Televisão digital terrestre - Codificação de dados e especificações de transmissão para transmissão digital – Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais

APRESENTAÇÃO

1) Este 1º Projeto foi elaborado pela ABNT/CEET-00:001.85 - Comissão de Estudo Especial Temporária de Televisão Digital, nas reuniões de:

09/04/2007	16/04/2007	02/05/2007
07/05/2007	15/05/2007	21/05/2007
04/06/2007	11/06/2007	18/06/2007
25/06/2007	02/07/2007	11/07/2007
16/07/2007	30/07/2007	06/08/2007
13/08/2007	22/08/2007	03/09/2007

2) Não tem valor normativo;

3) Aqueles que tiverem conhecimento de qualquer direito de patente devem apresentar esta informação em seus comentários, com documentação comprobatória;

4) Tomaram parte na elaboração deste Projeto:

Participante	Representante
CCE/DIGIBRAS	Walter Lervolino Junior
CESAR	Mário Fried
CESAR	Carlos André Guimarães Ferraz
CESAR	Paulyne Matthews Jucá
CESAR	Andrino Soares de Souza Coêlho
CESAR	Luiz Augusto Zelaquett de Souza
CESAR	Paulo Martinelli Hemmlepp
FUNTEC/AM	Wilkens de Figueiredo
GENIUS/GRADIENTE	Aguinaldo Silva
HIRIX	Zalkind Lincoln
INMETRO	Rodolfo Saboia L. Sousa
INMETRO	Alexsandro Nogueira Reis
INMETRO	Luiz da Silva Mello

LAVID/UFPB	Guido Lemos de Souza Filho
LAVID/UFPB	Luiz Eduardo Cunha Leite
LAVID/UFPB	Carlos Eduardo Coelho Freire Batista
LAVID/UFPB	Giuliano Maia Lins de Castro
LAVID/UFPB	Gilberto Farias de Sousa Filho
LAVID/UFPB	Lincoln David de Nery e Silva
LAVID/UFPB	Tiago Maritan Ugulino de Araujo
LAVID/UFPB	Erick Augusto Gomes de Melo
LAVID/UFPB	Jefferson Ferreira de Araujo Lima
LSI/USP	Marcelo Zuffo
LSI/USP	Eduardo Rodrigues de Carvalho
PUC/RIO	Marcio Ferreira Moreno
PUC-RIO	Luiz Fernando Gomes Soares
PUC-RIO	Rogério Ferreira Rodrigues
PUC-RIO	Marcelo Moreno
PUC-RIO	Romualdo Rezende Costa
PUC-RIO	Carlos Salles Soares Netto
PUC-RIO	Rafael Ferreira Rodrigues
PUC-RIO	Rodrigo Layola
RF/TELAVO	Gabriel Hideki Waoa
SAMSUNG	Domingos K. Stavridis
SAMSUNG	Fábio R. Campanhã
SBT	José Olairson Valentim
SET	Olímpio J. Franco
TV BANDEIRANTES	Lyzbeth Cronembold
TV GLOBO	Carlos Fini
TV GLOBO	Paulo Henrique Corona Viveiros de Castro
TV GLOBO	Cleveland Albuquerque
TV GLOBO	Fábio Henrique Lemos de Castro
TV GLOBO	Isabelle Desbois
TV GLOBO	Eduardo Chimati Giannotto
TV GLOBO	Carolina Duca Novaes

TV GLOBO	Daniel Lourenço Domingos
UNICAMP	Luis Geraldo Meloni
UNICAMP	Rodrigo Cascão Araujo
UNICAMP	Davi Trindade dos Santos
UNICAMP	Douglas Terêncio do Vale

Televisão digital terrestre - Codificação de dados e especificações de transmissão para transmissão digital – Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais

Digital terrestrial television – Data codification and transmission specifications for digital broadcasting - Part 4: Ginga-J - The environment for the execution of procedural applications

Palavras-chave: Televisão digital terrestre. *Middleware.* Aplicações procedurais.
Descriptors: Digital terrestrial television. Middleware. Procedural applications.

Sumário

Prefácio
Introdução
1 Escopo
2 Referências normativas
3 Arquitetura do *middleware* Ginga
3.1 Visão geral da arquitetura Ginga
3.2 Arquitetura Ginga-J
3.2.1 Contexto
3.2.2 Arquitetura
4 Formato do conteúdo
5 Ginga-J – Aplicativos e ambiente
5.1 Compatibilidade com especificações internacionais
5.2 Comportamento dos aplicativos
5.3 Plataforma Java
5.4 Modelo de aplicativo
5.5 Sinalização de aplicações
5.5.1 Compatibilidade com normas internacionais
5.5.2 Exigências específicas Ginga
5.5.2.1 Sintaxe da AIT
5.5.2.2 Controle dinâmico do ciclo de vida do aplicativo
5.5.2.3 Transporte através do canal de interação
6 Ginga-J API
6.1 API aderentes à GEM DVB-J
6.1.1 Pacotes da API JavaTV
6.1.2 Pacotes DAVIC
6.1.3 Pacotes HAVi
6.1.4 Pacotes DVB
6.2 Extensões Ginga
6.2.1 API de controle de sintonizador

6.2.2 API de fluxos de mídia

6.2.3 API de apresentação

6.2.3.1 Especificação de pacotes componentes

6.2.3.2 Interface SBTVDVideoFormatControl

6.2.3.3 Classe AudioLanguageEvent

6.2.3.4 Classe AudioLanguageEventListener

6.2.4 API de canal de retorno

6.2.4.1 Especificação dos pacotes componentes

6.2.4.2 Classe RCMessage

6.2.4.3 Classe RCAsynchronous

6.2.4.4 Classe AsynchronousMessageTable

6.2.5 Configurações do usuário e API de preferências

6.2.6 Perfil e propriedades da versão

6.3 API aderentes à especificação ARIB STD B-23

6.3.1 Informações do serviço

6.3.2 API de informações de serviço independente de protocolo

6.4 Definições Ginga-J

6.4.1 API de integração de dispositivos

6.4.1.1 Especificação dos pacotes componentes

6.4.1.2 Classe GRemoteDeviceManager

6.4.1.3 Classe GRemoteDevice

6.4.1.4 Interface GRemoteDeviceActionListener

6.4.1.5 Classe GRemoteEvent

6.4.1.6 Classe GRemoteKeyEvent

6.4.1.7 Classe GRemoteUserEvent

6.4.2 API de ponte Ginga-NCL

6.4.2.1 Especificação dos pacotes componentes

6.4.2.2 Classe NCLBaseElement

6.4.2.3 Classe NCLCompositeNode

6.4.2.4 Classe NCLConnector

6.4.2.5 Classe NCLConnectorBase

6.4.2.6 Classe NCLDescriptor

6.4.2.7 Classe NCLDescriptorBase

6.4.2.8 Classe NCLDescrSwitch

6.4.2.9 Classe NCLDocument

6.4.2.10 Classe NCLEvent

6.4.2.11 Interface NCLEventListener

6.4.2.12 Classe NCLGingaSettingsNodes

6.4.2.13 Classe NCLImportBase

6.4.2.14 Classe NCLImportedDocumentBase

6.4.2.15 Classe NCLImportNCL

6.4.2.16 Classe NCLInterface

6.4.2.17 Classe NCLLink

6.4.2.18 Classe NCLMediaNode

6.4.2.19 Classe NCLNode

6.4.2.20 Classe NCLPrivateBase

6.4.2.21 Classe NCLRegion

6.4.2.22 Classe NCLRegionBase

6.4.2.23 Classe NCLRule

6.4.2.24 Classe NCLRuleBase

6.4.2.25 Classe NCLTransition

6.4.2.26 Classe NCLTransitionBase

6.4.2.27 Classe PrivateBaseManager

6.5 Lista completa de API Java

Anexo A (informativo) Harmonização com especificações internacionais

Bibliografia

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 Temporárias (ABNT/CEET), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidades, laboratórios e outros).

Os Projetos de Norma Brasileira, elaborados no âmbito dos ABNT/CB e ABNT/ONS, circulam para Consulta Nacional entre os associados da ABNT e demais interessados.

Introdução

A definição Ginga-J é composta por API (Interfaces de Programação de Aplicativos) 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 transmição 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 (HAN).

Esta Norma 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 Norma tem como objetivo garantir a interoperabilidade dos aplicativos Ginga e diferentes implementações Ginga.

Esta Norma é harmozinada com especificações internacionais, conforme detalhado no Anexo A.

1 Escopo

Esta parte do Projeto 00:001.85-006 especifica os requisitos para a parte procedural do *middleware* para o sistema brasileiro de televisão digital terrestre (SBTVD).

2 Referências normativas

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

Projeto 00:001.85-006/1, *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiofusão digital – Parte 1: Codificação de dados*

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

GEM 1.1:2007, *Globally executable MHP (GEM) specification*

3 Arquitetura do *middleware* Ginga

3.1 Visão geral da arquitetura Ginga

O universo das aplicações para televisão digital pode ser particionado em dois conjuntos: o das aplicações declarativas e o das aplicações procedurais. Uma aplicação declarativa é aquela em que sua entidade “inicial” é do tipo “conteúdo declarativo”. Analogamente, uma aplicação procedural é aquela em que sua entidade “inicial” é do tipo “conteúdo procedural”.

