 15608-3) devem estar disponíveis para as aplicações.

Todas as referências de tempo são baseadas na data e hora transmitidas na tabela TOT (ver ABNT NBR 15608-3).

9.7.2 Interface *javax.tv.service.guide.ContentRatingAdvisory*

Os valores retornados por todos os métodos abaixo são derivados do campo *rating* do *parental rating descriptor* presente na tabela EIT (ABNT NBR 15608-3):

— *String[] getDimensionNames()*

— *String getDisplayText()*

— *short getRatingLevel(String dimensionName)*

— *String getRatingLevelText(String dimensionName)*

9.7.3 Interface *javax.tv.service.guide.ProgramEvent*

A Tabela 6 mostra o mapeamento entre os valores retornados pelos métodos da interface *javax.tv.service.guide.ProgramEvent* e os campos presentes na tabela EIT (ver ABNT NBR 15608-3).

Tabela 6 — Métodos da interface *javax.tv.service.guide.ProgramEvent*

Método	Descritor	Campo(s)
long getDuration()		*duration*
Date getEndTime()		*start_time and duration*
String getName()	*short event descriptor*	*text_char*
Date getStartTime()		*start_time*

9.7.4 Interface *javax.tv.service.guide.ProgramEventDescription*

O método *javax.tv.service.guide.ProgramEventDescription#getProgramEventDescription()* retorna o valor do campo *text_char* presente no *short event descriptor* da tablea *EIT* (ver ABNT NBR 15608-3).

9.8 Pacote *javax.tv.service.navigation*

9.8.1 Considerações gerais

As informações de serviço e componente de serviço estão disponíveis para a aplicação de acordo com os dados contidos nas tabelas PAT, PMT, EIT e SDT. Ver ABNT NBR 15603-2 e ABNT NBR 15608-3 para mais detalhes.

O suporte à filtragem utilizando *javax.tv.service.navigation.PreferenceFilter* é opcional, já que as preferências de usuário para todos os critérios possíveis podem não ser fornecidas pelo receptor. Quando não houver suporte, ao aplicar esse filtro a *javax.tv.service.navigation.ServiceList#filterServices()* ou *javax.tv.service.SIManager#filterServices()*, o resultado retornado é uma lista de serviços vazia.

9.8.2 Interface *javax.tv.service.navigation.ServiceComponent*

A Tabela 7 mostra o mapeamento entre os métodos da interface *javax.tv.service.navigation.ServiceComponent* e os campos do *component descriptor* presentes na tabela EIT (ABNT NBR 15603-2).

Tabela 7 — Métodos da interface *javax.tv.service.navigation.ServiceComponent*

Método	Campo de descritor
java.lang.String getAssociatedLanguage()	*ISO_639_language_code*
java.lang.String getName()	*Text*
StreamType getStreamType()	*component_type*

9.8.3 Interface *javax.tv.service.navigation.ServiceDetails*

A Tabela 8 mostra o mapeamento entre os valores retornados pelos métodos da interface *javax.tv.service.navigation.ServiceDetails* e os campos do *service descriptor* presentes na tabela EIT (ABNT NBR 15603-2).

Tabela 8 — Métodos da interface *javax.tv.service.navigation.ServiceDetails*

Método	Campo de descritor
SIRequest retrieveServiceDescription(SIRequestor requestor)	*service_name*
ServiceType getServiceType()	*service_type*
String getLongName()	*service_name*

9.9 Pacote *javax.tv.service.selection*

Ao utilizar *javax.tv.service.selection.ServiceContext#select()*, é possível para uma aplicação Ginga-J alternar entre *serviços* no mesmo *transport stream*. Não é possível alternar para serviços pertencentes a diferentes *transport streams*.

No caso de a aplicação estar sinalizada com o *service_bound_flag* em 1 ou *service_bound_flag em 0,* porém não presente no serviço destino, a aplicação deve ser destruída.

9.10 Pacote *javax.tv.service.transport*

9.10.1 Considerações gerais

As seguintes convenções definem as considerações gerais relacionados às informações sinalizadas nas tabelas NIT, BAT e PMT (ver ABNT NBR 15603-2):

— Bouquet não é operado em ISDB-TB (ver ABNT NBR 15608-3). Por esse motivo, o suporte para as API relacionadas a Bouquet não está disponível para aplicações Ginga-J.

— A informação de rede é restrita ao *transport stream* atual.

— A nomeação de *transport stream* não está disponível em Ginga-J.

