NORMA
BRASILEIRA

ABNT NBR 15606-7

Segunda edição
11.04.2013

Válida a partir de
11.05.2013

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiofusão digital
Parte 7: Ginga-NCL – Diretrizes operacionais para as ABNT NBR 15606-2 e ABNT NBR 15606-5

Digital terrestrial television – Data coding and transmission specification for digital broadcasting
Part 7: Ginga-NCL: Operational guidelines to ABNT NBR 15606-2 and ABNT NBR 15606-5

ICS 33.080; 33.160.01

ISBN 978-85-07-04157-3

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referência
ABNT NBR 15606-7:2013
59 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 das Diretivas 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.

A ABNT NBR 15606-7 foi elaborada na Comissão de Estudo Especial de Televisão Digital (ABNT/CEE-85). O seu Projeto circulou em Consulta Nacional conforme Edital nº 12, de 21.12.2010 a 18.02.2011, com o número de Projeto 85:000.00-006-7. O seu Projeto de Emenda circulou em Consulta Nacional conforme Edital nº 12, de 12.12.2012 a 11.02.2013, com o número de Projeto de Emenda 1 ABNT NBR 15606-7.

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

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.

Esta segunda edição incorpora a Emenda 1 de 11.04.2013 e cancela e substitui a edição anterior (ABNT NBR 15606-7:2011).

O Escopo desta Norma Brasileira em inglês é o seguinte:

Scope

This part of ABNT NBR 15606 details and explains the XML application language named Nested Context Language (NCL), the declarative language of the middleware Ginga, the data coding and the data transmission for digital broadcasting, providing the operational guidelines for an implementation in accordance with ABNT NBR 15606-2 and ABNT NBR 15606-5.

Introdução

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para o fato de que a exigência de conformidade com este documento ABNT pode envolver o uso de um direito de propriedade relativo a NCL.

A ABNT não se posiciona a respeito de evidências, validade e escopo desse direito de propriedade.

O proprietário desse direito de propriedade assegurou à ABNT que ele está preparado para negociar licenças sobre termos e condições razoáveis e não discriminatórias com os solicitantes. Sobre isso, uma declaração do proprietário desse direito está registrada na ABNT. Informações podem ser obtidas com:

Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Transferência de Tecnologia

Rua Marquês de São Vicente, 225 – Gávea, 22451-900 - Rio de Janeiro - RJ - Brasil.

A ABNT chama atenção para a possibilidade de que alguns dos elementos deste documento ABNT podem ser objeto de outras patentes e direitos de propriedade além dos identificados acima. A ABNT não pode considerada responsável pela identificação de quaisquer direitos de patente.

Esta parte da ABNT 15606 complementa e esclarece a padronização da linguagem NCL, que permite a especificação de apresentações multimídia interativas. Esta parte da ABNT NBR 15606 é parte das especificações de codificação de dados para o Sistema Brasileiro de Televisão Digital Terrestre e fornece as diretrizes operacionais relativas à máquina de apresentação Ginga-NCL do middleware chamado Ginga.

Esta parte da ABNT 15606 é primordialmente destinada às entidades que estão implementando terminais e/ou padrões com base no Ginga. Também é destinada aos desenvolvedores de aplicações que utilizam as funcionalidades do Ginga e de suas API. O middleware Ginga tem como objetivo garantir a interoperabilidade das aplicações em diferentes implementações de plataformas que o suportam.

As aplicações Ginga são classificadas sob duas categorias, dependendo se a aplicação inicialmente processada possui conteúdo de natureza declarativa ou imperativa. Essas categorias de aplicações são chamadas de aplicações declarativas e aplicações imperativas, respectivamente. Os ambientes de aplicação são igualmente classificados em duas categorias, dependendo se eles processam aplicações declarativas ou imperativas, sendo então chamados de Ginga-NCL e Ginga-J, respectivamente, no Sistema Brasileiro de Televisão Digital Terrestre. Apenas o ambiente Ginga-NCL é obrigatório em receptores portáteis. Usando o Ginga-NCL, o middleware Ginga dá suporte a código imperativo através da linguagem Lua.

Na Seção 5 é discutido como a semântica dos elementos e atributos NCL 3.0 são interpretados, quais são seus valores possíveis e predeterminados e as diretrizes operacionais para um formatador NCL (agente do usuário) ao gerenciar esses elementos e atributos. Na seção 6 os procedimentos dos exibidores dos objetos de mídia NCL são estabelecidos. A Seção 7 apresenta as diretrizes para o gerenciamento do ciclo de vida de uma aplicação NCL.

NOTA Ginga é uma marca comercial da PUC-Rio e da UFPB e NCL é uma marca comercial da PUC-Rio.

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiofusão digital
Parte 7: Ginga-NCL — Diretrizes operacionais para as ABNT NBR 15606-2 e ABNT NBR 15606-5

1 Escopo

Esta parte da ABNT NBR 15606 detalha e explica a especificação da linguagem de aplicação XML, denominada NCL (*Nested Context Language*), a linguagem declarativa do *middleware* Ginga, a codificação de dados e a transmissão de dados para radiodifusão digital, fornecendo as diretrizes operacionais para uma implementação de acordo com as ABNT NBR 15606-2 e ABNT NBR 15606-5.

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 15604, *Televisão digital terrestre – Receptores*

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

ABNT NBR 15606-2:2007, *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 15606-5:2008, *Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações*

ISO/IEC 13818-6*; Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*

ITU-T J.201, *Harmonization of declarative content format for interactive television applications*.

ITU-T H.761, *Nested Context Language (NCL) and Ginga-NCL for IPTV Services*

Nested Context Language 3.0, Part 11, *Declarative Hypermedia Objects in NCL: Nesting Objects with NCL code in NCL Documents*

Nested Context Language 3.0, Part 12, *Support to Multiple Exhibition Devices*

Nested Context Language 3.0, Part 10, *Imperative Objects in NCL: The NCLua Scripting Language*

Nested Context Language 3.0, Part 9, *NCL Live Editing Commands*

Guia Postal Brasileiro da Empresa Brasileira de Correios e Telégrafos, disponível em http://www.correios.com.br/cep/cep_estrutura.cfm

3 Termos e definições

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os seguintes termos e definições.

3.1
agente do usuário
qualquer programa que interprete um documento escrito na linguagem NCL conforme os termos presentes nas ABNT NBR 15606-2 e ABNT NBR 15606-5

NOTA Um agente do usuário pode exibir um documento, procurando garantir que os relacionamentos especificados pelo autor entre os objetos sejam respeitadas, ler em voz alta, imprimir, convertê-lo para outro formato etc.

3.2
ambiente da aplicação
contexto ou ambiente do *software* em que uma aplicação é processada

3.3
ambiente de aplicação declarativa
ambiente que suporta o processamento de aplicações declarativas

NOTA O agente de usuário NCL (formatador) é um exemplo de ambiente de aplicação declarativa.

3.4
aplicação
informações que expressam um conjunto específico de procedimentos observáveis

3.5
aplicação declarativa
aplicação que se inicia com, e utiliza principalmente, informações declarativas para expressar seu comportamento

NOTA Uma instância de documento NCL é um exemplo de aplicação declarativa.

3.6
aplicação nativa
função intrínseca, implementada por uma plataforma de receptor

NOTA A exibição de legendas é um exemplo de uma aplicação nativa.

3.7
atributo
parâmetro que representa uma propriedade

3.8
atributo de um elemento
propriedade de um elemento XML

3.9
áudio principal
áudio básico
fluxo básico de áudio cuja component_tag é igual a 0x10 para o receptor *full-seg* e 0x83 ou 0x85 para o receptor *one-seg*

3.10
autor
pessoa que especifica documentos NCL

3.11
canal de interatividade
canal de retorno
mecanismo de comunicação que fornece a conexão entre o receptor e um servidor remoto

3.12
caractere
letra específica ou outro símbolo de identificação
EXEMPLO "A"

3.13
ciclo de vida de uma aplicação
período de tempo do momento em que a aplicação é carregada até o momento em que é destruída

3.14
codificação de caracteres
mapeamento entre um valor de entrada inteiro e o caractere textual que é representado por esse mapeamento

3.15
comando e controle do armazenamento de mídia digital
DSM-CC
método de controle que fornece acesso a um arquivo ou fluxo de serviços digitais interativos

NOTA Para mais informações sobre DSM-CC, pode-se consultar a ISO/IEC 13818-6.

3.16
conteúdo NCL
conjunto de informações que consiste em um documento NCL e um grupo de dados, incluindo objetos (objetos de mídia ou de execução), acompanhando o documento NCL

3.17
elemento
unidade de estruturação do documento delimitada por delimitadores XML

NOTA Um elemento geralmente é delimitado por uma marca de início e uma de final, com exceção de um elemento vazio, que é delimitado por uma marca de elemento vazio.

3.18
elemento *property*
elemento NCL que define um nome de propriedade e seu valor associado

3.19
entidade da aplicação
unidade de informação que expressa parte de uma aplicação

3.20
evento
ocorrência no tempo, que pode ser instantânea ou ter uma duração mensurável

3.21
exibidor de mídia
componente de um ambiente de aplicação que decodifica ou executa um tipo específico de conteúdo

3.22
fluxo elementar
ES
fluxo básico que contém dados de vídeo, áudio ou dados privados

NOTA Um fluxo elementar é transmitido em uma sequência de pacotes PES com uma única *stream_id.*

3.23
fluxo de transporte
fluxo de transporte MPEG-2 contendo o empacotamento e multiplexação de vídeo, áudio e sinais de dados para os sistemas de transmissão digital

3.24
fonte
mecanismo que permite a interpretação específica de um caractere especial

EXEMPLO Tirésias, 12 pontos.

NOTA Na prática, um formato de fonte incorpora alguns aspectos de uma codificação de caracteres.

3.25
formatador NCL
componente de *software* que é responsável por receber a especificação de um documento NCL e controlar sua apresentação, com o objetivo de garantir que os relacionamentos entre os objetos de mídia especificados pelo autor sejam respeitados.

NOTA Os termos renderizador de documento, agente de usuário e exibidor são usados no mesmo sentido de formatador de documento.

3.26
eXtensible HTML
XHTML
versão estendida do HTML

NOTA Na especificação do XHTML, um documento HTML é reconhecido como aplicação XML.

3.27
identificador de recurso uniforme
URI
método de endereçamento que permite o acesso a objetos em uma rede

3.28
informações sobre serviços
SI
dados que descrevem programas e serviços

3.29
interface de programação de aplicações
API
bibliotecas de *software* que oferecem acesso uniforme ao sistema de serviços

3.30
linguagem de *script*
linguagem utilizada para descrever um conteúdo de objeto ativo, integrado em documentos NCL e HTML

3.31
locator
identificador que fornece a referência para uma aplicação ou recurso

3.32
máquina de apresentação
subsistema de um receptor que avalia e apresenta aplicações declarativas compostas de conteúdos de mídia, como áudio, vídeo, gráficos e texto, essencialmente com base em regras de apresentação definidas pela máquina de apresentação

NOTA A máquina de apresentação é responsável por controlar o procedimento da apresentação e iniciar outros procedimentos, em resposta a requisições do usuário e a outros eventos.

EXEMPLO navegador HTML e formatador NCL.

3.33
nó NCL
elementos NCL de <media>, <context>, <body> ou <switch>

3.34
objeto de mídia
conjunto de fragmentos de dados que pode representar um conteúdo de mídia ou um programa escrito em linguagem específica

3.35
perfil
especificação de uma classe de recursos que oferece diferentes níveis de funcionalidade em um receptor

3.36
perfil *one-seg*
serviço que pode ser recebido por um sintonizador de banda estreita (430 KHz), portanto, com economia de energia

NOTA O perfil *one-seg* também é conhecido como perfil portátil.