Um conteúdo declarativo deve ser baseado em uma linguagem declarativa, isto é, em uma linguagem que enfatiza a descrição declarativa do problema, ao invés da sua decomposição em uma implementação algorítmica. Um conteúdo procedural deve ser baseado em uma linguagem não declarativa. Linguagens não declarativas podem seguir diferentes paradigmas. Tem-se assim, as linguagens baseadas em módulos, orientadas a objetos etc. A literatura sobre televisão digital, no entanto, utiliza o termo procedural para representar todas as linguagens que não são declarativas. Numa programação procedural, o computador deve obrigatoriamente ser informado sobre cada passo a ser executado. Pode-se afirmar que, em linguagens procedurais, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa - como existem mais recursos disponíveis o grau de complexidade é maior. A linguagem mais usual encontrada nos ambientes procedurais de um sistema de TV digital é Java.

O Ginga-NCL (ou Máquina de Apresentação) é um subsistema lógico do Sistema Ginga que processa documentos NCL. Um componente-chave do Ginga-NCL é o mecanismo de decodificação do conteúdo informativo (NCL *formatter*). Outros módulos importantes são o usuário baseado em XHTML, que inclui uma linguagem de estilo (CSS) e intérprete ECMAScript, e o mecanismo LUA, que é responsável pela interpretação dos *scripts* LUA.

O Ginga-J (ou Máquina de Execução) é um subsistema lógico do Sistema Ginga que processa aplicações procedurais (Xlets Java). Um componente-chave do ambiente do aplicativo procedural é o mecanismo de execução do conteúdo procedural, que tem por base uma Máquina Virtual Java.

Decodificadores comuns de conteúdo devem servir para as necessidades de aplicativos tanto procedurais quanto informativos de decodificação e apresentação de conteúdos comuns do tipo PNG, JPEG, MPEG e outros formatos. O Ginga-Core é composto por decodificadores e procedimentos comuns de conteúdo para obter conteúdos transportados em fluxos de transporte MPEG-2 (TS) e através de um canal de retorno. O Ginga-Core também deve suportar o modelo de exibição conceitual descrito no Projeto 00:001.85-006/1.

A arquitetura (ver Figura 1) e as facilidades da especificação Ginga devem ser destinadas à aplicação em sistemas e receptores de transmissão para transmissão terrestre (*over-the-air*). Além disso, a mesma arquitetura, e facilidades, podem ser aplicadas a outros sistemas de transporte (como sistemas de televisão via satélite ou cabo).

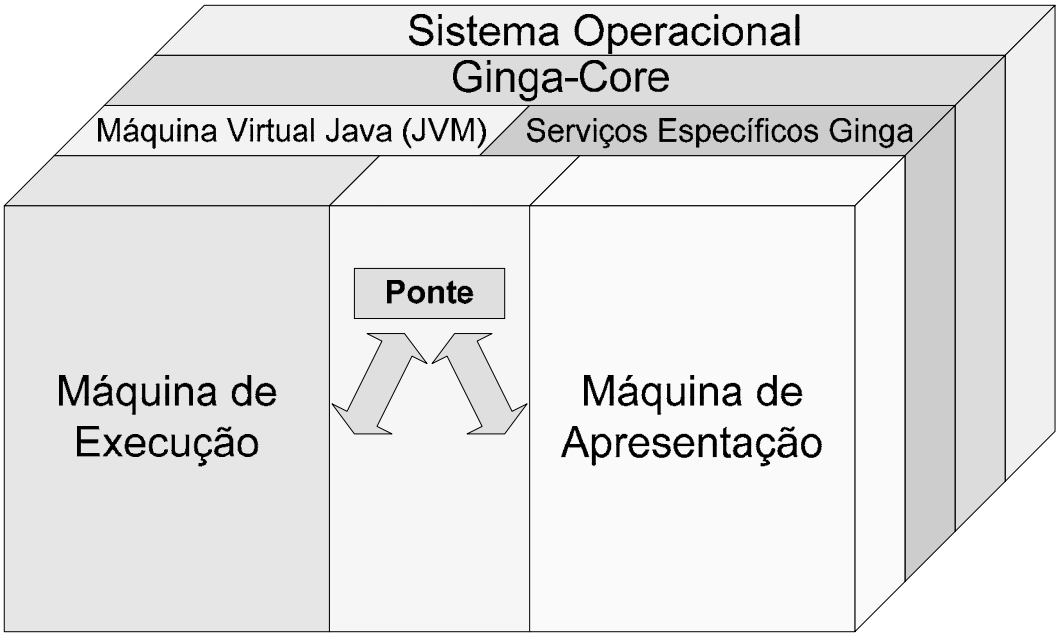

Figura 1 - Arquitetura em alto nível do *middleware* Ginga

3.2 Arquitetura Ginga-J

3.2.1 Contexto

A Figura 2 apresenta o contexto em que a pilha do *software* Ginga-J é executada. O Ginga-J é uma especificação de *middleware* distribuído, que reside em um dispositivo Ginga (dispositivo que embarque o *middleware* Ginga – um receptor de televisão digital), com possibilidade de possuir componentes de software nos dispositivos de interação (celulares, PDA etc).

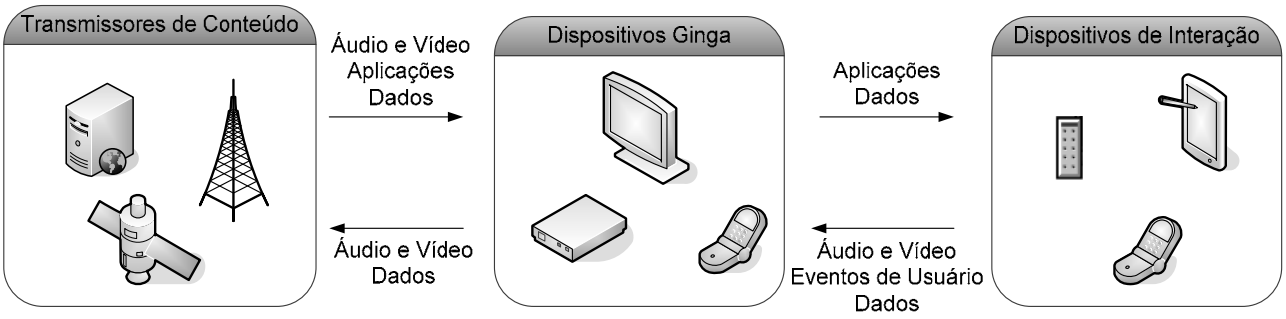

Figura 2 - Contexto do Ginga-J

O dispositivo Ginga deve ter acesso a fluxos de vídeo, áudio, dados e outros recursos de mídia, que devem ser transmitidos através do ar, cabo, satélite ou através de redes IP. As informações recebidas devem ser processadas e apresentadas aos telespectadores.

O telespectador pode interagir com o dispositivo Ginga através de dispositivos de interação que podem conter componentes de *software* Ginga de forma que o dispositivo de interação possa enviar informações para o dispositivo Ginga utilizando as funcionalidades providas na especificação Ginga, ou seja, estes componentes de *software* que podem ser instalados nos dispositivos de interação permitem que as funcionalidades dos mesmos sejam exploradas, utilizando funcionalidades da API Ginga-J, por aplicações nos dispositivos Ginga (receptores de televisão digital). Para que um dispositivo de interação possa ser utilizado, ele deve estar registrado com o dispositivo Ginga, e durante esse processo o dispositivo de interação pode receber o componente de *software* necessário para viabilizar a comunicação com o dispositivo Ginga.

Como resposta à informação enviada pelo telespectador, o dispositivo de Ginga deve apresentar a saída de vídeo e áudio utilizando seu próprio monitor e auto-falantes ou os dos dispositivos de interação. Um único dispositivo pode ter capacidade de entrada e saída simultâneas.

EXEMPLO Um dispositivo de interação pode ser um PDA conectado à plataforma Ginga através de uma rede sem fio. Utilizando tal dispositivo de interação, um telespectador pode enviar comandos (eventos de usuário) à plataforma através do teclado PDA e os aplicativos da plataforma podem enviar conteúdo visual para ser apresentado na tela do PDA.

Um dispositivo de interação pode também ter capacidades de captura e reprodução de som, de forma que o telespectador possa enviar fluxos de áudio e vídeo para o dispositivo Ginga, utilizando os dispositivos de interação que dêem suporte à essa funcionalidade.

Vários telespectadores podem interagir com a plataforma Ginga simultaneamente. Nesse caso, cada telespectador pode ter um dispositivo de interação e a plataforma deve distinguir os comandos enviados por e para cada dispositivo. O dispositivo Ginga pode também enviar informações para os transmissores de conteúdo quando da existência de um canal de retorno (conexão com a Internet, por exemplo).

3.2.2 Arquitetura

O modelo Ginga-J distingue entre as entidades e recursos de *hardware*, *software* do sistema e aplicativos conforme descrito na Figura 3.

As aplicações residentes podem ser implementadas usando funções não padronizadas, fornecidas pelo Sistema Operacional do dispositivo de Ginga, ou por uma implementação particular do Ginga. Os aplicativos residentes também podem incorporar funcionalidades providas pelas API padronizadas Ginga-J. Aplicativos transmitidos (*Xlets*) sempre devem utilizar API padronizadas fornecidas pelo Ginga-J.