9.10.2 Interface de *javax.tv.service.transport.Network*

A Tabela 9 mostra o mapeamento entre os valores retornando pelos métodos da classe *javax.tv.service.transport.Network* e os campos presentes na estrutura *network_information_section* da tabela *EIT*, conforme descrito na ABNT NBR 15603-2.

Tabela 9 — Métodos de *javax.tv.service.transport.Network*

Método	*Section/Descriptor*	Campo
String getName()	*network name descriptor*	*char*
int getNetworkID()		*network_id*

9.11 Pacote *javax.tv.util*

As instâncias de *javax.tv.util.TVTimer* devem utilizar a referência de tempo transmitida na *TOT* (ver ABNT NBR 15603-2) como tempo absoluto.

9.12 Pacote *javax.tv.xlet*

O pacote *javax.tv.xlet* deve ser considerado *deprecated* e não pode ser utilizado por aplicações Ginga-J. Alternativamente, as aplicações devem utilizar o pacote *javax.microedition.xlet*.

10 API JavaDTV 1.3

10.1 Considerações gerais

Esta seção especifica as orientações operacionais para as API referentes às ABNT NBR 15606-4 e ABNT NBR 15606-6.

Os requisitos da ABNT NBR 15606-4:2010, Anexo A, devem também ser considerados.

10.2 Pacote *com.sun.dtv.application*

Os requisitos da ABNT NBR 15606-4:2010, A.2.9, devem ser considerados.

10.3 Pacote *com.sun.dtv.broadcast*

Os requisitos da ABNT NBR 15606-4:2010, A.2.1, devem ser considerados.

10.4 Pacote *com.sun.dtv.broadcast.event*

As condições a seguir definem o comportamento esperado da classe *com.sun.dtv.broadcast.BroadcastEventManager*:

— As instâncias de *Locators* passadas para os construtores da classe *com.sun.dtv.broadcast.BroadcastEventManager* devem utilizar o esquema a seguir, a fim de fazer referência a *elementary streams* que contenham *stream events*:

  — *dtv://</original_network_id>.<transport_stream_id>.<service_id>/<component_tag>/*

10.5 Pacote *com.sun.dtv.filtering*

Os requisitos da ABNT NBR 15606-4:2010, A.2.4, devem ser considerados.

10.6 Pacote *com.sun.dtv.io*

Os requisitos da ABNT NBR 15606-4:2010, A.2.27, devem ser considerados.

10.7 Pacote *com.sun.dtv.locator*

Os requisitos da ABNT NBR 15606-4:2010, A.2.21, devem ser considerados.

10.8 Pacote *com.sun.dtv.lwuit* .*

As condições a seguir definem o comportamento esperado da classe *com.sun.dtv.lwuit.Style*:

— Por padrão, as imagens de fundo não são escalonadas pela classe *com.sun.dtv.lwuit.Style*. Assim, o método *Style#isScaledImage()* retorna *false*.

— O tamanho da fonte-padrão das fontes retornado por *com.sun.dtv.lwuit.Font* é 14 pontos.

— O estilo da fonte-padrão retornado por *com.sun.dtv.lwuit.Font* é PLAIN.

— O nome da fonte-padrão retornado por *com.sun.dtv.lwuit.Font* é *"default"*.

10.9 Pacotes *com.sun.dtv.media* .*

Os requisitos da ABNT NBR 15606-4:2010, A.2.7, A.2.8, A.2.10, A.2.17, A.2.18, A.2.24, A.2.25 e A.2.30, devem ser considerados.

10.10 Pacote *com.sun.dtv.net*

Os requisitos da ABNT NBR 15606-4:2010, A.2.23, devem ser considerados.

10.11 Pacote *com.sun.dtv.platform*

Os requisitos da ABNT NBR 15606-4:2010, A.2.26, devem ser considerados.

10.12 Pacote *com.sun.dtv.resources*

Os requisitos da ABNT NBR 15606-4:2010, A.2.22, devem ser considerados.

10.13 Pacote *com.sun.dtv.security*

Os requisitos da ABNT NBR 15606-4:2010, A.2.19, devem ser considerados.

10.14 Pacote *com.sun.dtv.service*

A condição a seguir define o comportamento esperado da classe *com.sun.dtv.service.SIDatabase*:

— O método *getAllSIDatabases()* deve retornar um vetor de instâncias do tipo *br.org.sbtvd.si.SIDatabase*.

10.15 Pacote *com.sun.dtv.smartcard*

A condição a seguir define o comportamento esperado da classe *com.sun.dtv.smartcard.CardTerminal*:

— Todos os métodos da classe *com.sun.dtv.smartcard.CardTerminal* retornam valores *null* em receptores sem qualquer suporte à funcionalidade de *SmartCards*.