3.37
perfil *full-seg*
serviço que precisa de um demodulador de banda larga (5,7 MHz) para ser recebido

NOTA Dependendo da configuração de transmissão e das funcionalidades específicas do receptor, o serviço pode ser recebido por receptores móveis ou apenas por receptores fixos, porém, sem os benefícios da economia de energia. A resolução de vídeo pode ser de alta definição ou de definição *default*.

3.38
plataforma do receptor
hardware do receptor, sistema operacional e bibliotecas de *software* nativo escolhidos pelo fabricante

3.39
recurso
objeto ou serviço de rede identificado univocamente em uma rede

3.40
sintonização
ato de transferência entre dois fluxos de transporte MPEG

NOTA A transferência entre dois serviços SBTVD, realizada no mesmo fluxo de transporte, não é sintonização.

3.41
tempo normal de exibição
NPT
coordenada temporal absoluta que representa a posição em um fluxo

3.42
usuário
pessoa que interage com um agente de usuário para ver, ouvir ou utilizar um documento NCL

3.43
vídeo principal
vídeo básico
fluxo básico de vídeo cuja component_tag é igual a 0x00 para o receptor *full-seg* e 0x81 para o receptor *one-seg*

4 Símbolos e abreviaturas

Para os efeitos desta parte da ABNT NBR 15606, aplicam-se os seguintes símbolos e abreviaturas.

API	*Application Programming Interface* (Interface de programação de aplicações)
BML	*Broadcast Markup Language*
CSS	*Cascading Style Sheets*
DSM-CC	*Digital Storage Media Command and Control*
DTV	*Digital Television* (televisão digital)
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
MIME	*Multipurpose Internet Mail Extensions*
NCL	*Nested Context Language*
NIT	*Network Information Table*
NPT	*Normal Play Time*
PES	*Packetized Elementary Stream*
PID	*Packet Identifier*
PMT	*Program Map Table*
PSI	*Program Specific Information*
SBTVD	Sistema Brasileiro de Televisão Terrestre Digital
SMIL	*Synchronized Multimedia Integration Language*
TS	*Transport Stream* (fluxo de transporte)
URI	*Universal Resource Identifier*
URL	*Universal Resource Locator*
XHTML	*eXtensible HTML*
XML	*Extensible Markup Language*
W3C	*World-Wide Web Consortium*

5 Módulos NCL nos perfis EDTV e BDTV

5.1 Considerações gerais

O descrito em 5.2 a 5.26 está relacionado e complementa a ABNT NBR 15606-2:2007, Seção 7.

A definição completa dos módulos NCL 3.0, utilizando esquemas XML, é apresentada na ABNT NBR 15606-2.

Para cada perfil NCL (perfil avançado EDTV e perfil básico BDTV), é apresentada uma tabela indicando os elementos do módulo e seus atributos. Para um determinado perfil, os atributos e conteúdos (elementos filhos) dos elementos podem ser definidos no próprio módulo ou no perfil da linguagem que agrupa os módulos. Portanto, as Tabelas 1 a 37 mostram os atributos e conteúdos provenientes dos perfis, além daqueles definidos nos próprios módulos. Os atributos necessários de cada elemento estão sublinhados. Nas tabelas utilizam-se os seguintes símbolos: (?) opcional (zero ou uma ocorrência), (|) ou (*) zero ou mais ocorrências, (+) uma ou mais ocorrências. A ordem dos elementos filhos não é especificada nas tabelas.

O descrito em 5.4 a 5.26 discute valores e diretrizes de operação para cada módulo.

5.2 Perfil EDTV NCL 3.0

As Tabelas 1 a 20 descrevem os elementos e atributos definidos pelo perfil EDTV.

Tabela 1 — Elementos e atributos do módulo Structure estendido, utilizados no perfil

Elementos	Atributos	Conteúdo
ncl	*id, title, xmlns*	(head?, body?)
head		(importedDocumentBase?, ruleBase?, transitionBase?, regionBase*, descriptorBase?, connectorBase?, meta*, metadata*)
body	*id*	(port | property | media | context | switch| link | meta | metadata)*

Tabela 2 — Elementos e atributos do módulo *Layout* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
regionBase	*id, device, region*	(importBase|region)+
region	*id, title, left, right, top, bottom, height, width, zIndex*	(region)*

Tabela 3 — Elementos e atributos do módulo *Media* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
media	*id, src, refer, instance, type, descriptor*	(area|property)*

Tabela 4 — Elementos e atributos do módulo *Context* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
context	*id, refer*	(port|property|media|context|link|switch|meta|metadata)*

Tabela 5 — Elementos e atributos do módulo *MediaContentAnchor* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
area	*id, coords, begin, end, beginText, beginPosition, endText, endPosition, first, last, label, clip*	vazio

Tabela 6 — Elementos e atributos do módulo *CompositeNodeInterface* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
port	*id, component, interface*	vazio

Tabela 7 — Elementos e atributos do módulo *PropertyAnchor* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
property	*name, value, externable*	vazio

Tabela 8 — Elementos e atributos do módulo *SwitchInterface* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
switchPort	*id*	mapping+
mapping	*component, interface*	vazio

Tabela 9 — Elementos e atributos do módulo *Descriptor* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
descriptor	*id, player, explicitDur, region, freeze, moveLeft, moveRight, moveUp, moveDown, focusIndex, focusBorderColor, focusBorderWidth, focusBorderTransparency, focusSrc,focusSelSrc, selBorderColor, transIn, transOut*	(descriptorParam)*
descriptorParam	*name, value*	vazio
descriptorBase	*id*	(importBase|descriptor|descriptorSwitch)+

Tabela 10 — Elementos e atributos do módulo *Connector* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
bind	*role, component, interface, descriptor*	(bindParam)*
bindParam	*name, value*	vazio
linkParam	*name, value*	vazio
link	*id, xconnector*	(linkParam*, bind+)

Tabela 11 — Elementos e atributos do módulo *CausalConnectorFunctionality* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
causalConnector	*id*	(connectorParam*, (simpleCondition | compoundCondition), (simpleAction | compoundAction))
connectorParam	*name, type*	vazio
simpleCondition	*role, delay, eventType, key, transition, min, max, qualifier*	vazio
compoundCondition	*operator, delay*	((simpleCondition | compoundCondition)+, (assessmentStatement | compoundStatement)*)
simpleAction	*role, delay, eventType, actionType, value, min, max, qualifier, repeat, repeatDelay, duration, by*	vazio
compoundAction	*operator, delay*	(simpleAction | compoundAction)+
assessmentStatement	*comparator*	(attributeAssessment, (attributeAssessment | valueAssessment))
attributeAssessment	*role, eventType, key, attributeType, offset*	vazio
valueAssessment	*value*	vazio
compoundStatement	*operator, isNegated*	(assessmentStatement | compoundStatement)+

Tabela 12 — Elementos e atributos do módulo *ConnectorBase* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
connectorBase	*id*	((importBase|causalConnector)*

Tabela 13 — Elementos e atributos do módulo *TestRule* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
ruleBase	*id*	(importBase|rule|compositeRule)+
rule	*id, var, comparator, value*	vazio
compositeRule	*id, operator*	(rule | compositeRule)+

Tabela 14 — Elementos e atributos do módulo *TestRuleUse* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
bindRule	*constituent*, *rule*	vazio

Tabela 15 — Elementos e atributos do módulo *ContentControl* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
switch	*id*, *refer*	defaultComponent?, (switchPort | bindRule | media | context | switch)*)
defaultComponent	*component*	vazio

Tabela 16 — Elementos e atributos do módulo *DescriptorControl* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
descriptorSwitch	*id*	(defaultDescriptor?, (bindRule | descriptor)*)
defaultDescriptor	*descriptor*	vazio

Tabela 17 — Elementos e atributos do módulo *Import* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
importBase	*alias*, *documentURI*, *region*, *baseId*	vazio
importedDocumentBase	*id*	(importNCL)+
importNCL	*alias*, *documentURI*	vazio

Tabela 18 — Elementos e atributos do módulo *TransitionBase* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
transitionBase	*id*	(importBase, transition)+

Tabela 19 — Elementos e atributos do módulo *BasicTransition* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
transition	*id*, *type*, *subtype*, *dur*, *startProgress*, *endProgress*, *direction*, *fadeColor*, *horRepeat*, *vertRepeat*, *borderWidth*, *borderColor*	vazio

Tabela 20 — Elementos e atributos do módulo *Metainformation* estendido, utilizados no perfil EDTV

Elementos	Atributos	Conteúdo
meta	*name*, *content*	vazio
metadata	*empty*	RDF tree

5.3 Perfil BDTV NCL 3.0

As Tabelas 21 a 37 descrevem os elementos e atributos definidos pelo perfil BDTV.

Tabela 21 — Elementos e atributos do módulo *Structure* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
ncl	*id, title, xmlns*	(head?, body?)
head		(importedDocumentBase? ruleBase?, regionBase*, descriptorBase?, connectorBase?),
body	*id*	(port| property| media|context|switch|link)*

Tabela 22 — Elementos e atributos do módulo *Layout* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
regionBase	*id, device, region*	(importBase|region)+
Region	*id, title, left, right, top, bottom, height, width, zIndex*	(region)*

Tabela 23 — Elementos e atributos do módulo *Media* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
media	*id, src, refer, instance, type, descriptor*	(area|property)*

Tabela 24 — Elementos e atributos do módulo *Context* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
context	*id, refer*	(port|property|media|context|link|switch)*

Tabela 25 — Elementos e atributos do módulo Media*ContentAnchor* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
area	*id, coords, begin, end, beginText, beginPosition, endText, endPosition, first, last, label, clip*	vazio

Tabela 26 — Elementos e atributos do módulo *CompositeNodeInterface* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
port	*id, component, interface*	vazio

Tabela 27 — Elementos e atributos do módulo *PropertyAnchor* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
property	*name, value*	vazio

Tabela 28 — Elementos e atributos do módulo *SwitchInterface* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
switchPort	*id*	mapping+
mapping	*component, interface*	vazio

Tabela 29 — Elementos e atributos do módulo *Descriptor* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
descriptor	*id, player, explicitDur, region, freeze, moveLeft, moveRight, moveUp, moveDown, focusIndex, focusBorderColor; focusBorderWidth, focusBorderTransparency, focusSrc,focusSelSrc, selBorderColor*	(descriptorParam)*
descriptorParam	*name, value*	vazio
descriptorBase	*id*	(importBase | descriptor | descriptorSwitch)+

Tabela 30 — Elementos e atributos do módulo *Connector* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
bind	*role, component, interface, descriptor*	(bindParam)*
bindParam	*name, value*	vazio
linkParam	*name, value*	vazio
link	*id, xconnector*	(linkParam*, bind+)

Tabela 31 — Elementos e atributos do módulo *CausalConnectorFunctionality* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
causalConnector	*id*	(connectorParam*, (simpleCondition | compoundCondition), (simpleAction | compoundAction))
connectorParam	*name, type*	vazio
simpleCondition	*role, delay, eventType, key, transition, min, max, qualifier*	vazio
compoundCondition	*operator, delay*	((simpleCondition | compoundCondition)+, (assessmentStatement | compoundStatement)*)
simpleAction	*role, delay, eventType, actionType, value, min, max, qualifier, repeat, repeatDelay*	vazio
compoundAction	*operator, delay*	(simpleAction | compoundAction)+
assessmentStatement	*comparator*	(attributeAssessment, (attributeAssessment | valueAssessment))
attributeAssessment	*role, eventType, key, attributeType, offset*	vazio
valueAssessment	*value*	vazio
compoundStatement	*operator, isNegated*	(assessmentStatement | compoundStatement)+