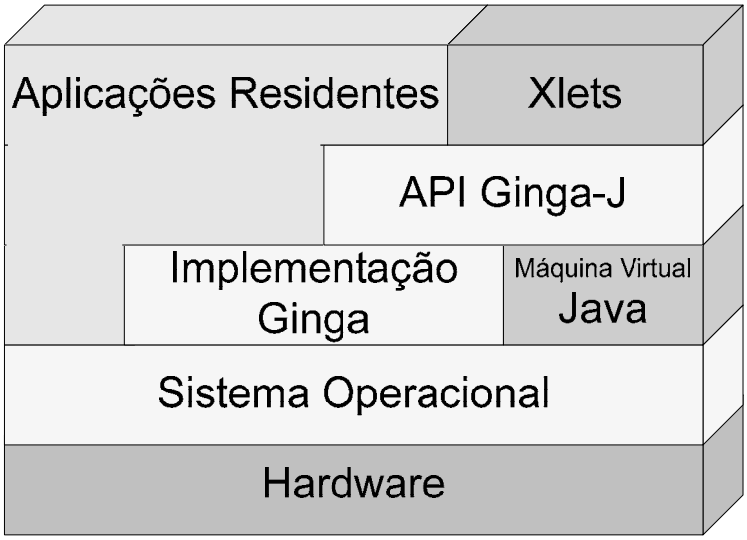

Figura 3 - Arquitetura Ginga-J e ambiente de execução

Em geral, o Ginga é alheio a quaisquer aplicativos residentes. Estas aplicações residentes incluem, mas não limitam-se a: *closed caption*, mensagens do sistema de acesso condicional (*Conditional* Access – CA), menus do receptor e guias de programação eletrônica (*Eletronic Program Guide* – EPG) residente.

Os aplicativos residentes podem ter prioridade sobre os aplicativos Ginga. Como exemplo, o *closed caption* e mensagem de emergência devem ter prioridade no Sistema Ginga.

4 Formato do conteúdo

O formato do conteúdo para o sistema brasileiro de televisão digital terrestre deve estar de acordo com o Projeto 00:001.85-006/1.

5 Ginga-J – Aplicativos e ambiente

5.1 Compatibilidade com especificações internacionais

A definição de um aplicativo e ambiente Ginga-J deve ser baseada nas definições estabelecidadas na GEM 1.1, de forma que uma aplicação em conformidade com o GEM deve ser como conseqüência, ser considerada uma "aplicação compatível com Ginga-J".

5.2 Comportamento dos aplicativos

A descrição de um comportamento de aplicativo especificado na GEM 1.1:2007, seção 9.1, deve se aplicar aos aplicativos Ginga-J e implementações de um ambiente Ginga-J.

5.3 Plataforma Java

O ambiente do aplicativo de procedimento Ginga-J deve possuir uma Máquina Virtual Java (*Java Virtual* Machine – JVM) e deve seguir o estabelecido na GEM 1.1:2007, seções 11.1 e 11.2.

5.4 Modelo de aplicativo

Para promover a compatibilidade com outras especificações de sistemas de *middleware*, o Modelo de Aplicativo Ginga-J deve incorporar incorpora tecnologias descritas na GEM 1.1:2007, seção 9.

5.5 Sinalização de aplicações

5.5.1 Compatibilidade com normas internacionais

O GINGA-J deve adotar a definição de sinalização de aplicação funcional equivalente conforme descrito na GEM 1.1:2007, seção 10. Ao ler-se a especificação GEM 1.1:2007 a respeito da correspondência Ginga, o termo "DVB" ou "GEM" deve ser lido como "Ginga" e o termo "DVB-J" deve ser lido como “Ginga-J”.

5.5.2 Exigências específicas Ginga

5.5.2.1 Sintaxe da AIT

Ginga deve definir que os tipos utilizados para suas aplicações devem ser modificados, com os valores 0x0008 para aplicativo Ginga-J e 0x0009 para aplicativo Ginga-NCL, conforme demonstrado na Tabela 1.

Tabela 1 – Tipo de aplicação

Tipo de aplicativo	Descrição
0x00	Reservado
0x0008 [a]	Ginga-J
0x0009 [a]	Ginga-NCL
[a] Valores ainda passíveis de registro.	

5.5.2.2 Controle dinâmico do ciclo de vida do aplicativo

O controle dinâmico do ciclo de vida do aplicativo deve seguir as definições constantes na GEM 1.1:2007, seção 10.4.3. As definições da GEM 1.1:2007, seção 10.6.2, devem ser estendidas de forma que os códigos de controle de aplicações utilizado pelo Ginga-J sejam os que são apresentados na Tabela 2.

Tabela 2 – Valores do código de controle do aplicativo Ginga-J

Código	Identificador	Semântica
0x00	-	Reservado para uso futuro
0x01	*AUTOSTART*	Aplicativos com o código de controle *AUTOSTART* devem ser iniciados automaticamente quando o receptor muda para este serviço
0x02	*PRESENT*	Os aplicativos com código de controle *PRESENT* não devem ser iniciados automaticamente, mas devem ser adicionados à lista de aplicativos disponíveis do receptor. O usuário pode então escolher iniciar este aplicativo selecionando-o na lista
0x03	*DESTROY*	Os aplicativos com código de controle definidos para *DESTROY* devem ser automaticamente eliminados pelo receptor
0x04	*KILL*	Os aplicativos com código de controle definidos para *KILL* devem ser automaticamente eliminados pelo receptor. A diferença desse código para o código de controle D*ESTROY* é que um aplicativo com o código de controle *KILL* pode ter opção de garantia de continuidade de operação se o aplicativo escolher
0x05	-	Reservado para uso futuro
0x06	*REMOTE*	Os aplicativos com código de controle definidos para *REMOTE* devem identificar um aplicativo remoto que somente é lançado após a seleção do serviço
0x07	*UNBOUND*	Aplicações com código de controle *UNBOUND* devem ser similares aos aplicativos *PRESENT*. A diferença do *PRESENT* é que o receptor pergunta ao usuário se o aplicativo deve ser armazenado para execução posterior
0x08…0xFF	-	Reservado para uso futuro

5.5.2.3 Transporte através do canal de interação

O transporte através do canal de interação deve estar de acordo com a GEM 1.1:2007, seção 10.

6 Ginga-J API

6.1 API aderentes à GEM DVB-J

6.1.1 Pacotes da API JavaTV

A API Java TV é uma extensão para a plataforma Java para sustentar a produção de conteúdo interativo de forma procedural para a televisão digital, com base nas tecnologias Java. Basicamente, a API Java TV oferece os mesmos mecanismos de controle e acesso dos receptores de televisão digital, escondendo alguns aspectos do Sistema Operacional e detalhes de *hardware*. A principal finalidade da API Java TV é fornecer um conjunto de métodos, classes e interfaces para facilitar a construção de aplicativos destinados a serem executados através de plataformas de recepção de televisão digital independentes das tecnologias utilizadas na rede de transmissão.

A seguir são listados os pacotes que são parte da API Java TV que são incluídos no Ginga-J:

- java.awt
- java.awt.event
- java.awt.image
- java.beans
- java.io
- java.lang
- java.lang.reflect
- java.net
- java.security
- java.security.cert
- java.util
- java.util.zip
- javax.media
- javax.media.protocol
- javax.tv.graphics
- javax.tv.locator
- javax.tv.media
- javax.tv.media.protocol
- javax.tv.net
- javax.tv.service
- javax.tv.service.guide
- javax.tv.service.navigation
- javax.tv.service.selection
- javax.tv.service.transport
- javax.tv.util
- javax.tv.xlet
- java.math
- java.rmi
- java.security.spec
- javax.net
- javax.net.ssl
- javax.security.cert

6.1.2 Pacotes DAVIC

O sistema DAVIC (Conselho de Áudio e Vídeo Digital) é um conjunto de especificações que têm como principal objetivo sustentar uma interoperabilidade real de ponta a ponta para as plataformas envolvidas na execução de serviços de áudio e vídeo transmitidos via radiodifusão. Assim, esses padrões podem ser utilizados nos sistemas

de televisão digital para fornecer conteúdo ao usuário final e também para permitir interatividade com o mesmo usuário.

O sistema DAVIC tem suas próprias API, e portanto ele também pode ser considerado como um *middleware* de alto nível (apesar de ser comum para ele operar em conjunto com um middleware de nível mais baixo).

A seguir são listados os pacotes que são parte da API DAVIC incluídos no Ginga-J:

- org.davic.media
- org.davic.resources
- org.davic.mpeg
- org.davic.mpeg.sections
- org.davic.net
- org.davic.net.dvb
- org.davic.net.tuning

6.1.3 Pacotes HAVi

As definições HAVi (interoperabilidade doméstica de áudio e vídeo) determinam uma rede doméstica padrão focada na interoperatividade de dispositivos de áudio e vídeo, de forma que todos os dispositivos de áudio e vídeo componentes de uma rede HAVi possam interagir uns com os outros.

O principal objetivo de uma interface de usuário em uma rede doméstica é oferecer um ambiente operacional fácil de usar. A arquitetura HAVi permite que os usuários controlem dispositivos de forma familiar, utilizando uma tela frontal ou através de um controle remoto. A interface de usuário HAVi Nível 2 API é um subconjunto Java AWT 1.1 com algumas extensões.