10.16 Pacote *com.sun.dtv.test*

A condição a seguir define o comportamento esperado da classe *com.sun.dtv.test.TestHarness*:

— Por padrão, o campo público *log* deve ser definido a *System.out java.io.PrintStream*.

10.17 Pacote *com.sun.dtv.transport*

Os requisitos da ABNT NBR 15606-4:2010, A.2.31, devem ser considerados.

10.18 Pacote *com.sun.dtv.tuner*

As condições a seguir definem o comportamento esperado da classe *com.sun.dtv.tuner.Tuner*:

— Considerando que não é possível a sintonização para aplicações Ginga-J, os métodos *tune(javax.tv.locator.Locator)* e *tune(com.sun.dtv.transport.TransportStream)* lançam sempre *com.sun.dtv.tuner.TuningException*.

10.19 Pacote *com.sun.dtv.ui*

As condições a seguir definem o comportamento esperado da classe *com.sun.dtv.ui.DTVContainer*:

— O método *com.sun.dtv.ui.DTVContainer#getCurrentDTVContainer()* retorna a instância *DTVContainer* associada ao Plano Gráfico.

— O método *com.sun.dtv.ui.DTVContainer#getDTVContainer(com.sun.dtv.ui.Plane, java.awt.Container)* retorna a instância *DTVContainer* associada ao plano gráfico.

— O método *com.sun.dtv.ui.DTVContainer#getDTVContainer(com.sun.dtv.ui.Plane, java.awt.Container)* lança um *java.lang.IllegalArgumentException* se o primeiro parâmetro não for uma instância do Plano Gráfico ou se o segundo parâmetro não for a raiz *Xlet java.awt.Container* retornada pelo método *XletContext#getContainer()*.

As condições a seguir definem o comportamento esperado da classe *com.sun.dtv.ui.DTVContainerPattern*:

— O método *com.sun.dtv.ui.DTVContainerPattern#getPreferenceValue(int preference)* retorna o valor atualmente válido para a preferência especificada. Isto tem que ser un objeto *PlaneSetup* para *DTV_CONTAINER_PLANE_SETUP*, um objeto *Dimension* para *DTV_CONTAINER_DIMENSION* e um objeto *Point* para *DTV_CONTAINER_LOCATION*.

Para mais definições, ver 8.4.2.

10.20 Pacote *com.sun.dtv.ui.event*

Os requisitos da ABNT NBR 15606-4:2010, A.2.5, devem ser considerados.

11 API definido em Ginga-J: API de informações de serviço dependente de protocolo

Os requisitos da ABNT NBR 15606-4:2010, Anexo B, devem ser considerados além do estabelecido em 8.10.

A interface *br.org.sbtvd.si.SIIterator* deve estender a interface *java.util.Enumeration*.

12 API definido em Ginga-J: API para sintonização estendida (*br.org.sbtvd.net.tuning*)

Os requisitos da ABNT NBR 15606-4:2010, Anexo B, devem ser considerados.

Esta API não se destina a ser utilizada pelas aplicações Ginga-J baixadas a partir do sinal de *broadcast* ou a partir do canal de interatividade. O receptor não pode conceder permissões para que esses tipos de aplicações Ginga-J sintonizem um *transport stream* diferente (canal de frequência diferente).

13 API definido em Ginga-J: API Ponte NCL (*br.org.sbtvd.bridge*)

Os requisitos da ABNT NBR 15606-4:2010, Anexo D, devem ser considerados além do estabelecido em 8.10.

14 API definida em Ginga-J: API para suporte a planos de tela (*br.org.sbtvd.ui*)

Os requisitos da ABNT NBR 15606-4:2010, Anexo E, devem ser considerados além do estabelecido em 8.4.3.

Bibliografia

[1] ABNT NBR 15603-1, *Televisão digital terrestre – Multiplexação e serviços de informação (SI) – Parte 1: SI do sistema de radiodifusão*

[2] ABNT NBR 15606-7, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 7: Ginga-NCL – Diretrizes operacionais para as ABNT NBR 15606-2 e ABNT NBR 15606-5*

[3] CDC 1.1:2008, *Connected Device Configuration 1.1 (JSR 218), disponível em http://jcp.org/en/jsr/detail?id=218*

[4] FP 1.1:2008, *Foundation Profile 1.1 (JSR 219), disponível em http://jcp.org/en/jsr/detail?id=219*

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

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

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

[8] JAVADTV 1.3:2009, *Java DTV Specification, Sun Microsystems*

[9] *Programming in Lua, Roberto Ierusalimsky, 2003.*