Tabela 32 — Elementos e atributos do módulo *ConnectorBase* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
connectorBase	*id*	(importBase|causalConnector)*

Tabela 33 — Elementos e atributos do módulo *TestRule* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
ruleBase	*id*	(importBase|rule|compositeRule)+
rule	*id*, *var*, *comparator*, *value*	vazio
compositeRule	*id*, *operator*	(rule | compositeRule)+

Tabela 34 — Elementos e atributos do módulo *TestRuleUse* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
bindRule	*constituent*, *rule*	vazio

Tabela 35 — Elementos e atributos do módulo *ContentControl* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
switch	*id*, *refer*	(defaultComponent?,(switchPort| bindRule|media| context | switch)*)
defaultComponent	*component*	vazio

Tabela 36 — Elementos e atributos do módulo *DescriptorControl* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
descriptorSwitch	*id*	(defaultDescriptor?, (bindRule | descriptor)*)
defaultDescriptor	*descriptor*	vazio

Tabela 37 — Elementos e atributos do módulo de *Import* estendido, utilizados no perfil BDTV

Elementos	Atributos	Conteúdo
importBase	*alias*, *documentURI*, *region*	vazio
importedDocumentBase	*id*	(importNCL)+
importNCL	*alias*, *documentURI*,	vazio

5.4 Módulo Structure

5.4.1 Considerações gerais

O atributo *xmlns* do elemento <ncl> declara um *namespace* XML, ou seja, declara o grupo primário de construções XML que o documento utiliza. São permitidos três valores para o atributo *xmlns*: http://www.ncl.org.br/NCL3.0/EDTVProfile, http://www.ncl.org.br/NCL3.0/BDTVProfile e http://www.ncl.org.br/NCL3.0/CausalConnectorProfile, para os perfis TVD avançado, TVD básico e conector causal, respectivamente. Um formatador NCL deve obrigatoriamente saber que o schemaLocation para esses namespaces são, por *default*, respectivamente:

http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd,
http://www.ncl.org.br/NCL3.0/profiles/NCL30BDTV.xsd, e
http://www.ncl.org.br/NCL3.0/profiles/NCL30CausalConnector.xsd

5.4.2 Valores *default*

Não há nenhum valor *default*.

5.4.3 Tratamento de exceções

Recomenda-se que documentos com atributo *xmlns* diferente dos três valores mencionados anteriormente sejam ignorados em uma implementação de acordo com as ABNT NBR 15606-2 e ABNT NBR 15606-5.

Recomenda-se que documentos com atributos id, cujos valores não são sequências de caracteres compatíveis com a produção NCName [Namespaces in XML: 1999] sejam ignorados em uma implementação de acordo com as ABNT NBR 15606-2 e ABNT NBR 15606-5.

5.5 Módulo Layout

5.5.1 Considerações gerais

Cada elemento <regionBase> está associado a uma classe de dispositivos onde a apresentação ocorre. Com o intuito de identificar a associação, o elemento <regionBase> define o atributo *device*, que pode ter valores: “systemScreen (i)” ou “systemAudio(i)”, onde *i* é um número inteiro maior ou igual a 1.

O suporte a múltiplos dispositivos é definido nas diretrizes estabelecidas na Nested Context Language 3.0, Part 12.

No SBTVD, o systemScreen (1) e o systemAudio (1) são reservados para as classes passivas e o systemScreen (2) e systemAudio (2) são reservados para as classes ativas.

5.5.2 Valores default

O elemento <regionBase> que define uma classe passiva também pode ter um atributo *region*. Se o atributo não for especificado, a exibição ocorre apenas nos dispositivos da classe passiva.

Quando uma região aninhada não especificar um valor de posicionamento ou de tamanho, e esse valor não puder ser calculado a partir de outros atributos, deve-se obrigatoriamente assumir o mesmo valor absoluto do atributo da geometria-pai correspondente. Em particular, quando uma região de primeiro nível não especifica nenhum valor de posicionamento ou de tamanho, é adotado o valor de toda a área de apresentação do dispositivo.

Quando não especificado, o atributo *zIndex* é definido como zero.

5.5.3 Tratamento de exceções

Quando o usuário especifica os valores para *top*, *bottom* e *height* para a mesma <region>, podem ocorrer inconsistências espaciais. Nesse caso, os valores *top* e *height* devem obrigatoriamente ter prioridade sobre o valor *bottom.* Analogamente, quando o usuário especifica valores inconsistentes para os atributos *left*, *right* e *width* do elemento <region>, os valores *left* e *width* serão utilizados para calcular um novo valor *right*.

As regiões-filhas devem obrigatoriamente estar contidas por completo na área estabelecida por suas regiões-pai. Quando parte da região-filha se encontra fora de sua região-pai, a região-filha deve ser ignorada (considerada como se não fosse especificada).

5.6 Módulo Media

5.6.1 Considerações gerais

O módulo Media define tipos básicos de objetos de mídia. Para tanto, esse módulo define o elemento <media>. Cada objeto de mídia tem dois atributos principais, além de seu atributo *id*: *src*, que define a URI do conteúdo do objeto, e *type*, que define o tipo de objeto.

Observar que os objetos de mídia com o mesmo valor para *src* e com o esquema URI diferente de “ncl-mirror” possuem o mesmo conteúdo a ser apresentado, porém são duas instâncias de objetos de mídia. Como consequência, o conteúdo de cada um dos objetos pode ter sua apresentação iniciada em momentos diferentes, dependendo de quando os objetos de mídia forem iniciados, e suas exibições são totalmente independentes. Por outro lado, se o esquema de URI for igual ao “ncl-mirror”, o objeto de mídia cujo atributo *src* define esse esquema e o objeto de mídia referenciado pelo esquema devem obrigatoriamente ter o mesmo conteúdo em apresentação e no mesmo instante do tempo, se os dois objetos de mídia estiverem sendo apresentados, independentemente de quando iniciaram. Por serem objetos de mídia diferentes, suas propriedades podem ter valores diferentes, por exemplo, as que especificam a localização da apresentação. Deve-se salientar que a relação de espelhamento é reflexiva, simétrica e transitiva.

O sistema “ncl-mirror” não pode se referir a um elemento de <media> dos tipos application/x-ncl-NCL, application/x-ncl-NCLet e application/x-ncl-NCLua.

5.6.2 Objeto de mídia contínua em fluxos elementares TS

Se mais de um objeto de mídia que tem o atributo *src* com um mesmo valor referente a um conteúdo transportado no fluxo de transporte (TS) for iniciado, deve-se obrigatoriamente iniciar mais de uma apresentação. Além disso, como de costume, esses objetos podem ter diferentes regiões de apresentação, que podem ser redimensionadas por uma aplicação NCL. No entanto, deve-se observar que o número de objetos de mídia que podem ser apresentados no plano de vídeo SBTVD depende da implementação do receptor. Dos exibidores H.264 implementados no *hardware*, exige-se apenas uma apresentação de conteúdo por vez. A permissão de mais de um objeto de mídia de vídeo no plano de vídeo é opcional. Se o número de objetos de mídia de um determinado tipo exceder o número máximo permitido, a exibição dos objetos de mídia excedentes deve obrigatoriamente ser ignorada.

Com relação ao plano de vídeo do dispositivo-base de exibição, se, e somente se, não houver nenhum objeto de mídia sendo apresentado nesse plano, referindo-se (por meio de seu atributo *src*) a um fluxo elementar de vídeo de um serviço sintonizado (não importando em qual aplicação da base privada que representa esse canal de TV), deve-se obrigatoriamente apresentar um vídeo ES em tela cheia, que não está no momento sendo referenciado por nenhum objeto de mídia em exibição. Esse vídeo ES é aquele referenciado pelo último objeto de mídia que teve sua apresentação interrompida no plano de vídeo, ou então o vídeo principal ES do fluxo de transporte sintonizado, conforme definido na ABNT NBR 15604.

Se, e somente se, não houver qualquer objeto de mídia de áudio sendo apresentado no dispositivo-base de exibição, referindo-se (por meio de seu atributo *src*) a um fluxo de áudio de um serviço sintonizado (objeto este em qualquer aplicação da base privada que representa um canal de TV), um áudio ES, que não está no momento sendo referenciado por nenhum objeto de mídia em exibição, deve obrigatoriamente ser apresentado. Esse áudio ES é aquele referenciado pelo último objeto de mídia que teve sua apresentação interrompida, ou então o áudio principal ES do fluxo de transporte sintonizado, conforme definido na ABNT NBR 15604.

5.6.3 Tipos especiais de objetos NCL

5.6.3.1 Considerações gerais

Cinco tipos especiais de objetos de mídia NCL são definidos: application/x-ginga-NCL (ou application/x-ncl-NCL); application/x-ginga-NCLua (ou application/x-ncl-NCLua); application/x-ginga-NCLet (ou application/x-ncl-NCLet), que é opcional no Ginga para receptores portáteis; application/x-ginga-settings (ou application/x-ncl-settings); e application/x-ginga-time (ou application/x-ncl-time).

Quatro objetos importantes são aqueles que permitem códigos imperativos ou hipermídia declarativos integrados em um documento NCL: text/html; application/x-ginga-NCL (ou application/x-ncl-NCL); application/x-ginga-NCLua (ou application/x-ncl-NCLua); application/x-ginga-NCLet (ou application/x-ncl-NCLet), que é opcional no Ginga para receptores portáteis. Como tal, são discutidos em 5.6.3.2 a 5.6.3.4. A apresentação dos objetos de mídia é apresentada na Seção 6.

5.6.3.2 Objetos hipermídia declarativos integrados em apresentações NCL

5.6.3.2.1 Considerações gerais

Objetos hipermídia declarativos (com código NCL ou codificados com outra linguagem declarativa) podem ser inseridos em documentos NCL. A maneira de adicionar um objeto hipermídia declarativo em um documento NCL é definir um elemento <media>, cujo conteúdo (localizado por meio do atributo *src*) é o código declarativo a ser executado. Ambos os perfis EDTV e BDTV da NCL 3.0 permitem que o elemento <media> dos tipos text/html e application/x-Ginga-NCL (ou application/x-ncl-NCL) sejam aninhados em um documento NCL.

5.6.3.2.2 Objetos NCL integrados em aplicações NCL

Objetos NCL aninhados em aplicações NCL seguem as diretrizes estabelecidas em Nested Context Language 3.0, Part 11.

5.6.3.2.3 Objetos XHTML integrados em aplicações NCL

Qualquer implementação de objeto de mídia com base em XHTML, de acordo com a ABNT NBR 15606, deve obrigatoriamente oferecer suporte a todas as marcações (markups) XML e propriedades de folhas de estilo (stylesheets) comuns à BML para serviços básicos ("fixed terminal profile"), ACAP-X e DVB-HTML, conforme definido na ITU-T J.201. As funcionalidades comuns dos objetos ECMAScript nativos e API DOM são opcionais.

Embora um *browser* baseado em XHTML deva ser suportado, a utilização de elementos XHTML para definir relacionamentos (inclusive links XHTML e eventos de fluxo) é desaconselhada na autoria de documentos NCL.

Os objetos baseados em HTML aninhados em aplicações NCL seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 11.

5.6.3.3 Objetos codificados em Lua integrados em aplicações NCL

Os objetos NCLua integrados em aplicações NCL seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 10.