A seguir são listados os pacotes que são parte do HAVi API incluídos no Ginga-J API:

- org.havi.ui
- org.havi.ui.event

6.1.4 Pacotes DVB

Ao desenvolver o *middleware* padrão MHP, o DVB incluiu alguns pacotes para extender as funcionalidades oferecidas pelo JavaTV, HAVi e DAVIC. Essas funcionalidades incluíram API de Informações de Serviço, Intercomunicação entre Xlets, persistência etc. Algumas das funcionalidades também foram incluídas na GEM. Os pacotes DVB inclusos na GEM e então aplicados às especificações Ginga são descritos na GEM 1.1:2007, Anexos J, L, N, P, R, S, U e W, e são listados abaixo:

- org.dvb.application
- org.dvb.dsmcc
- org.dvb.event
- org.dvb.io.ixc
- org.dvb.io.persistent
- org.dvb.lang
- org.dvb.media
- org.dvb.net

- org.dvb.net.tuning
- org.dvb.net.rc
- org.dvb.test
- org.dvb.ui
- org.dvb.user

6.2 Extensões Ginga

6.2.1 API de controle de sintonizador

A API br.org.sbtvd.net.tuning é uma extensão da API GEM org.davic.net.tuning, as novas funções são *zapping* de canais de forma interativa e varredura em todas as interfaces da rede existentes em um receptor.

O seguinte pacote compõe essa API:

- br.org.sbtvd.net.tuning

A classe *ChannelManager* especifica um objeto responsável pela atividade de *zapping* (troca) de canais sintonizáveis através da interface de rede (terrestre, cabo, satélite, IPTV) existente no receptor de televisão digital.

Os métodos públicos são os seguintes:

- void tuneChannel (int num, org.davic.net.tuning.NetworkInterfaceListener lis) throws StreamNotFoundException
  - Sintoniza assincronicamente o canal fornecido pelo parâmetro inteiro *num*. Este médoto pode lançar uma exceção do tipo *StreamNotFoundException*.
- void tuneNextChannel (org.davic.net.tuning.NetworkInterfaceListener lis) throws StreamNotFoundException
  - Sintoniza assincronicamente o próximo canal da tabela *TransportStream* gerada durante a varredura. Este médoto pode lançar uma exceção do tipo *StreamNotFoundException*.
- void tunePreviousChannel (org.davic.net.tuning.NetworkInterfaceListener lis) throws StreamNotFoundException
  - Sintoniza assincronicamente o canal anterior da tabela *TransportStream* gerada durante a varredura. Este médoto pode lançar uma exceção do tipo *StreamNotFoundException*.
- int getNumberofChannels ()
  - Retorna o número de canais encontrados em todas as interfaces de rede disponíveis no sistema.
- static ChannelManager getInstance ()
  - Retorna um objeto (instância) *ChannelManager*.

6.2.2 API de fluxos de mídia

A API de fluxos de mídia adota a *Java Media Framework* (JMF) 2.1, ao invés da *Java Media Framework* (JMF) 1.0, definida pela GEM 1.1:2007, Seção 11.4.2. A JMF 2.1 é compatível com JMF 1.0, portanto, a *Streamed Media API* Ginga-J é compatível com a *Streamed Media API* GEM 1.1:2007.

A API JMF 2.1 estende o *framework* JMF 1.0 para fornecer suporte à captura e armazenamento de dados, controlando o tipo de processamento que é efeito durante a reprodução, e desempenhando processo personalizado em feixes de dados de mídia. Além disso, a JMF 2.1 define uma API *plug-in* que permite aos desenvolvedores avançados e provedores de tecnologia personalizar de forma mais fácil e estender a funcionalidade JMF.

Os seguintes pacotes da JMF 2.1 foram incluídos:

- javax.media
- javax.media.bean.playerbean
- javax.media.cdm
- javax.media.control
- javax.media.datasink
- javax.media.format
- javax.media.pim
- javax.media.pm
- javax.media.protocol
- javax.media.renderer
- javax.media.rtp
- javax.media.rtp.event
- javax.media.rtp.rtcp
- javax.media.util

6.2.3 API de apresentação

6.2.3.1 Especificação de pacotes componentes

A API de apresentação é uma extensão do pacote *org.dvb.media*, que é parte da API GEM 1.1:2007.

As extensões são destinadas a dar suporte às especificações de fluxo de vídeo definidas para o Sistema Brasileiro de TV Digital.

O seguinte pacote compõe a API de apresentação:

- br.org.sbtvd.media

6.2.3.2 Interface SBTVDVideoFormatControl

A interface SBTVDVideoFormatControl estende a interface *org.dvb.media.VideoFormatControl*, descrita na API GEM 1.1:2007.

A interface deve fornecer um método de adquirir informações especificadas pelo Sistema Brasileiro de TV Digital relevantes ao vídeo atualmente exibido.

Seus métodos públicos são os seguintes:

- java.awt.Dimension getDisplayVideoSize()
  - Este médodo retorna o tamanho da área de exibição especificada por *sequence_display_extension*.
- int getProgressiveSequence()
  - Este método retorna a *progressive_sequence* especificada pela *sequence_extension*.
- java.awt.Dimension getSourceVideoSize()
  - Este médodo retorna o tamanho da fonte de vídeo especificada por *sequence_header*.

6.2.3.3 Classe AudioLanguageEvent

A classe AudioLanguageEvent estende a classe *org.davic.media.AudioLanguageControl*, descrita na API GEM 1.1:2007. Esta classe deve oferecer uma função para validar e/ou invalidar o monitoramento de eventos de alteração de linguagem de áudio.

Os métodos públicos são os seguintes:

- void addAudioLanguageEventListener(AudioLanguageEventListener lis)
  - Este método adiciona o *Listener* para monitorar as alterações da linguagem de áudio.
- void removeAudioLanguageEventListener(AudioLanguageEventListener lis)
  - Este método remove os *Listeners* que monitoram as alterações da linguagem de áudio.

6.2.3.4 Classe AudioLanguageEventListener

A classe AudioLanguageEventListener estende a classe *java.util.EventListener*. Ela é um *Listener* que é notificado quando das alterações da linguagem do áudio.

Os métodos públicos são os seguintes:

- *void audioLanguageChanged* (*AudioLanguageChangeEvent* e)
  - Este método é utilizado quando o dispositivo principal de áudio é alterado.

6.2.4 API de canal de retorno

6.2.4.1 Especificação dos pacotes componentes

A API br.org.sbtvd.net.rc é uma extensão do pacote org.dvb.net.rc que é parte da API GEM 1.1 (2007). Através desta API o sistema Ginga possui a habilidade de enviar mensagens assíncronas utilizando o canal de retorno do receptor. Deve ser capaz de enviar a mensagem após o estabelecimento da conexão ou em um horário pré-definido.O pacote seguinte compõe a API de canal de retorno do Ginga-J:

- br.org.sbtvd.net.rc

6.2.4.2 Classe RCMessage

A classe RCMessage define um objeto que deve ser responsável por enviar um objeto serializável a um localizador definido em um objeto URL passado como parâmetro.

Os métodos públicos desta classe são os seguintes:

- RCMessage (Serializable message)
  - Gera um novo elemento RCMessage para enviar um objeto serializável (que implementa a interface *Serializable*), passado como parâmetro.
- void setLocator(URL locator)
  - Determina o localizador para onde a mensagem será enviada.
- void sendMessage() throws java.net.IOException
  - Envia o objeto serializável ao localizador pré-definido. Este médoto pode lançar uma exceção do tipo *java.net.IOException*.

6.2.4.3 Classe RCAsynchronous

A classe RCAsynchronous define um objeto que deve ativar o RCMessage para enviar seus dados em uma data específica ou quando a conexão do canal de retorno do receptor é estabelecida.

Os métodos públicos são os seguintes:

- RCAsynchronous (java.util.Date date)
  - Através deste método cria-se um novo objeto RCAsynchronous com data específica (passada como parâmetro). Se a data do objeto for nula, a mensagem pode ser enviada na primeira vez que a conexão for estabelecida.
- void setRCMessage(RCMessage message)
  - Determina o objeto RCMessage que deve ser ativado quando a data for alcançada.
- boolean isReady()
  - Este método retorna um valor booleano (verdadeiro ou falso) representando o estado da mensagem: verdadeiro (*true*) se a mensagem estiver pronta para ser enviada, falso (*false*) em caso contrário.

6.2.4.4 Classe AsynchronousMessageTable

Esta classe define a tabela de mensagem assíncrona contendo os objetos RCAsynchronous registrados para enviar mensagens assíncronas no canal de retorno do receptor. O objeto *AsynchronousMessageTable* implementa a interface *org.dvb.net.rc.ConnectionListener* para capturar os eventos de estabelecimento de conexão.

Os métodos públicos são os seguintes:

- static AsynchronousMessageTable getInstance ()
  - Retorna um objeto *AsynchronousMessageTable*.
- void registerRCAsynchronous (RCAsynchronous rca)
  - Registra um objeto RCAsynchronous (passado como parâmetro) na tabela de mensagens assíncronas – mensagens que serão enviadas quando uma conexão for estabelecida.
- void unregisterRCAsynchronous (RCAsynchronous rca)
  - Desvincula um objeto RCAsynchronous (passado como parâmetro) da tabela de mensagens assíncronas.

6.2.5 Configurações do usuário e API de preferências

O Ginga-J inclui definições da GEM 1.1:2007, seção 11.9.2, e a API especificada na GEM 1.1:2007, Anexo L.

O seguinte pacote compõe essa API:

- org.dvb.user

6.2.6 Perfil e propriedades da versão

O Ginga-J inclui definições da GEM 1.1:2007, seção 11.9.3.

6.3 API aderentes à especificação ARIB STD B-23

6.3.1 Informações do serviço

A API de informações de serviço da especificação Ginga-J é responsável por fornecer aos aplicativos o acesso às informações presentes nas tabelas de informações de serviço MPEG. Tais informações devem incluir os fluxos de áudio e vídeo presentes em cada serviço multiplexado, incluindo também descrição textual dos serviços e eventos

que o compõem, entre outros. A API de informações do serviço está de acordo com a definição ARIB STD-B.23:2006 (anexo M).

Os seguintes pacotes compõem a API de informações do serviço:

- jp.or.arib.tv.si
- jp.or.arib.tv.net

6.3.2 API de informações de serviço independente de protocolo

A API de informações de serviço independente de protocolo deve estar de acordo com a ARIB STD-B23:2006, Anexo O.

6.4 Definições Ginga-J

6.4.1 API de integração de dispositivos

6.4.1.1 Especificação dos pacotes componentes

O pacote *br.org.sbtvd.interactiondevices* oferece funcionalidades inovadoras aos dispositivos que inconporarem o middleware Ginga-J. A API de integração de dispositivos tem a capacidade de forncer acesso para interação a vários usuários simultaneamente, através dispositivos de interação.

O seguinte pacote compõe esta API:

- br.org.sbtvd.interactiondevices

6.4.1.2 Classe GRemoteDeviceManager

A classe GRemoteDeviceManager define um objeto que deve administrar todas as conexões com os dispositivos de interatividade registrados com o Ginga.

Os métodos públicos disponíveis são:

- static GRemoteDeviceManager getInstance ()
  - Este método retorna um objeto *GRemoteDeviceManager*.
- GRemoteDevice[] getActiveDeviceList()
  - Este método retorna um *array* contendo objetos *GRemoteDevice* referenciando cada um dos dispositivos de interatividade registrados com o Ginga.

6.4.1.3 Classe GRemoteDevice

A classe GRemoteDevice define um objeto que deve ser uma representação abstrata de um dispositivo de interação. Esta classe oferece métodos que possibilitam a recuperação de informações acerca dos dispositivos registrados (tipo do dispositivo, funcionalidades disponíveis etc), bem como explorar as funcionalidades disponíveis do mesmo (utilizar recursos de gravação de áudio, vídeo, captura de imagens etc).

As constantes públicas estáticas presentes na classe GRemoteDevice são:

- int KEYBOARD_FACILITY
  - Um valor inteiro que representa a funcionalidade de teclado de um dispositivo de interação.
- int SCREEN_FACILITY

  - Um valor inteiro que representa a funcionalidade de tela de um dispositivo de interação.
- int SOUND_CAPTURE_FACILITY
  - Um valor inteiro que representa a funcionalidade de captura de áudio de um dispositivo de interação.
- int SOUND_PLAYER_FACILITY
  - Um valor inteiro que representa a funcionalidade de reprodução de áudio de um dispositivo de interação.
- int PICTURE_CAPTURE_FACILITY
  - Um valor inteiro que representa a funcionalidade de captura de imagens de um dispositivo de interação.
- int VIDEO_CAPTURE_FACILITY
  - Um valor inteiro que representa a funcionalidade de captura de vídeo de um dispositivo de interação.
- int VIDEO_PLAYER_FACILITY
  - Um valor inteiro que representa a funcionalidade de reprodução de vídeo de um dispositivo de interação.
- int DIALING_FACILITY
  - Um valor inteiro que representa a funcionalidade de efetuar ligações telefônicas de um dispositivo de interação.

Os métodos públicos da classe GRemoteDevice são:

- int getID()
  - Este método retorna um inteiro representando o identificador de um dispositivo de interação.
- String getDescription()
  - Este método retorna uma *string* (cadeia de caracteres) contendo a descrição do dispositivo de interação.
- String getParameter(String name)
  - Este método recebe uma *string* contendo um mnemônico para um parâmetro (característica) relacionada a um dispositivo de interação (por exemplo, a área da tela em *pixels* representado pelo mnemônico "*screen_area*") e retorna o valor correspondente em uma *String* (por exemplo "640x480"). Estes parâmetros (e seus respectivos mnemônicos) dependem do dispositivo de interação em questão.
- boolean isActive()
  - Este método retorna um valor booleano (verdadeiro ou falso) representando o estado do dispositivo: verdadeiro (*true*) se o dispositivo estiver ativo, falso (*false*) se o dispositivo estiver inativo ou não registrado com o Ginga.
- void addActionListener(GRemoteDeviceActionListener lis)
  - Este método registra o dispositivo em um *Listener* do tipo *GRemoteDeviceActionListener* (passado como parâmetro).
- void removeActionListener(HRemoteDeviceActionListener lis)
  - Este método desregistra o dispositivo de *Listener* do tipo *GRemoteDeviceActionListener* (passado como parâmetro).
- int submitFile(java.io.File file) throws java.io.IOException
  - Este método envia um arquivo (passado como parâmetro) para o dispositivo. O método retornará

um valor inteiro igual ao tamanho do arquivo em bytes em caso de sucesso ou -1 (menos um) em caso de falha. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.

- void startAudioRecording() throws IOException
    - Este método inicia a captura de áudio no dispositivo. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- void stopAudioRecording() throws IOException
    - Este método finaliza a captura de áudio no dispositivo. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- void startVideoRecording() throws IOException
    - Este método inicia a captura de vídeo no dispositivo. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- void stopVideoRecording() throws IOException
    - Este método finaliza a captura de vídeo no dispositivo. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- int takePicture() throws java.io.IOException
    - Este método dispara a captura de uma fotografia no dispositivo. Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- void dialNumber(String number)
    - Este método faz com que o dispositivo efetue uma ligação telefônica para o número passado como parâmetro em uma cadeia de caracteres (*string*). Este médoto pode lançar uma exceção do tipo *java.io.IOException*.
- org.havi.HScene getHScene()
    - Este método retorna um objeto do tipo org.havi.HScene, relativo ao dispositivo, de forma que elementos de interface possam ser manipulados no objeto HScene de cada dispositivo de interação.

6.4.1.4 Interface GRemoteDeviceActionListener

A interface GRemoteDeviceActionListener deve conter métodos que devem ser implementados por qualquer objeto que deva ser notificado sobre eventos relacionados às atividades dos dispositivos de interação registrados com o Ginga.

O método público que deverá ser implementado:

- void notifyDeviceEvent(GRemoteEvent event)
    - Este método notifica o objeto que implementa a interface *GRemoteDeviceActionListener* acerca de eventos que ocorreram nos dispositivos, através da passagem de um objeto *GRemoteEvent* como parâmetro.

6.4.1.5 Classe GRemoteEvent

A classe GRemoteEvent descreve um objeto que representa um evento relacionado a um dispositivo de interação registrado com o Ginga. Este objeto encapsula dados relacionados ao evento.

As constantes públicas estáticas presentes na classe GRemoteEvent são:

- int AUDIO_REQUEST
    - Um valor inteiro que define que o evento está relacionado a uma requisição por áudio e contém o estado dessa requisição encapsulado.
- int VIDEO_REQUEST

    - Um valor inteiro que define que o evento está relacionado a uma requisição por vídeo e contém o estado dessa requisição encapsulado.
- int PICTURE_REQUEST
    - Um valor inteiro que define que o evento está relacionado a uma requisição por imagem e contém o estado dessa requisição encapsulado.
- int FILE_TRANSFER
    - Um valor inteiro que define que o evento está relacionado a uma requisição por transferência e contém o estado dessa requisição encapsulado.
- int NUMBER_DIALED
    - Um valor inteiro que define que o evento está relacionado ao estabelecimento de uma ligação telefônica (por parte do dispositivo de interação) e contém o estado dessa requisição encapsulado.
- int AUDIO_DATA
    - Um valor inteiro que define que o evento possui dados de áudio encapsulados.
- int VIDEO_DATA
    - Um valor inteiro que define que o evento possui dados de vídeo encapsulados.
- int PICTURE_DATA
    - Um valor inteiro que define que o evento possui uma imagem encapsulada.
- int KEY_DATA
    - Um valor inteiro que define que o evento possui um evento de teclas encapsulado.

Os métodos públicos da classe GRemoteEvent são:

- Object getSource()
    - Este método retorna um *Object* referenciando o dispositivo de interação (*GRemoteDevice*) que foi a fonte do evento em questão.
- int getType()
    - Este método retorna um valor inteiro de acordo com o tipo do evento em questão (de acordo com as constantes estáticas definidas na mesma classe).
- byte[] getData()
    - Este método retorna os dados relacionados ao evento em questão.
- String getDescription()
    - Este método retorna uma cadeia de caracteres (*String*) relacionada a uma descrição do evento em questão.
- boolean isSuccessful()
    - Este método retorna um valor booleano (verdadeiro ou falso) representando o estado do evento: se verdadeiro (*true*) o evento foi bem sucessidido, falso (*false*) em caso contrário.

6.4.1.6 Classe GRemoteKeyEvent

A classe GRemoteKeyEvent estende *java.awt.event.KeyEvent*, e é relacionada a eventos de teclas originados em dispositivos de interatividade.