5.6.3.4 Objetos codificados em Java integrados em aplicações NCL

Os objetos NCLet integrados em aplicações NCL seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 10.

Quando um Xlet é referido como um arquivo ".class" em um elemento <media>, um *classpath* alternativo pode ser definido por meio de um elemento <property> do objeto, com o atributo *name* com o valor "x-classpath" e o atributo *value* tendo o caminho relativo para o *classpath*.

NOTA 1 Objetos NCLet são opcionais em receptores *one-seg*.

NOTA 2 Aplicações Xlet podem ser encapsuladas em arquivos JAR, se elas vierem pelo canal de interatividade. Nesse caso, um arquivo *manifest* precisa estar presente (caminho “META-INF/MANIFEST.MF”) com o parâmetro “Main-Class” se referindo ao Xlet principal.

5.6.3.5 Objeto do tipo application/x-ncl-settings

O tipo application/x-ginga-settings (ou application/x-ncl-settings) é aplicado a um elemento <media> especial, cujas propriedades são variáveis globais definidas pelo autor do documento ou variáveis de ambiente reservadas, que podem ser manipuladas pelo processamento do documento NCL.

A propriedade *user.location* depende de qual país adote o Ginga. Ela deve obrigatoriamente ser o código do país concatenado com o código postal do país. A especificação do código do país deve obrigatoriamente seguir o formato ISO 3166-1 alfa 3. No caso de uma implementação para o Brasil, o número do código postal do país (CEP) deve obrigatoriamente ser especificado sem caracteres de hífen, sublinhado (_) ou barra (/) e seguir o Guia Postal Brasileiro da Empresa Brasileira de Correios e Telégrafos.

Uma nova propriedade reservada é adicionada ao grupo *system* do objeto de mídia *settings* em receptores portáteis: a propriedade *system.screenOrientation*. Esta propriedade recebe o valor “portrait” ou “landscape”, definindo a orientação da tela.

Uma outra propriedade reservada deve ser adicionada ao grupo “*system*” do objeto de mídia Settings: system.luaSupportedEventClasses. Essa propriedade recebe como valor uma lista das classes de eventos a que NCLua oferece suporte, separadas por “,”. Entre os valores possíveis para essa variável de ambiente, a serem separados por vírgulas, estão: key, pointer, ncl, edit, tcp, sms, si e user (com as palavras em letras minúsculas).

Ainda uma outra propriedade reservada deve ser adicionada ao grupo “*system*” do objeto de mídia Settings: system.gingaNCLProfile. Essa propriedade recebe como valor o nome do perfil NCL a que o conversor digital oferece suporte. Entre os valores possíveis para essa variável de ambiente estão: EDTV, e BDTV (com as palavras em letras maiúsculas)

5.6.4 Valores *default*

O atributo *type* é opcional (exceto para os elementos <media> sem atributo *src* definido) e é usado para orientar a escolha do exibidor (ferramenta de apresentação) pelo formatador. Quando o atributo *type* não é especificado, o formatador deve obrigatoriamente utilizar a especificação de extensão de conteúdo no atributo *src* para escolher o exibidor.

Para objetos de mídia com o atributo *src*, cujo valor identifica o esquema “sbtvd-ts”, a parte específica do esquema, mais precisamente, o “program_number.component_tag”, pode ser substituído pelas palavras reservadas dadas na Tabela 38.

Tabela 38 — Palavras reservadas para a parte específica do esquema sbtvd-ts

Palavra reservada	Descrição
video	Correspondente ao vídeo ES principal que está sendo apresentado no plano de vídeo, conforme definido pela ABNT NBR 15604
audio	Correspondente ao áudio ES principal, conforme definido pela ABNT NBR 15604
text	Correspondente ao texto ES principal, conforme definido pela ABNT NBR 15604
video(i)	Correspondente ao i-ésimo menor vídeo ES *component_tag* listado na PMT dos serviços sintonizados

Tabela 38 (continuação)

Palavra reservada	Descrição
audio(i)	Correspondente ao i-ésimo menor áudio ES *component_tag* listado na PMT dos serviços sintonizados
text(i)	Correspondente ao i-ésimo menor texto ES *component_tag* listado na PMT dos serviços sintonizados

No objeto de mídia do tipo application/x-ginga-settings (ou application/x-ncl-settings), as propriedades system.ncl. version, system.GingaNCL.version e system.GingaJ.version, recebem como default os valores 3.0, 1.0 e 1.0, respectivamente.

Na definição do teclado virtual na propriedade channel.keyboardBounds, é permitido o uso de um teclado virtual de tamanho prefixado pelo fabricante do receptor.

5.6.5 Tratamento de exceções

As referências a recursos de *streaming* de vídeo ou de áudio não podem causar sintonização. As referências que implicam em sintonização para acesso a um recurso devem obrigatoriamente ser tratadas como se os recursos não estivessem disponíveis.

Recomenda-se que toda ação sobre um elemento <media> representando um recurso indisponível seja ignorada pelo formatador NCL. Recomenda-se que qualquer *condition* ou *assessment* com base em um elemento <media> representando um recurso indisponível recomenda-se que seja considerada falsa.

Se o número de objetos de mídia de um determinado tipo exceder o número máximo permitido para aquele tipo em um dispositivo de exibição específico, recomenda-se que a exibição dos objetos de mídia excedentes seja ignorada.

Se o arquivo de origem associado a um nó de mídia for atualizado em um carrossel de objetos, e se o nó de mídia não estiver sendo apresentado quando o arquivo atualizado chegar, o novo arquivo substitui o anterior. Portanto, quando o nó de mídia for iniciado após a atualização ser concluída, ele carrega os conteúdos do arquivo atualizado. Se o nó de mídia for iniciado durante a atualização, ele carrega os conteúdos da última versão do arquivo completo recebido. Essa versão é mantida durante todo o processo de atualização, sendo substituída apenas no final. Se o arquivo associado a um nó de mídia for atualizado em um carrossel de objetos, e se o nó de mídia já estiver sendo apresentado quando o arquivo atualizado chegar, a apresentação do nó de mídia não é afetada. O arquivo atualizado só é utilizado em apresentações futuras.

Se um elemento <media> tiver um atributo *src* que especifique o esquema “ncl-mirror” e se esse esquema fizer referência a um elemento <media> de application/x-ncl-NCL, ou application/x-ncl-NCLet, ou application/x-ncl-NCLua, recomenda-se que o elemento <media> seja ignorado.

5.7 Módulo Context

Não há comentários que complementem ou esclareçam a ABNT NBR 15606-2.

5.8 Módulo MediaContentAnchor

5.8.1 Considerações gerais

O módulo MediaContentAnchor define o elemento <area>.

Para objetos de mídia do tipo texto, os atributos *beginText* e *beginPosition* especificam o início do texto âncora. O final do texto âncora pode ser especificado utilizando-se os atributos *endTex* e *endPosition*, que também definem uma sequência e a ocorrência da sequência no texto, respectivamente.

EXEMPLO Supondo o conteúdo do texto a seguir: AAA AA AA AAA e os atributos *beginText*=”AA”.

Para *beginPosition* = 1, as âncoras começam na sequência sublinhada e em negrito: **AA**A AA AA AAA

Para *beginPosition* = 2, as âncoras começam na sequência sublinhada e em negrito: A**AA** AA AA AAA

Para *beginPosition* = 3, as âncoras começam na sequência sublinhada e em negrito: AAA **AA** AA AAA

Para elementos <area> com atributos *first* e *last*, quando seus valores forem especificados em NPT, eles se referem à base temporal especificada pelo atributo *contentId* do mesmo elemento <media> que contém o elemento <área>.

Para objetos de mídia do tipo application/x-ginga-NCL (ou application/x-ncl-NCL), os valores dos atributos *clip* e *label* seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 11.

Para objetos de mídia do tipo application/x-ginga-NCLua (ou application/x-ncl-NCLua), os valores do atributo *label* seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 10.

5.8.2 Valores *default*

Na NCL, cada nó (elementos de <media>, <body> ou <context>) tem uma âncora com uma região que representa todo o seu conteúdo. Essa âncora é denominada *whole content anchor* e é declarada por *default* na NCL. Com exceção do elemento de mídia com código imperativo (por exemplo, <media type=“application/x-ginga-NCLua” ...>), cada vez que um componente NCL é referenciado sem especificar uma de suas âncoras, deve-se obrigatoriamente assumir a *whole content anchor.*

Em âncoras de conteúdo temporal, se o atributo *begin* de um elemento <area> for definido, mas o atributo *end* não for especificado, o final de toda a apresentação do conteúdo de mídia deve obrigatoriamente ser considerado como o final da âncora. Por outro lado, se o atributo *end* for definido, mas sem uma definição explícita de *begin*, o início de toda a apresentação do conteúdo de mídia deve obrigatoriamente ser considerado como o início da âncora. Espera-se um funcionamento semelhante dos atributos *first* e *last*. No caso de um elemento <media> do tipo application/x-ginga-time, os atributos *begin* e *end* devem obrigatoriamente ser definidos e devem obrigatoriamente assumir um valor absoluto do Tempo Universal Coordenado (UTC).

Em âncoras de conteúdo textual, se o final da região da âncora não for definido, deve-se obrigatoriamente assumir o final do conteúdo do texto.

Em âncoras de conteúdo textual, se o início da região da âncora não for definido, deve-se obrigatoriamente assumir o início do conteúdo do texto.

5.8.3 Tratamento de exceções

Não há comentários que complementem ou esclareçam a ABNT NBR 15606-2.

5.9 Módulo PropertyAnchor

5.9.1 Considerações gerais

O elemento <property> define o atributo *name*, que indica o nome da propriedade ou grupo de propriedades.

Os elementos <body>, <context> e <media> podem ter várias propriedades (ver ABNT NBR 15606-2) que não são explicitamente declaradas. No entanto, quando uma propriedade é utilizada em um relacionamento, ela deve obrigatoriamente ser explicitamente declarada em um elemento <property> (interface). Portanto, cada propriedade possui um atributo booleano denominado *externable,* que é definido como “false” por *default* e como “true” quando a propriedade é explicitamente declarada.

As propriedades que têm como valor strings com nomes de cores reservadas (“white”, “black”, “silver”, “gray”, red”, “maroon”, fuchsia”, “purple”, “lime”, “green”, “yellow”, “olive”, “blue”, “navy”, “aqua”, ou “teal”) seguem o padrão de cor CSS1, conforme definido na Tabela 39.

Tabela 39 — Definição das palavras reservadas para cores

	Nome	Hexadecimal	R	G	B	Matiz	Saturação Light	Saturação Value
	White	#FFFFFF	100 %	100 %	100 %	0°	0 % 100 %	0 % 100 %
	Silver	#C0C0C0	75 %	75 %	75 %	0°	0 % 75 %	0 % 75 %
	Gray	#808080	50 %	50 %	50 %	0°	0 % 50 %	0 % 50 %
	Black	#000000	0 %	0 %	0 %	0°	0 % 0 %	0 % 0 %
	Red	#FF0000	100 %	0 %	0 %	0°	100 % 50 %	100 % 100 %
	Maroon	#800000	50 %	0 %	0 %	0°	100 % 25 %	100 % 50 %
	Yellow	#FFFF00	100 %	100 %	0 %	60°	100 % 50 %	100 % 100 %
	Olive	#808000	50 %	50 %	0 %	60°	100 % 25 %	100 % 50 %
	Lime	#00FF00	0 %	100 %	0 %	120°	100 % 50 %	100 % 100 %
	Green	#008000	0 %	50 %	0 %	120°	100 % 25 %	100 % 50 %
	Aqua	#00FFFF	0 %	100 %	100 %	180°	100 % 50 %	100 % 100 %
	Teal	#008080	0 %	50 %	50 %	180°	100 % 25 %	100 % 50 %
	Blue	#0000FF	0 %	0 %	100 %	240°	100 % 50 %	100 % 100 %
	Navy	#000080	0 %	0 %	50 %	240°	100 % 25 %	100 % 50 %
	Fuchsia	#FF00FF	100 %	0 %	100 %	300°	100 % 50 %	100 % 100 %
	Purple	#800080	50 %	0 %	50 %	300°	100 % 25 %	100 % 50 %

O valor da propriedade *contentId* (associada a um objeto de mídia contínua cujo conteúdo é definido fazendo-se referência a um fluxo elementar) é definido pelo *middleware*. Inicialmente, ele deve obrigatoriamente ter o valor "null" e deve obrigatoriamente assumir o valor do identificador transportado no descritor de referência NPT (em um campo identificado pelo mesmo nome: contentId), assim que o objeto de mídia contínua associado é iniciado.