Os métodos públicos da classe GRemoteKeyEvent são:

- GRemoteDevice getSourceDevice()
    - Este método retorna um objeto *GRemoteDevice* referenciando o dispositivo de interação que foi a fonte do evento em questão.

6.4.1.7 Classe GRemoteUserEvent

A classe GRemoteUserEvent estende *org.dvb.event.UserEvent*, e é relacionada a eventos do usuário originados em dispositivos de interatividade.

Os métodos públicos da classe GRemoteUserEvent são:

- GRemoteDevice getSourceDevice()
    - Este método retorna um objeto *GRemoteDevice* referenciando o dispositivo de interação que foi a fonte do evento em questão.

6.4.2 API de ponte Ginga-NCL

6.4.2.1 Especificação dos pacotes componentes

O pacote *br.org.sbtvd.bridge* contém o conjunto de classes disponíveis para a ponte entre os aplicativos de informação e processo, em ambiente Ginga. As funções disponíveis nas categorias que são descritas abaixo permitem o desenvolvimento de aplicativos de procedimento Ginga-J incluindo aplicativos Ginga-NCL.

O seguinte pacote compõe esta API:

- br.org.sbtvd.bridge

6.4.2.2 Classe NCLBaseElement

A classe NCLBaseElement é uma superclasse de classes representando os elementos [regionBase], [descriptorBase], [ruleBase], [transitionBase], ou [connectorBase] de um documento NCL.

Os métodos públicos da classe NCLBaseElement são:

- NCLBaseElement()
    - Método construtor para classe NCLBaseElement.
- boolean addImportBase(NCLImportBase importBase)
    - Adiciona um elemento [importBase] a um elemento de base ([regionBase], [descriptorBase], [ruleBase], [transitionBase], ou [connectorBase]) de um documento NCL em uma base privada.
- NCLImportBase[] getAImportBase()
    - Retorna uma lista de todas as NCLImportBases registradas neste NCLBaseElement.
- boolean removeImportBase(NCLImportBase importBase)
    - Remove um elemento [importBase] de um elemento de base de um documento NCL em uma base privada.

6.4.2.3 Classe NCLCompositeNode

A classe NCLCompositeNode é uma classe que representa um nó composto NCL ([body], [context], ou [switch]) de um documento NCL.

Os métodos públicos da classe NCLCompositeNode são:

- NCLCompositeNode(java.lang.String nodeId)
    - Método construtor para NCLCompositeNode.
- boolean addLink(NCLLink link)
    - Adiciona um elemento NCL [link] a um nó composto (elemento [body], [context], ou [switch]) de um documento NCL em uma base privada.
- boolean addNode(NCLNode node)

  - Adiciona um nó NCL (elemento [media], [context], ou [switch]) a um nó composto (elemento [body], [context], ou [switch]) de um documento NCL em uma base privada.
- NCLNode[] getANCLNodes()
  - Retorna uma lista de todos os NCLNodes contidos neste NCLNode.
- boolean removeLink(NCLLink link)
  - Remove um elemento NCL [link] de um nó composto (elemento [body], [context], ou [switch]) de um documento NCL em uma base privada.
- boolean removeNode(NCLNode node)
  - Remove um nó NCL (elemento [media], [context], ou [switch]) de um nó composto (elemento [body], [context], ou [switch]) de um documento NCL em uma base privada.

6.4.2.4 Classe NCLConnector

A classe NCLConnector é uma classe que representa um elemento [connector] de um documento NCL.

Os métodos públicos da classe NCLConnector são:

- NCLConnector(java.lang.String connectorId)
  - Método construtor para NCLConnector.

6.4.2.5 Classe NCLConnectorBase

A classe NCLConnectorBase é uma classe que representa um elemento [connectorBase] de um documento NCL.

Os métodos públicos da classe NCLConnectorBase são:

- NCLConnectorBase(java.lang.String NCLConnectorId)
  - Método construtor para NCLConnectorBase.
- boolean addConnector(NCLConnector conector)
  - Adiciona um elemento [connector] ao [connectorBase] de um documento NCL em uma base privada.
- NCLConnector[] getAConnector()
  - Retorna uma lista de todos os NCLConnector registrados neste NCLConnectorBase
- boolean removeConnector(NCLConnector conector)
  - Remove um elemento [connector] do [connectorBase] de um documento NCL em uma base privada.

6.4.2.6 Classe NCLDescriptor

A classe NCLDescriptor é uma classe que representa um elemento [descriptor] de um documento NCL.

O método público da classe NCLDescriptor são:

- NCLDescriptor(java.lang.String descriptorId)
  - Método construtor para NCLDescriptor.

6.4.2.7 Classe NCLDescriptorBase

A classe NCLDescriptorBase é uma classe que representa um elemento [descriptorBase] de um documento NCL.

Os métodos públicos da classe NCLDescriptorBase são:

- NCLDescriptorBase(java.lang.String descriptorBase)
  - Método construtor para NCLDescriptorBase.
- boolean addDescriptor(NCLDescriptor descritor)

  - Adiciona um elemento [descriptor] ao [descriptorBase] de um documento NCL em uma base privada.
- boolean addDescriptorSwitch(NCLDescrSwitch descrSwitch)
  - Adiciona um elemento [descriptorSwitch] ao [descriptorBase] de um documento NCL em uma base privada.
- NCLDescriptor[] getADescriptor()
  - Retorna uma lista de todos os NCLDescriptors registrados neste NCLDescriptorBase.
- NCLDescrSwitch[] getADescrSwitch()
  - Retorna uma lista de todos os NCLDescrSwitches registrados neste NCLDescriptorBase.
- boolean removeDescriptor(NCLDescriptor descritor)
  - Remove um elemento [descriptor] do [descriptorBase] de um documento NCL em uma base privada.
- boolean removeDescriptorSwitch(NCLDescrSwitch descrSwitch)
  - Remove um elemento [descriptorSwitch] do [descriptorBase] de um documento NCL em uma base privada.

6.4.2.8 Classe NCLDescrSwitch

A classe NCLDescrSwitch é uma classe que representa um elemento [descriptorSwitch] de um documento NCL.

O método público da classe NCLDescrSwitch é:

- NCLDescrSwitch(java.lang.String descrSwitchId)
  - Método construtor para NCLDescrSwitch.

6.4.2.9 Classe NCLDocument

A classe NCLDocument é uma classe que representa um documento NCL.

Os métodos públicos da classe NCLDocument são:

- NCLDocument(java.lang.String documentId)
  - Método construtor para NCLDocument.
- boolean addConnectorBase(NCLConnectorBase connectorBase)
  - Adiciona um elemento [connectorBase] ao elemento [head] de um documento NCL em uma base privada.
- boolean addDescriptorBase(NCLDescriptorBase descriptorBase)
  - Adiciona um elemento [descriptorBase] ao elemento [head] de um documento NCL em uma base privada.
- boolean addImportedDocumentBase(NCLImportedDocumentBase importedDocumentBase)
  - Adiciona um elemento [importedDocumentBase] ao elemento [head] de um documento NCL em uma base privada.
- boolean addRegionBase(NCLRegionBase regionBase)
  - Adiciona um elemento [regionBase] ao elemento [head] de um documento NCL em uma base privada.
- boolean addRuleBase(NCLRuleBase ruleBase)
  - Adiciona um elemento [ruleBase] ao elemento [head] de um documento NCL em uma base privada.
- boolean addTransitionBase(NCLTransitionBase transitionBase)
  - Adiciona um elemento [transitionBase] ao elemento [head] de um documento NCL em uma base privada.
- NCLNode getBody()
  - Retorna o elemento [body] de um documento NCL em uma base privada (um nó NCL).
- NCLConnectorBase getConnectorBase()
  - Rertorna o [connectorBase] de um documento NCL em uma base privada.
- NCLDescriptorBase getDescriptorBase()
  - Rertorna o [descriptorBase] de um documento NCL em uma base privada.
- NCLImportedDocumentBase getImportedDocumentBase()

  - Rertorna o [importedDocumentBase] de um documento NCL em uma base privada.
- NCLRegionBase getRegionBase()
  - Rertorna o [regionBase] de um documento NCL em uma base privada.
- NCLRuleBase getRuleBase()
  - Rertorna o [ruleBase] de um documento NCL em uma base privada.
- NCLTransitionBase getTransitionBase()
  - Rertorna o [transitionBase] de um documento NCL em uma base privada.
- boolean pauseDocument()
  - Pausa a apresentação de um documento NCL em uma base privada.
- boolean removeConnectorBase(NCLConnectorBase connectorBase)
  - Remove um elemento [connectorBase] do elemento [head] de um documento NCL em uma base privada.
- boolean removeDescriptorBase(NCLDescriptorBase descriptorBase)
  - Remove um elemento [descriptorBase] do elemento [head] de um documento NCL em uma base privada.
- boolean removeImportedDocumentBase(NCLImportedDocumentBase importedDocumentBase)
  - Remove um elemento [importedDocumentBase] do elemento [head] de um documento NCL em uma base privada.
- boolean removeRegionBase(NCLRegionBase regionBase)
  - Remove um elemento [regionBase] do elemento [head] de um documento NCL em uma base privada.
- boolean removeRuleBase(NCLRuleBase ruleBase)
  - Remove um elemento [ruleBase] do elemento [head] de um documento NCL em uma base privada.
- boolean removeTransitionBase(NCLTransitionBase transitionBase)
  - Remove um elemento [transitionBase] do elemento [head] de um documento NCL em uma base privada.
- boolean resumeDocument()
  - Resume a apresentação de um documento NCL em uma base privada.
- boolean startDocument(java.lang.String interfaceId)
  - Inicia a apresentação de um documento NCL em uma base privada, começando a apresentação a partir da interface especificada do documento.
- boolean stopDocument()
  - Para a apresentação de um documento NCL em uma base privada.