O valor da propriedade *standby* também é definido pelo *middleware*. Ele deve obrigatoriamente ter "true" como valor, enquanto um objeto de mídia contínua já iniciado e que faz referência a um fluxo elementar estiver temporariamente interrompido por outro conteúdo intercalado no mesmo fluxo elementar.

NOTA A propriedade *standby* recebe o valor "true" automaticamente pelo *middleware*, quando o valor do identificador transportado no descritor de referência NPT (em um campo identificado pelo mesmo nome: contentId) sinalizado como não pausado for diferente do valor da propriedade *contentId*.

A propriedade *standby* pode ser usada para pausar uma aplicação, quando o conteúdo do objeto de mídia contínua que faz referência a um fluxo elementar transportando o vídeo principal de um programa de TV for temporariamente interrompido por outro conteúdo intercalado, por exemplo, uma propaganda (comercial de TV). A mesma propriedade pode ser utilizada para retomar a aplicação.

Quando a propriedade *visible,* associada a um elemento <context> ou <body>, for igual a “true”, a propriedade *visible* de cada elemento-filho da composição deve obrigatoriamente ser considerada para definir a forma como cada elemento-filho será exibido.

Quando a propriedade *visible,* associada a um elemento <context> ou <body> for igual a “false”, todos os elementos-filhos da composição são exibidos de forma oculta. Especificamente, quando um documento tem seu elemento <body> com a propriedade *visible* configurada como “false” e seu evento de apresentação está no estado *paused*, diz-se que o documento está em espera (*stand-by)*. Quando há uma única aplicação em execução e essa aplicação está em espera, o vídeo principal do serviço deve obrigatoriamente ser dimensionado para 100 % da tela e o áudio principal deve obrigatoriamente ser configurado para 100 % de seu volume.

Deve-se observar que um objeto com propriedade *visible* igual a “false”, ou seja, um objeto em exibição mas oculto, não pode transicionar as máquinas de estados de eventos de seleção definidas pelas suas âncoras de conteúdo para o estado “occurring”, enquanto o valor da propriedade *visible* continuar como “false”.

Se as propriedades *left, right, top, bottom, width* ou *height*, definidas pelo element <property> ou pelo elemento <descriptorParam>, tiverem valores em porcentagem (%), os valores se referem ao tamanho da tela do dispositivo onde o objeto de mídia será exibido.

Nas propriedades *soundLevel, trebleLevel e bassLevel* de um objeto de mídia do tipo áudio, seus valores devem ser interpretados como relativos ao volume gravado do conteúdo do objeto de mídia. Estabelecer o valor como “0 %” causa a apresentação do conteúdo em silêncio. Um valor igual “100 %” apresenta o conteúdo com o valor gravado (0dB). O valor default para essas propriedades é “100 %”.

5.9.2 Valores *default*

O atributo *value* de um elemento <property> é opcional e define um valor inicial para a propriedade declarada em *name*. Quando o valor não é especificado, a propriedade assume como valor inicial aquele definido nos atributos homônimos do descritor ou região associados ao nó (objeto), onde a propriedade foi definida, ou então um valor *default*. Quando o atributo *value* é especificado, ele tem prioridade sobre o valor definido nos atributos homônimos do descritor ou região associados ao nó.

NOTA Todas as propriedades (e seus valores iniciais) de um objeto NCL podem ser definidas usando apenas elementos <property>. Os elementos <descriptor>, <descriptorParam> e <region> são apenas uma opção a mais (opção de reuso) para definir valores iniciais para as propriedades.

Se as propriedades *left, right, top, bottom, width* ou *height* não estiverem definidas e não puderem ser deduzidas a partir dos valores de propriedade definidos em <property>, <descriptor> e seus elementos-filhos, ou nos elementos <region>, elas devem obrigatoriamente assumir o valor “0”.

5.9.3 Tratamento de exceções

Quando o formatador trata uma mudança em um grupo de propriedades, ele deve testar a consistência do processo somente ao seu final.

Quando o usuário especifica as informações de *top*, *bottom* e *height* para o mesmo elemento <media>, podem ocorrer inconsistências espaciais. Nesse caso, os valores de *top* e *height* devem obrigatoriamente ter prioridade sobre o valor de *bottom*. Analogamente, quando o usuário especifica valores inconsistentes para as propriedades de *left*, *right* e *width*, os valores de *left* e *width* são utilizados para calcular um novo valor de *right*.

Quando as propriedades de *left, right, top, bottom, width* ou *height* excederem a dimensão do dispositivo de exibição, apenas parte do conteúdo dentro da dimensão do dispositivo é exibida.

Se dois ou mais elementos <property> com o mesmo atributo *name* forem definidos como elementos filhos do mesmo elemento <media>, deve-se obrigatoriamente considerar apenas o último *value* definido.

5.10 Módulo CompositeNodeInterface

O módulo CompositeNodeInterface define o elemento <port>, que especifica a porta de um nó de composição, com o seu respectivo mapeamento para uma interface (atributo *interface*) de um e apenas um de seus componentes-filhos (especificado pelo atributo *component*).

5.11 Módulo SwitchInterface

5.11.1 Considerações gerais

O módulo SwitchInterface possibilita a criação de interfaces do elemento <switch>, que podem ser mapeadas para um conjunto de interfaces alternativas dos objetos internos do switch, permitindo que um *link* ancore na interface escolhida, quando o <switch> é processado. Esse módulo introduz o elemento <switchPort>, que contém um conjunto de elementos *mapping*. Um elemento *mapping* define um caminho da <switchPort> para uma interface (atributo *interface*) de um dos componentes do switch (especificado pelo seu atributo *component*).

NOTA Um elemento <switchPort> pode definir um mapeamento para um subconjunto dos componentes do switch. Quando um elo referencia uma <switchPort> e nenhuma das regras, associadas aos componentes definidos por seus elementos filho <mapping>, for avaliada como verdadeira, o elemento <defaultComponent> é escolhido. Se o elemento <defaultComponent> não for definido, nenhum componente é selecionado para a apresentação.

5.11.2 Valores *default*

A referência a um componente interno a um switch deve obrigatoriamente ser feita por meio de um elemento <switchPort> ou, por *default*, ao elemento <switch> sem especificar qualquer <switchPort>. Nesse último caso, considera-se como se a referência fosse feita a um <switchPort> *default* que contém elementos <mapping> para cada objeto-filho do switch e referindo-se à sua *whole content anchor*.

5.12 Módulo Descriptor

5.12.1 Considerações gerais

Um elemento <descriptor> pode ter elementos-filhos <descriptorParam>, que são utilizados para parametrizar o controle da apresentação do objeto associado ao elemento descritor.

Um atributo *descriptor* pode ser associado a qualquer objeto de mídia por meio dos próprios elementos <media> ou por meio dos pontos terminais (elementos <bind>) dos elos (elementos <link>).

O parâmetro *plan* de um elemento <descriptorParam> define em qual plano de uma tela estruturada o objeto é posicionado. O valor desse atributo pode ser: “background”, “video” ou “graphic”, seguindo a definição de plano da ABNT NBR 15606-1.

Os atributos *reusePlayer* e *playerLife* oferecem suporte adicional para o gerenciamento do exibidor dos objetos de mídia e podem ser usados ou ignorados por um determinado implementador do *middleware*. O atributo *playerLife* especifica o que vai acontecer com uma instância do exibidor no final de uma apresentação do objeto de mídia. Manter uma instância do exibidor exige espaço na memória, porém diminui o tempo de carga do exibidor e a probabilidade de descasamentos na sincronização, como consequência. O atributo *reusePlayer* permite usar a mesma instância do exibidor na apresentação de mais de um objeto de mídia, incluindo o uso de um exibidor mantido no espaço de memória pelo atributo *playerLife*.

5.12.2 Valores *default*

A Tabela 40 resume alguns valores *default* para parâmetros-de-descritor/nomes-de-propriedades reservados.

Tabela 40 — Alguns parâmetros/propriedades reservados e seus valores *default*

Nome do parâmetro/propriedade	*Default*
top, left, width, height	Se qualquer valor dessas propriedades não for definido e não puder ser inferido das regras definidas pela especificação NCL, ele deve obrigatoriamente assumir o valor “0”
location	Ver a primeira linha desta tabela
size	Ver a primeira linha desta tabela
bounds	Ver a primeira linha desta tabela
plan	“video”, para objeto de mídia com o atributo *src* referindo um PES de um fluxo TS, “graphic”, para todos os outros casos.
baseDeviceRegion	
deviceClass	0
explicitDur	Para mídias contínuas deve ser assumido o valor da duração da apresentação natural do conteúdo; caso contrário, deve ser assumido o valor “nill”
background	transparent
visible	true
transparency	0
rgbChromakey	nill
fit	fill
scroll	none
style	nill
soundLevel, trebleLevel, bassLevel	1
balanceLevel	0
zIndex	0
fontColor	white
textAlign	left
fontFamily	
fontStyle	normal
fontSize	
fontVariant	normal
fontWeight	normal
player	
reusePlayer	false
playerLife	close
moveLeft, moveRight, moveUp, moveDown, focusIndex	nill
focusBorderColor;	O valor definido por *default.focusBorderColor*
selBorderColor	O valor definido por *default.selBorderColor*
focusBorderWidth	O valor definido por *default.focusBorderWidth*
focusBorderTransparency	O valor definido por *default.focusTransparency*
focusSrc, focusSelSrc	nill
freeze	false
transIn, transOut	string vazia

5.12.3 Tratamento de exceções

Se vários valores forem especificados para uma mesma propriedade, o valor definido em um elemento <property> tem prioridade sobre aquele definido em um elemento <descriptorParam>, que, por sua vez, tem prioridade sobre o valor definido em um atributo do elemento <descriptor> correspondente (inclusive o atributo *region*).

Recomenda-se que propriedades definidas utilizando-se CSS, que não são exigidas pela ABNT NBR 15606-2, sejam ignoradas pelo exibidor de mídia correspondente.

5.13 Módulo Linking

5.13.1 Valores *default*

O atributo *interface* de um elemento <bind> pode se referir a qualquer interface de um nó, ou seja, uma âncora, uma propriedade, uma porta (no caso de um nó de composição) ou uma switchPort (no caso de um switch). O atributo é opcional e, caso não seja especificado, a *whole content anchor* é assumida como a interface, exceto no caso de objetos de mídia com código imperativo, como detalhado em 6.4.1.

5.13.2 Tratamento de exceções

Recomenda-se ignorar um elemento <link> se o atributo *xconnector* referir-se a um conector hipermídia inexistente.

Se um elo definir um mesmo parâmetro usando os elementos <linkParam> e <bindParam>, a definição usando <bindParam> tem precedência.

Se o número de participantes especificado em um elemento <link> para uma condição especificada no elemento <causalConnector> referido for maior que o especificado no atributo *max* da condição, é recomendado que o elemento <link> seja ignorado.

Da mesma forma, se o número de participantes especificado em um elemento <link> para uma condição especificada no elemento <causalConnector> referido for menor que o especificado no atributo *min* da condição, é recomendado que o elemento <link> seja ignorado.

5.14 Funcionalidade Connectors

5.14.1 Considerações gerais

A funcionalidade Connectors define o módulo CausalConnectorFunctionality, que agrupa um conjunto de módulos básicos da Funcionalidade Connectors, a fim de facilitar a definição de um perfil de linguagem. Esse é o caso dos perfis EDTV e BDTV.

O elemento <compoundAction> de um conector possui o atributo *operator* (“par” ou “seq”) relacionando seus elementos-filhos. É importante mencionar que, ao utilizar o operador sequencial, as ações devem obrigatoriamente ser disparadas na ordem especificada. No entanto, uma ação não precisa esperar a conclusão da anterior para ser iniciada.

5.14.2 Valores *default*

Se o valor *eventType* de um elemento <simpleCondition> for “selection”, o papel também pode definir à qual dispositivo de seleção (por exemplo, teclado ou teclas de controle remoto) ele se refere, por meio de seu atributo *key*. Se esse atributo não for especificado, a seleção por meio de um dispositivo apontador (mouse, tela de toque teclas de navegação, conforme 5.23 etc.) deve obrigatoriamente ser assumida.

NOTA Quando um dispositivo de seleção é pressionado, mais de uma <simple condition> pode ser considerada satisfeita, caso esse dispositivo de seleção seja definido no atributo *key* das <simpleCondition> e caso as interfaces vinculadas pelos elementos <link> referindo-se à <simpleCondition> (por meio dos atributos *role* de seus elementos <bind>) estejam sendo apresentadas.

Em um elemento <simpleCondition> e em um elemento <simpleAction>, a cardinalidade do papel especifica o número mínimo (atributo *min*) e máximo (atributo *max*) de participantes que podem desempenhar o papel (número de *binds*), quando o <causalConnector> for usado para criar um <link>. Se as cardinalidades mínimas e máximas não forem informadas, deve-se obrigatoriamente assumir "1" como valor *default* para ambos os parâmetros.

Um atributo *qualifier* informa o relacionamento lógico entre os atores da mesma condição simples. Se não for especificado, deve-se obrigatoriamente assumir o valor *default* "or".

Um atributo *qualifier* informa o relacionamento lógico entre os atores da mesma ação simples. Se não for especificado, deve-se obrigatoriamente assumir o valor *default* "par".

Se o valor *eventType* de um elemento <attributeAssessment> for "selection", a função também pode definir à qual dispositivo de seleção (por exemplo, teclado ou teclas de controle remoto) ele se refere, por meio de seu atributo *key*. Se esse atributo não for especificado, a seleção por meio de um dispositivo apontador (mouse, tela de toque etc.) deve obrigatoriamente ser assumida.

Se o valor de *eventType* de um elemento <attributeAssessment> for "attribution", o *attributeType* é opcional e tem o valor "nodeProperty" como *default*.

5.14.3 Tratamento de exceções

O valor mínimo da cardinalidade do papel é sempre um valor positivo finito, maior que zero e menor ou igual ao valor máximo da cardinalidade; caso contrário, recomenda-se que o *link* seja ignorado.

Se o valor de *eventType* for "attribution", a <simpleAction> também deve obrigatoriamente definir o valor a ser atribuído, por meio de seu atributo *value*. Se *value* for especificado como "$anyName" (onde $ é um símbolo reservado e anyName é qualquer sequência de caracteres, exceto nomes de papéis reservados), o valor atribuído deve obrigatoriamente ser recuperado a partir da propriedade associada com *role*="anyName", definida por um elemento <bind> filho do elemento <link> que faz referência ao conector. Se esse valor não puder ser recuperado, nenhuma atribuição é feita.

Se o valor de *eventType* for "attribution" e a <simpleAction> definir o valor que será atribuído como "$anyName", o valor a ser atribuído deve obrigatoriamente ser o valor de uma propriedade (elemento <property>) de um componente da mesma composição, onde o relacionamento (elemento <link>) que se refere ao evento é definido, ou de uma propriedade da composição, onde o relacionamento é definido, ou de uma propriedade de um elemento que pode ser alcançada por meio de um elemento <port> da composição, onde o relacionamento é definido, ou até mesmo uma propriedade de um elemento que pode ser alcançada por meio de uma porta (elementos <port> ou <switchPort>) de uma composição aninhada na mesma composição onde o relacionamento é definido. Caso contrário, nenhuma atribuição pode ser feita.

Um elemento <attributeAssessment> define um atributo *offset* cujo valor pode ser adicionado ao valor da variável referida pelo atributo *attributeType* correspondente. O valor *offset* deve obrigatoriamente ter o mesmo tipo e ser especificado com a mesma unidade do valor ao qual será adicionado, caso contrário, recomenda-se que o valor *offset* seja ignorado.

O valor do atributo *comparator* do elemento <asessmentStatement> deve obrigatoriamente assumir os valores: "eq", "ne", "gt", "lt", "gte", ou "lte". É recomendado que, se qualquer outro valor for especificado para esse atributo, o elemento seja ignorado.

Os eventos de seleção só podem ser definidos sobre unidades de informação de um objeto de mídia que está sendo apresentado, ou seja, sobre unidades de informação cujo evento de apresentação associado esteja no status "occurring".

5.15 Módulo TestRule

5.15.1 Considerações gerais

O módulo TestRule permite a definição de regras que, quando cumpridas, selecionam alternativas para a apresentação do documento. Essas regras podem ser simples, definidas pelo elemento <rule>, ou compostas, definidas pelo elemento <compositeRule>.

As comparações em regras simples são feitas com base na representação binária do valor da variável a ser comparado e na representação binária do outro valor utilizado na comparação.

Um elemento <rule> definido como elemento-filho de <compositeRule> pode ter seu atributo *id* omitido.

5.15.2 Tratamento de exceções

O valor do atributo *comparator* do elemento <rule> deve obrigatoriamente assumir os valores: “eq”, “ne”, “gt”, “lt”, “gte”, ou “lte”. É recomendado que, se qualquer outro valor for especificado para esse atributo, o elemento seja ignorado.

Em regras simples, o atributo *comparator* relaciona a variável a um valor. O tipo da variável e do valor devem obrigatoriamente ser iguais; caso contrário, recomenda-se que a definição da regra seja ignorada.

5.16 Módulo TestRuleUse

Não há comentários que complementem ou esclareçam a ABNT NBR 15606-2.

5.17 Módulo ContentControl

5.17.1 Considerações gerais

O módulo ContentControl especifica o elemento <switch>. Os formatadores NCL devem obrigatoriamente postergar a avaliação do switch para o momento no qual um relacionamento (elo) ancorando no switch precisar ser avaliado. As regras de teste utilizadas para escolher o componente do switch a ser apresentado devem obrigatoriamente ser definidas pelo módulo TestRule. As regra são avaliadas na ordem em que são definidas.

Durante a apresentação do documento, a partir do momento em que um <switch> é avaliado em diante, ele é considerado solucionado até ao final de sua atual apresentação, ou seja, enquanto seu evento de apresentação correspondente se encontrar no estado “occurring” ou “paused”. A alternativa escolhida pode ser referenciada por meio de um elemento <switchPort> mapeado para uma de suas interfaces.

Quando um <context> é definido como um filho de um elemento <switch>, os elementos <link> recursivamente contidos em <context> devem ser considerados por um exibidor NCL somente se o <context> for selecionado após a avaliação do switch. Caso contrário, os elementos <link> devem obrigatoriamente ser considerados desativados e não podem interferir na apresentação do documento.

5.17.2 Tratamento de exceções

Se o elemento <defaultComponent> não for definido em um elemento <switch> e se nenhuma das regras bindRule for avaliada como verdadeira para um componente vinculado a um elemento <mapping>, filho do <switchPort> através do qual o elemento <switch> é referenciado, nenhum componente é selecionado para a apresentação e o formatador NCL funciona como se o componente não existisse.

5.18 Módulo DescriptorControl

5.18.1 Considerações gerais

O módulo DescriptorControl especifica o elemento <descriptorSwitch>. Analogamente ao elemento <switch>, a seleção do <descriptorSwitch> é feita utilizando regras de teste definidas no módulo TestRule. A avaliação do descriptorSwitch deve obrigatoriamente ser adiada até que o objeto que faz referência ao descriptorSwitch precise ser preparado para a apresentação.

Durante a apresentação do documento, a partir do momento em que o <descriptorSwitch> é avaliado por um elemento <media> específico, ele é considerado resolvido para aquele elemento <media> até o final da apresentação desse elemento <media>, ou seja, enquanto qualquer evento de apresentação associado a esse elemento <media> estiver no estado “occurring” ou “paused”.

5.18.2 Tratamento de exceções

Se o elemento <defaultDescriptor> não for definido em um elemento <descriptorSwitch> e se nenhuma das regras bindRule for avaliada como verdadeira, nenhum descritor pode ser selecionado para apresentação e o formatador NCL funciona como se o descritor não existisse.

5.19 Módulo Timing

5.19.1 Considerações gerais

Este módulo define o atributo *freeze* para especificar o que acontece a um elemento <media> no final de sua apresentação. Quando o *freeze* é especificado com um valor igual a “true”, o último mapa de imagem do objeto deve obrigatoriamente ser congelado, até que seu final seja determinado por um evento externo (por exemplo, proveniente da avaliação de um <link>), ou por meio de um valor *explicitDur* definido para aquele objeto.

O módulo Timing também define o atributo *explicitDur* especificando a duração da apresentação de um objeto representado por um elemento <media>. Observar que o atributo *explicitDur* fornece a duração da apresentação do objeto e não a duração da apresentação do conteúdo do objeto. Se o valor *explicitDur* for maior que a duração da apresentação do conteúdo, o que acontecerá no final da apresentação desse conteúdo depende do atributo *freeze* mencionado anteriormente. Se o valor *explicitDur* for menor que a duração da apresentação do conteúdo, a apresentação do conteúdo é interrompida. Observar que um exibidor pode, eventualmente, fazer ajustes de tempo flexíveis no conteúdo da mídia, a fim de tornar a duração da apresentação do conteúdo o mais próximo possível do valor de *explicitDur*.

5.19.2 Valores *default*

Quando não especificado, o valor do atributo *freeze* deve obrigatoriamente ser considerado como “false”.

Quando não especificado, o valor do atributo *explicitDur* deve obrigatoriamente ser considerado igual à duração normal da apresentação do conteúdo.

5.20 Módulo Import

5.20.1 Considerações gerais

O elemento <importNCL> não inclui o documento NCL referenciado, apenas torna o documento referenciado visível para ter seus componentes reutilizados pelo documento que definiu o elemento <importNCL>. Novos relacionamentos podem ser definidos entre novos nós criados por meio do reuso, mas não é possível definir novos relacionamentos dentro dos nós de contexto importados, pois, quando reutilizado, o nó não pode ter seu conteúdo modificado, podendo apenas ser relacionado a outros nós.