6.4.2.10 Classe NCLEvent

A classe de evento raiz para todos os eventos NCL. As máscaras de evento definidas nesta classe são utilizadas para especificar quais tipos de eventos um NCLEventListener deve ouvir. As máscaras são *bitwise-ORed* juntas e passadas para NCLNode.addNCLEventListener.

As constantes públicas estáticas da Classe NCLEvent são:

- static long CHANGED_EVENT_MASK
  - Máscara de evento para eventos de nós alterados.
- protected int id
  - Identificação do evento.
- static long PRESENTED_EVENT_MASK
  - Máscara de evento para os eventos de nó apresentados.
- protected int id
  - Identificação do evento.
- static long SELECTED_EVENT_MASK
  - Máscara de evento para eventos de nó selecionados.
- protected int value
  - Valor do evento para Changed_Events.

Os métodos públicos da classe NCLEvent são:

- NCLEvent()
  - Método construtor para NCLEvent.
- int getID()
  - Retorna o tipo de evento.
- int getValue()
  - Retorna o valor do evento para CHANGED_EVENTS.

6.4.2.11 Interface NCLEventListener

A interface *Listener* que deve ser implementada por quem deseje receber notificação de eventos distribuídos para objetos que são elementos do NCLNode ou suas sub-categorias.

A categoria que se interessa em monitorar os eventos NCL de um NCLNode implementa esta interface, e o objeto cirado com esta categoria é registrado com o NCLNode, usando o método NCLNode's addNCLEventListener. Quando um evento é distribuído no NCL Node, o método eventDispatcheded desse objeto é executado.

O método público da interface NCLEventListener é:

- void NCLEventDispatched(NCLEvent event)
  - Método executado quando um evento é distribuído no NCLNode.

6.4.2.12 Classe NCLGingaSettingsNodes

A classe representando um nó NCL (elemento [node]) cujos atributos são variáveis globais definidas pelo autor do documento ou variáveis de ambiente que podem ser manipuladas pelo processamento do documento NCL.

Os métodos públicos da classe NCLGingaSettingsNodes são:

- NCLGingaSettingsNodes(java.lang.String nodeId)
  - Método construtor para NCLGingaSettingsNodes.
- String getDefaultFocusBorderColor()
  - Retorna a cor padrão aplicada à borda de um elemento em foco.
- String getDefaultFocusBorderWidth()
  - Retorna a largura padrão (in pixels) aplicada à borda de um elemento em foco.
- String getDefaultFocusTransparency()
  - Retorna a transparência padrão aplicada à borda de um elemento em foco.
- String getDefaultSelBorderColor()
  - Retorna a cor padrão aplicada à borda de um elemento em foco quando ativado.
- String getSystemAudioType(java.lang.String deviceId)
  - Retorna o tipo de áudio do dispositivo.
- String getSystemCaption()
  - Retorna a linguagem de *Caption*.
- String getSystemCPU()
  - Retorna o desempenho do CPU em MIPS.
- String getSystemLanguage()
  - Retorna a linguagem do áudio.
- String getSystemMemory()
  - Retorna o espaço da memória em Mbytes.
- String getSystemOperatingSystem()
  - Retorna o tipo de sistema operacional.
- String getSystemReturnBitRate(java.lang.String returnChannelId)
  - Retorna a taxa de bit do canal de retorno em Kbps.
- String getSystemScreenGraphicSize(java.lang.String deviceId)

    - Retorna a resolução configurada para o plano gráfico da tela do dispositivos em (linhas, pixels/linha).
- String getSystemScreenSize(java.lang.String deviceId)
    - Retorna o tamanho da Tela de um dispositivo em (linhas, pixels/linhas).
- String getsyStemSubtitle()
    - Retorna a linguagem da legenda.
- String getUserAge()
    - Retorna a idade do Usuário.
- String getUserGenre()
    - Retorna o sexo do Usuário.
- String getUserLocation()
    - Retorna a localização do Usuário (CEP).

6.4.2.13 Classe NCLImportBase

A classe NCLImportBase é a classe que representa um elemento [importBase] de um documento NCL.

O método público da classe NCLImportBase é:

- NCLImportBase(java.lang.String importBaseId)
    - Método construtor para NCLImportBase.

6.4.2.14 Classe NCLImportedDocumentBase

A classe NCLImportedDocumentBase é uma classe que representa um elemento [importedDocumentBase] de um documento NCL.

Os métodos públicos da classe NCLImportedDocumentBase são:

- NCLImportedDocumentBase(java.lang.String importedDocumentBase)
    - Método construtor para NCLImportedDocumentBase.
- boolean addImportNCL(NCLImportNCL importNCL)
    - Adiciona um elemento [importNCL] ao elemento [importedDocumentBase] de um documento NCL em uma base privada.
- NCLImportNCL[] getAImportNCL()
    - Retorna uma lista de todos os NCLImportNCLs registrados neste NCLImportedDocumentBase.
- boolean removeImportNCL(NCLImportNCL importNCL)
    - Remove um elemento [importNCL] do elemento [importedDocumentBase] de um documento NCL em uma base privada.

6.4.2.15 Classe NCLImportNCL

A classe NCLImportNCL é uma classe que representa um elemento [importNCL] de um documento NCL.

O método público da classe NCLImportBase é:

- NCLImportNCL(java.lang.String importNCL)
    - Método construtor para NCLImportNCL.

6.4.2.16 Classe NCLInterface

A classe NCLInterface é uma classe que representa uma definição de interface NCL (elemento [port], [area], [attribute], ou [switchPort]).

O método público da classe NCLImportBase é:

- NCLInterface(java.lang.String interfaceId)
  - Método construtor para NCLInterface.

6.4.2.17 Classe NCLLink

A classe NCLLink é uma classe que representa um elemento [link] de um documento NCL.

O método público da classe NCLImportBase é:

- NCLLink(java.lang.String linkId)
  - Método construtor para NCLLink.

6.4.2.18 Classe NCLMediaNode

A classe NCLMediaNode representa um Nó NCL de tipo de mídia.

Os métodos públicos da classe NCLMediaNode são:

- NCLMediaNode(java.lang.String NCLDocument)
  - Método construtor para NCLMediaNode.
- java.lang.String getType()
  - Retorna o nó do tipo de mídia.
- java.lang.String getUri()
  - Retorna o nó URI.

6.4.2.19 Classe NCLNode

A classe NCLNode é a classe que representa a definição de um elemento de nó NCL (elemento [media], [context], ou [switch]).

Os métodos públicos da classe NCLNode são:

- NCLNode(java.lang.String nodeId)
  - Método construtor para NCLNode.
- boolean addInterface(NCLInterface nclInterface)
  - Adiciona uma interface NCL (elemento [port], [area], [attribute], ou [switchPort]) a um nó (elemento [media], [body], [context], ou [switch]) de um documento NCL em uma base privada.
- boolean removeInterface(NCLInterface nclInterface)
  - Remove uma interface NCL (elemento [port], [area], [attribute], ou [switchPort]) de um nó (elemento [media], [body], [context], ou [switch]) de um documento NCL em uma base privada.
- boolean setAttributeValue(java.lang.String attributeId,java.lang.String value)
  - Configura o valor para um atributo para o qual é fornecido identidade. A Id atribuída deve identificar um elemento [attribute] ou [switchPort]. O [attribute] ou [switchPort] deve pertencer a um nó (elemento [body], [context], [switch] ou [media]) de um documento NCL em uma base privada.
- void addNCLEventListener(NCLEventListener listener, long nclEventMask)
  - Adiciona um NCLEventListener para receber todos os NCLEvents distribuídos pela máquina associada ao nó que se vincula à eventMask fornecida.
- void removeNCLEventListener(NCLEventListener listener)
  - Remove um NCLEventListener da categoria de recepção dos NCLEvents distribuídos.
- NCLEventListener[] getANCLEventListeners()
  - Retorna uma lista de todos os NCLEventListeners registrados neste NCLNode. Note que os objetos ouvintes adicionados várias vezes aparecem apenas uma vez na lista retornada.
- NCLEventListener[] getANCLEventListeners(long nclEventMask)
  - Retorna uma lista de todos os NCLEventListeners registrados neste NCLNode que ouvem a todos os tipos de eventos indicados na eventMask. Note que os objetos ouvintes adicionados várias vezes aparecem apenas uma vez na lista retornada.

6.4.2.20 Classe NCLPrivateBase

A classe NCLPrivateBase é uma classe que coleta Documentos NCL divididos pelo NCL Formater.

Os métodos públicos da classe NCLPrivateBase são:

- NCLPrivateBase()
  - Método construtor para NCLPrivateBase.
- boolean activateBase()
  - Liga uma base privada aberta.
- boolean deactivateBase()
  - Desliga uma base privada aberta.
- boolean saveBase(java.lang.String device)
  - Salva todo o conteúdo de uma rede privada em um dispositivo de armazenamento boleana permanente (se disponível saveBase(java.lang.String device)
- boolean closeBase()
  - Fecha a base privada e descarta todo o conteúdo da mesma.
- boolean addDocument(NCLDocument nclDocument)
  - Adiciona um documento NCL à base privada.
- boolean removeDocument(NCLDocument nclDocument)
  - Remove um documento NCL de uma base privada.
- NCLDocument[] getADocuments()
  - Retorna uma lista de todos os NCLDocuments registrados neste NCLPrivateBase

6.4.2.21 Classe NCLRegion

A classe NCLRegion é uma categoria que representa um elemento [region] de um documento NCL.