Quando se importa um documento, seu elemento <media> do tipo application/x-ginga-settings (ou application/x-ncl-settings) não tem qualquer influência sobre o elemento do mesmo tipo do documento importador, cujas propriedades são aquelas que têm validade para o documento importador.

5.20.2 Tratamento de exceções

A operação <importBase> é transitiva, ou seja, se a *baseA* importar a *baseB*, que importa a *baseC,* então a *baseA* importa a *baseC*. No entanto, o *alias* definido para a *baseC* dentro da *baseB* não pode ser considerado pela *baseA.*

A operação do elemento <importNCL> também tem a propriedade transitiva, ou seja, se o *documentA* importar o *documentB*, que importa o *documentC,* então o *documentA* importa o *documentC*. No entanto, o *alias* definido para o *documentC* dentro do *documentB* não pode ser considerado pelo *documentA*.

Por definição, não há recursão na operação de importação.

5.21 Módulo EntityReuse

5.21.1 Considerações gerais

O módulo EntityReuse permite a reutilização de um elemento NCL. Apenas <media>, <context>, <body> e <switch> podem ser reutilizados.

Quando um elemento declara um atributo *refer*, todos os atributos e elementos-filhos definidos pelo elemento referenciado são herdados. Para os elementos <media>, novos elementos-filhos <area> e <property> podem ser acrescentados e um novo atributo, *instance*, também pode ser definido. Todo elemento <media> comporta-se igualmente quanto ao reuso, incluindo o elemento com tipo igual a application/x-ginga-settings (ou application/x-ncl-settings).

O elemento referenciado e o elemento que se refere a ele devem obrigatoriamente ser considerados o mesmo em relação à especificação de seus dados.

5.21.2 Tratamento de exceções

Um elemento que se refere a outro elemento não pode ser reutilizado, ou seja, sua *id* não pode ser o valor de nenhum atributo *refer*. Quando um elemento tem o atributo *refer* com um valor correspondente à *id* de um elemento que se refere a outro, recomenda-se que o elemento seja considerado inexistente.

Se o nó referenciado for definido em um documento importado *D*, o valor do atributo *refer* deve ter o formato de "alias#id", onde “*alias*” é o valor do atributo *alias* associado à importação de *D*. Caso contrário, recomenda-se que o elemento que contém o atributo *refer* seja considerado inexistente.

Um elemento <media> só pode se referir a outro elemento <media>; um elemento <switch> só pode se referir a outro elemento <switch>, um elemento <context> só pode se referir a outro elemento <context> ou <body>. Em todos os outros casos, recomenda-se que o elemento que contém o atributo *refer* seja considerado inexistente.”

Todos os atributos e elementos-filhos definidos por um elemento que referencia um outro devem obrigatoriamente ser ignorados pelo formatador NCL, exceto o atributo *id*, que deve obrigatoriamente ser definido. A outra única exceção é para os elementos <media>, onde novos elementos filhos <area> e <property> podem ser acrescentados, e um novo atributo, *instance*, também pode ser definido.

Se o novo elemento <property> acrescentado possuir o mesmo atributo *name* de um elemento <property> já existente (definido no elemento <media> reutilizado), recomenda-se que a nova <property> acrescentada seja ignorada. Analogamente, se o novo elemento <area> acrescentado possuir o mesmo atributo *id* de um elemento

<area> já existente (definido no elemento <media> reutilizado), recomenda-se que a nova <area> acrescentada seja ignorada.

Os elementos <body>, <context> ou <switch> não podem incluir mais de um elemento do conjunto composto pelo objeto referente e pelos objetos referenciados correspondentes. Se este for o caso, recomenda-se que todos os objetos referenciados sejam considerados inexistentes.

5.22 Módulo ExtendedEntityReuse

5.22.1 Considerações gerais

O módulo ExtendedEntityReuse define o atributo *instance*.

O elemento referenciado e o elemento que se refere a ele são considerados objetos independentes com relação à sua apresentação, se o atributo *instance* receber o valor “new”.

O elemento referenciado e o elemento que se refere a ele serão considerados o mesmo objeto com relação à sua apresentação, se o atributo *instance* receber o valor “instSame” ou “gradSame”.

5.22.2Valores *default*

O atributo *instance* tem o valor “new” como seu valor *default*.

5.23 Módulo KeyNavigation

5.23.1 Considerações gerais

O módulo KeyNavigation fornece as extensões necessárias para descrever as operações de movimento de foco, usando um dispositivo de controle, como um controle remoto.

O atributo *focusIndex* especifica um índice para o elemento <media> no qual o foco pode ser aplicado. O *focusIndex* pode ser definido em um elemento <property> ou em um elemento <descriptor>.

Quando um elemento em foco é selecionado pressionando a tecla de ativação da navegação “(select, enter etc.)”, se houver um elemento <simpleCondition> com o atributo *role*=“onSelection” sem estar especificado o atributo *key*, essa condição é considerada como satisfeita se o elemento em foco for aquele especificado no atributo *component* do elemento <simpleCondition>. Note assim que as teclas de navegação operam de forma similar a um dispositivo apontador (como mouse etc.).

Quando um elemento em foco é selecionado pressionando a tecla de ativação “(select, enter etc.)”, o controle de foco deve obrigatoriamente ser passado para o renderizador (exibidor) do elemento <media>. O exibidor pode então seguir suas próprias regras de navegação. O controle do foco é retonado para o formatador NCL quando a tecla “back” for pressionada. Nesse caso, o foco vai para o elemento identificado pela propriedade *service.currentFocus* do nó de *settings* (elemento <media> do tipo application/x-ginga-settings). Em um ambiente com diversos dispositivos, as regras hierárquicas para o controle da tecla de entrada seguem as diretrizes estabelecidas na Nested Context Language 3.0, Part 12.

Quando um exibidor NCL começa a apresentação de uma aplicação, ele deve receber notificações do acionamento das teclas navegacionais “CURSOR_DOWN”, “CURSOR_LEFT”, “CURSOR_RIGHT”, “CURSOR_UP”, “ENTER” e “BACK, e das seleções a partir de um dispositivo ponteiro (por exemplo, um mouse, uma seleção em uma tela sensível ao toque etc.). A partir do momento em que o objeto em foco é selecionado (quando a tecla “ENTER” é pressionada ou o dispositivo ponteiro realiza a seleção), ou quando o valor do atributo service.currentKeyMaster do nó settings for alterado para um valor igual ao *id* de um nó de mídia válido, o exibidor deve parar de receber notificações das teclas navegacionais (e todas as demais teclas que ele anteriormente controlava), exceto da tecla “BACK”, da qual ele é ainda notificado. Todas as teclas anteriormente controladas,

incluindo a tecla “BACK” (e as navegações e seleções a partir de um dispositivo ponteiro), quando acionadas, devem notificar o exibidor do objeto de mídia em foco (a notificação do acionamento que provocou a seleção do exibidor do objeto de mídia, não pode ser notificada a ele). Notar assim que tanto o exibidor NCL quanto o exibidor do objeto de mídia podem tratar a tecla “BACK”, contudo o exibidor NCL não pode usar as notificações de acionamento da tecla “BACK” para disparar os papéis de condição de seus elementos <link> durante esse período de tempo, mas apenas controlar os direitos de uso das teclas. Quando o objeto de mídia em foco termina sua apresentação, o exibidor NCL ganha novamente o controle das teclas “CURSOR_DOWN”, “CURSOR_LEFT”, “CURSOR_RIGHT”, “CURSOR_UP”, “ENTER” (bem como dos acionamentos a partir do dispositivo ponteiro e de todas as teclas anteriormente alocadas por ele), e pode voltar a usar as notificações de acionamento da tecla “BACK” nas condições de disparo de seus elementos <link>. Durante a apresentação de um objeto de mídia em foco, seu exibidor pode passar o controle das teclas navegacionais (e do dispositivo ponteiro) para exibidores de conteúdo de seus objetos de mídia filhos (objetos internos), e assim por diante. Nesse ínterim, se a tecla “BACK” for pressionada, o controle volta para o exibidor do objeto de mídia pai (o exibidor de conteúdo do elemento filho perde o controle), até que, a cada acionamento da tecla “BACK”, o controle seja finalmente retornado ao exibidor NCL que iniciou todo o processo. A partir de então, apenas o exibidor NCL receberá as notificações de acionamento das teclas navegacionais (e das teclas anteriormente alocadas por ele e de seleções vindas de um dispositivo ponteiro). Quando o exibidor de um objeto de mídia receber o controle das teclas navegacionais e não quiser esse controle, ele pode recusar, voltando o controle como se a tecla “BACK” tivesse sido acionada.

O controle do foco também pode ser transmitido configurando a propriedade service.currentfocus, e o que tem o controle das teclas pela propriedade service.currentKeyMaster do nó *settings* (elemento <media> do tipo application/x-ginga-settings). Isso pode ser feito por meio de uma ação de um relacionamento ou por meio de um comando de edição NCL executado por um nó de código imperativo (objeto NCLua, por exemplo). O exibidor de um nó de mídia que tenha o controle das teclas não pode alterar diretamente a propriedade service.currentKeyMaster.

O atributo *focusSrc* pode especificar uma alternativa de conteúdo a ser apresentado, em lugar da apresentação corrente, quando o elemento receber o foco. De forma análoga, quando um elemento em foco for selecionado, o atributo *focusSelSrc* pode especificar uma alternativa de conteúdo a ser apresentado, em lugar da apresentação corrente. Esses dois atributos seguem as mesmas regras do atributo *src* do elemento <media>, mas não podem endereçar conteúdo com código imperativo (NCLua ou NCLet) ou declarativo (NCL ou HTML) como valor; eles devem endereçar conteúdo de mídia perceptual (vídeo, imagens, texto e áudio). O comportamento do conteúdo de mídia original não para; quando substituído, apenas continua sendo exibido escondido (visible=”false”).

5.23.2 Valores *default*

Quando a propriedade *focusIndex* não for definida, considera-se como se nenhum foco pudesse ser estabelecido no objeto correspondente.

Quando os atributos *focusBorderColor, focusBorderWidth, focusBorderTransparency,* ou *selBorderColor* não estão definidos, considera-se os valores *default*. Esses valores são especificados nas propriedades do elemento <media> do tipo application/x-ginga-settings (ou application/ x-NCL-settings): *default.focusBorderColor, default.focusBorderWidth, default.focusTransparency*, *default.selBorderColor*.

5.23.3 Tratamento de exceções

Em um determinado momento da apresentação, se o foco ainda não tiver sido definido, ou estiver perdido, o foco é inicialmente aplicado ao elemento que estiver sendo apresentado que tenha o menor valor para index.

Os valores do atributo *focusIndex* devem obrigatoriamente ser exclusivos em um documento NCL. Caso contrário,

recomenda-se que os atributos repetidos sejam ignorados, se, em determinado momento, houver mais de um elemento de <media> para ganhar o foco.

Quando um elemento de <media> faz referência a outro elemento de <media> (usando o atributo *refer*), o *focusIndex*, associado ao elemento referenciado de <media>, deve obrigatoriamente ser ignorado.

Quando o foco é aplicado a um elemento com a propriedade visível configurada como falsa, ou a um elemento que não está sendo apresentado, o foco atual não se move.

Quando o foco é aplicado a um elemento com a propriedade visível configurada como falsa, qualquer seleção no elemento deve obrigatoriamente ser ignorada.

Se o atributo *focusSrc* ou *focusSelSrc* receber um valor inválido, é recomendado que ele seja ignorado; é recomendado também que o exibidor NCL proceda como se esses atributos fossem inexistentes.