Os métodos públicos da classe NCLRegion são:

- NCLRegion(java.lang.String regionId)
  - Método construtor para NCLRegion.
- boolean addRegion(NCLRegion região)
  - Adiciona um elemento [region] como uma criança de outra região [region] na [regionBase] de um documento NCL.
- boolean removeRegion(NCLRegion região)
  - Remove um elemento [region] da [regionBase] de um documento NCL em uma base privada.
- NCLRegion[] getARegion()
  - Retorna uma lista de todos os NCLRegions registrados neste NCLRegionBase.

6.4.2.22 Classe NCLRegionBase

A classe NCLRegionBase é uma classe que representa um elemento [regionBase] de um documento NCL.

Os métodos públicos da classe NCLRegionBase são:

- NCLRegionBase(java.lang.String regionBaseId)
  - Construtor
- boolean addRegion(NCLRegion parentRegion, NCLRegion região)
  - Adiciona um elemento [region] como uma criança de outra região [region] na [regionBase] de um documento NCL.
- boolean removeRegion(NCLRegion região)
  - Remove um elemento [region] da [regionBase] de um documento NCL em uma base privada.

- NCLRegion[] getARegion()
  - Retorna uma lista de todos os NCLRegions registrados neste NCLRegionBase.

6.4.2.23 Classe NCLRule

A classe NCLRegionBase representa um elemento [rule] de um documento NCL

O método público da classe NCLRegionBase é:

- NCLRule(java.lang.String ruleId)
  - Método construtor para NCLRule.

6.4.2.24 Classe NCLRuleBase

A classe NCLRuleBase representa um elemento [ruleBase] de um documento NCL.

Os métodos públicos da classe NCLRuleBase são:

- NCLRuleBase(java.lang.String ruleBaseId)
  - Método construtor para NCLRuleBase.
- boolean addRule(NCLRule regra)
  - Adiciona um elemento [rule] ao [ruleBase] de um documento NCL em uma base privada.
- boolean removeRule(NCLRule regra)
  - Remove um elemento [rule] do [ruleBase] de um documento NCL em uma base privada.
- NCLRule[] getARule()
  - Retorna uma lista de todos os NCLRules registrados neste NCLRuleBase.

6.4.2.25 Classe NCLTransition

A classe NCLTransition representa um elemento [transition] de um documento NCL.

Os métodos públicos da classe NCLTransition são:

- NCLTransition(java.lang.String transitionId)
  - Método construtor para PrivateBaseManager.

6.4.2.26 Classe NCLTransitionBase

A classe NCLTransitionBase representa um elemento [transitionBase] de um documento NCL.

Os métodos públicos da classe NCLTransitionBase são:

- NCLTransitionBase(java.lang.String transitionBaseId)
  - Método construtor para NCLTransitionBase.
- boolean addTransition(NCLTransition transition)
  - Adiciona um elemento [transition] ao [transitionBase] de um documento NCL em uma base privada..
- boolean removeTransition(NCLTransition transition)
  - Remove um elemento [transition] do [transitionBase] de um documento NCL em uma base privada.
- NCLTransition[] getAConnector()
  - Retorna uma lista de todos os NCLTransitions registrados neste NCLTransitionBase.

6.4.2.27 Classe PrivateBaseManager

A classe PrivateBaseManager é responsável pela recepção do documento NCL editando comandos e mantendo os documentos NCL ativos (documentos sendo apresentados).

Os métodos públicos da classe PrivateBaseManager são:

- PrivateBaseManager()
  - Método construtor para PrivateBaseManager.
- NCLPrivateBase openBase(java.lang.String baseId)
  - Abre uma nova base privada. Se a base privada não existir, uma nova base é criada.

6.5 Lista completa de API Java

Todos os pacotes incluídos na especificação Ginga-J são listados abaixo:

- java.awt
- java.awt.event
- java.awt.image
- java.beans
- java.io
- java.lang
- java.lang.reflect
- java.net
- java.security
- java.security.cert
- java.util
- java.util.zip
- javax.media
- javax.media.protocol
- javax.media.bean.playerbean
- javax.media.cdm
- javax.media.control
- javax.media.datasink
- javax.media.format
- javax.media.pim
- javax.media.pm
- javax.media.renderer
- javax.media.rtp
- javax.media.rtp.event
- javax.media.rtp.rtcp
- javax.media.util
- javax.tv.graphics

- javax.tv.locator
- javax.tv.media
- javax.tv.media.protocol
- javax.tv.net
- javax.tv.service
- javax.tv.service.guide
- javax.tv.service.navigation
- javax.tv.service.selection
- javax.tv.service.transport
- javax.tv.util
- javax.tv.xlet
- java.math
- java.rmi
- java.security.spec
- javax.net
- javax.net.ssl
- javax.security.cert
- org.davic.media
- org.davic.resources
- org.davic.mpeg
- org.davic.mpeg.sections
- org.davic.net
- org.davic.net.dvb
- org.davic.net.tuning
- org.havi.ui
- org.havi.ui.event
- org.dvb.application
- org.dvb.dsmcc
- org.dvb.event
- org.dvb.io.ixc
- org.dvb.io.persistent
- org.dvb.lang
- org.dvb.media
- org.dvb.net
- org.dvb.net.tuning
- org.dvb.net.rc
- org.dvb.test

- org.dvb.ui
- org.dvb.user
- jp.or.arib.tv.si
- jp.or.arib.tv.net
- br.org.sbtvd.net.tuning
- br.org.sbtvd.media
- br.org.sbtvd.net.rc
- br.org.sbtvd.interactiondevices
- br.org.sbtvd.bridge

Anexo A
(informativo)
Harmonização com especificações internacionais

A especificação Ginga-J é compatível com a ITU Recommendation J.202 (Harmonização de Procedimentos para Formatos de Conteúdos para Aplicativos de Televisão Interativa), que é totalmente compatível com o Padrão ETSI TS 102 819 (GEM 1.0.2 – MHP Executável Globalmente). A especificação Ginga-J também é compatível com o padrão ARIB STD-B23 (*Application Execution Engine Platform for Digital Broadcasting – English Translation*).

Além da compatibilidade com os padrões mencionados, a especificação Ginga-J introduz funcionalidades adicionais para adequar-se às exigências brasileiras.

A compatibilidade com a ARIB STD-B23 justifica-se pela adoção do ISDB-T (ARIB STD-B31 – *Transmission Coding Standard*) como base do Projeto 00:001.85-001. Alguns parâmetros relevantes para as informações sobre o serviço de televisão digital recaem sobre o método de transmissão utilizado, assim, as informações sobre o serviço brasileiro especificados pelo Projeto 00:001.85-003 devem ser compatíveis com a Especificação ARIB B.10. Desta forma, a API de Informações do Serviço Ginga-J está de acordo com a API de Informações de Serviços da ARIB B.23, garantindo a compatibilidade necessária.

GEM fornece um *framework* para a definição de uma especificação de terminal GEM, que atualmente orienta os esforços para harmonizar as definições de *middleware* ACAP, OCAP, MHP e ARIB STD-B23. A compatibilidade do Ginga-J com o GEM tem por objetivo possibilitar a execução dos aplicativos GEM em receptores Ginga.

Para oferecer compatibilidade com outras especificações de sistemas *middleware*, a proposta Ginga-J incorpora as tecnologias descritas na especificação GEM 1.1:2007. Esta estratégia torna o Ginga-J compatível, em termos de *Common Core API* (API de núcleo comum) (listadas na Seção 5.2.1), com as especificações de *middleware* DVB-MHP, ARIB B.23, ATSC ACAP e OCAP.

Bibliografia

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

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

[3] ITU Recommendation J.200:2001, *Worldwide common core – Application environment for digital interactive television services*

[4] ITU Recommendation J.201:2004, *Harmonization of declarative content format for interactive television applications*

[5] ITU Recommendation J.202:2003, *Harmonization of procedural content formats for interactive TV applications*

[6] Sun Microsystems, Java TV AP:2007, <http://java.sun.com/products/javatv/>

[7] HAVi Level 2 Graphical User-Interface:2006, Specification of the Home Audio/Video Interoperability (HAVi) Architecture". HAVi, Inc. 2001. <http://www.havi.org>

[8] DAVIC 1.4:1998, Part 2 – DAVIC Specification Reference Models and Scenarios, <http://www.davic.org>

[9] DVB Document A103 – Globally Executable MHP (GEM) Specification 1.1". DVB Bluebook, 2007. <http://www.mhp.org/mhp_technology/gem/a103r1.tm3567r1.GEM1.1.1.pdf>

[10] TS 102 812:2003, D*igital video broadcasting (DVB) multimedia home platform (MHP)*

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

[12] ARIB STD-B10:2004, *Service information for digital broadcasting system*

[13] ATSC Standard:2005, *Advanced Common Application Platform* (ACAP)

[14] CableLabs:2005, *OpenCable Application Platform Specification* – OCAP 1.0 *Profile*