5.24 Módulo Animation

5.24.1 Considerações gerais

Uma vez que uma animação NCL pode exigir muitos recursos computacionais, ela é suportada somente no perfil EDTV, e apenas as propriedades que definem valores numéricos e as cores podem ser animadas. Um formatador NCL seguindo o perfil BDTV pode ignorar os atributos de animação.

O atributo *by* representa o passo a ser usado no incremento ou decremento do valor de uma propriedade (elemento <property>) até que seja atingido o valor final da atribuição. Ele tem uma string como valor, que deve ser um número positivo ou a string “indefinite”. Quando o valor for “indefinite”, deve ser usado o menor passo que a implementação do middleware seja capaz. Quando o valor da propriedade é uma tupla, o mesmo passo especificado no atributo *by* deve ser usado em cada elemento da tupla.

5.24.2 Valores *default*

Ao definir um novo valor para uma propriedade, a mudança é imediata, por *default* (*duration* = ″0 ″).

Ao definir um novo valor para uma propriedade, a alteração do valor antigo para o novo, se não especificado o contrário, é linear por *default* (*by* = ″indefinite″).

5.24.3 Tratamento de exceções

Se o valor atribuído a uma propriedade por um evento de atribuição for diferente do valor corrente da propriedade e o atributo by for definido como “0”, é recomendado que esse atributo seja assumido como “indefinite”.

5.25 Módulo Transition

5.25.1 Considerações gerais

As transições são classificadas de acordo com uma taxonomia de dois níveis: tipos e subtipos. O atributo *type* é exigido e utilizado para especificar a transição geral. O atributo *subtype* fornece o controle, específico por transição.

Apenas o subtipo *default* para os cinco tipos de transições necessárias, listados na Tabela 41, deve obrigatoriamente ser suportado; os outros, definidos em *SMIL* 2.1, são opcionais.

Tabela 41 — Tipos e subtipos de transição necessários

Tipo de transição	Subtipo de transição *default*
barWipe	leftToRight
irisWipe	rectangle
clockWipe	clockwiseTwelve
snakeWipe	topLeftHorizontal
fade	crossfade

O módulo de transição também define os atributos a serem usados nos elementos <descriptor> para utilizar os templates de transição definidos pelos elementos <transition>: atributos *transIn* e *transOut*.

Todas as transições definidas no módulo de transição aceitam quatro atributos adicionais provenientes de *SMIL 2.1*, que podem ser utilizados para controlar o aspecto visual das transições: *horRepeat; vertRepeat; borderWidth;* e *borderColor*.

5.25.2 Valores *default*

O atributo *subtype* é opcional. Se esse atributo não for especificado, a transição assume o subtipo *default* para o tipo especificado de transição, conforme apresentado na Tabela 41.

O valor *default* para o atributo *dur* é de 1 s.

O atributo *startProgress* especifica a progressividade na qual começa a execução da transição. O valor *default* é 0.0.

O atributo *endProgress* especifica a progressividade na qual termina a execução da transição. O valor *default* é 1.0.

O atributo *direction* especifica a direção em que a transição é executada. O valor *default* é “forward”.

O valor *default* para o atributo *fadeColor* é “black”.

O valor *default* para os atributos *transIn* e *transOut* é uma sequência de caracteres (string) vazia, que indica que nenhuma transição é realizada.

O atributo *horRepeat* especifica a quantidade de vezes que o padrão de transição é executado na linha do eixo horizontal. O valor *default* é 1.

O atributo *vertRepeat* especifica a quantidade de vezes que o padrão de transição é executado na linha do eixo vertical. O valor *default* é 1.

O atributo *borderWidth* especifica a largura de uma borda em uma barra de varredura. O valor *default* é 0.

O atributo *borderColor* especifica o conteúdo de uma borda em uma barra de varredura. O valor *default* para esse atributo é a cor "black".

5.25.3 Tratamento de exceções

Se o tipo de transição especificada não for suportado pelo formatador NCL, recomenda-se que a transição seja ignorada.

O atributo *subtype*, se especificado, deve obrigatoriamente ser um dos subtipos apropriado para o tipo especificado; caso contrário, recomenda-se que a transição seja ignorada.

O valor do atributo *endProgress* é um número real no intervalo [0.0, 1.0] e deve obrigatoriamente ser igual ou maior que o valor do atributo *startProgress*. Se o valor *endProgress* for especificado como inferior ao valor de *startProgress*, recomenda-se que o efeito de transição seja ignorado. Se *endProgress* for igual a *startProgress*, então a transição permanece em um progressividade fixa.

Observar que nem todas as transições terão interpretações inversas significativas. Por exemplo, um crossfade não é uma transição geométrica e, portanto, não tem interpretação de sentido inverso. As transições que não apresentam uma interpretação inversa devem obrigatoriamente ter o atributo *direction* ignorado e o valor *default* "forward" adotado.

Se o valor do atributo *type* não for "fade", ou o valor do atributo *subtype* não for "fadeToColor" ou "fadeFromColor", então o atributo *fadeColor* deve obrigatoriamente ser ignorado.

Se o valor do atributo *transIn* ou do atributo *transOut* não corresponder ao valor do identificador XML de qualquer um dos elementos de transição definidos, então recomenda-se que o valor do atributo seja considerado como sendo a sequência (string) vazia e, portanto, nenhuma transição deve ser realizada.

5.26 Módulo Metainformation

Não há comentários que complementem ou esclareçam a ABNT NBR 15606-2.

6 Objetos de mídia em apresentações NCL

6.1 Considerações gerais

O descrito em 6.2 a 6.6 está relacionado e complementa a ABNT NBR 15606-2:2007, Seção 8, e ABNT NBR 15606-5:2008, Seção 8.

Como a arquitetura e a implementação do Ginga-NCL é uma opção de cada desenvolvedor do receptor, o objetivo do que está descrito em 6.2 a 6.6 é apenas definir o funcionamento esperado de um exibidor de mídia ao controlar os objetos que fazem parte de um documento (aplicação) NCL.

Um objeto de mídia em execução é identificado pelo atributo *id* do elemento <media> correspondente e pelos atributos *id* dos decritores que foram associados ao objeto. Essa identificação do objeto de mídia em exibição é denominada *representationObjectId* nesta Seção 6.

Um exibidor de mídia (ou seu adaptador) deve obrigatoriamente ser capaz de receber comandos de apresentação, controlar as máquinas de estado dos eventos do objeto de mídia e responder a requisições do formatador. As disposição dadas em 6.2 a 6.6 descrevem como um exibidor de mídia reage a cada instrução emitida pelo formatador e qual o comportamento de um objeto de composição (representado pelos elementos <context>, <body> e <switch>).

6.2 Funcionamento esperado dos exibidores básicos de mídia

6.2.1 Considerações gerais

O descrito em 6.2.2 a 6.2.11 trata dos exibidores de mídia para os elementos <media> cujos tipos são diferentes de "application/x-ginga-NCL" (ou "application/x-ncl-NCL"), "application/x-ginga-NCLua" (ou "application/x-ncl-NCLua"), "application/x-ginga-NCLet" (ou "application/x-ncl-NCLet") e text/html.

O descrito em 6.2.2 a 6.2.11 é inteiramente baseado na ABNT NBR 15606-2:2007, 8.2.

6.2.2 Instrução *start* para eventos de apresentação

Antes de enviar uma instrução *start*, é recomendado que o formatador encontre o exibidor de mídia mais adequado para ser acionado, com base no tipo de conteúdo a ser exibido. Para tanto, o formatador leva em consideração o atributo *player* associado ao objeto de mídia a ser exibido. Se esse atributo não for especificado, o formatador deve obrigatoriamente considerar o atributo *type* do elemento <media>. Se esse atributo também não for especificado, o formatador deve obrigatoriamente considerar a extensão de arquivo especificada no atributo *src* do elemento <media>.

A instrução *start* emitida por um formatador deve obrigatoriamente informar os seguintes parâmetros para o exibidor de mídia: a localização do conteúdo do objeto de mídia a ser controlado, a lista de todas as propriedades associadas ao objeto de mídia, a identificação *representationObjectId* do objeto em exibição, a lista de eventos (apresentação, seleção ou atribuição) que precisam ser controlados pelo exibidor de mídia, o evento de apresentação cuja exibição precisa ser iniciada (chamado aqui de evento principal), o deslocamento no tempo *offset-time* (quando especificado) e o tempo de retardo *delay-time* (quando especificado).

NOTA 1 Os parâmetros *offset-time* e *delay-time* são computados pelo formatador NCL, antes de desencadear o início da instrução, com base no controle do ciclo de vida da aplicação, como, por exemplo, baseando-se em um NCLEdittingCommand recebido.

NOTA 2 É recomendado que as propriedades de apresentação que não são obrigatórias na ABNT NBR 15606-2 sejam ignoradas pelo exibidor de mídia.

O atributo *src* do elemento <media> é utilizado pelo exibidor de mídia para localizar o conteúdo a ser exibido e iniciar sua apresentação. Se o conteúdo não puder ser localizado, ou se o exibidor de mídia não souber como lidar com o tipo de conteúdo, o exibidor de mídia conclui a operação de início sem efetuar qualquer ação.

Os descritores devem obrigatoriamente ser selecionados pelo formatador seguindo as diretrizes definidas no documento NCL. Se a instrução *start* resultar de uma ação de um relacionamento (elo) que tem um descritor explicitamente declarado em seu elemento <bind> (atributo *descriptor* do elemento <bind> filho do elemento <link>), o descritor resultante deve unificar os atributos do descritor especificado no elo com os atributos do descritor especificado no elemento <media> correspondente, se esse atributo tiver sido especificado. Para os atributos comuns aos dois descritores, a informação do descritor especificado no elemento <bind> deve obrigatoriamente sobrepor a informação especificada no descritor definido pelo elemento <media>. Se o elemento <bind> não contiver um descritor explícito, o descritor calculado pelo formatador é aquele especificado pelo elemento <media>, se esse atributo tiver sido especificado. Caso contrário, um descritor *default* para o tipo do elemento <media> é escolhido pelo formatador (um descritor *default* que deve estar em conformidade com os vallores *default* recomendados para cada atributo ou parâmetro de descritores). Com base nesse procedimento, a lista dos atributos *id* dos decritores que são associados ao objeto é construída. Unificando as propriedades do descritor resultante com as propriedades explicitamente definidas no elemento <media>, a lista de propriedades associadas ao objeto de mídia é definida.

É recomendado que a lista de eventos a ser monitorada por um exibidor de mídia também seja computada pelo formatador, levando-se em consideração a especificação do documento NCL. Na computação são avaliados todos os relacionamentos em que o objeto de mídia e o descritor resultante participam. Ao computar os eventos que serão monitorados, o formatador deve considerar obrigatoriamente a perspectiva do objeto de mídia, ou seja, o caminho de elementos <body> e <context> aninhados até atingir o elemento <media>. Apenas os relacionamentos contidos no elemento <body> e nos elementos <context> do caminho devem ser obrigatoriamente considerados no cálculo dos eventos monitorados.

O parâmetro *offset-time* é opcional, seu valor *default* é “zero”, e ele é relevante somente para mídias contínuas ou mídias estáticas com duração explícita. Nesse caso, esse parâmetro define um tempo de deslocamento, do início (tempo de início) do evento principal, a partir do qual a apresentação do evento principal será imediatamente iniciada (ou seja, ele comanda o exibidor para avançar para o *tempo de entrada*, igual ao *tempo de início* mais o *tempo de deslocamento*). Obviamente, o valor do tempo de deslocamento deve obrigatoriamente ser inferior à duração do evento principal; caso contrário, a instrução *start* deve obrigatoriamente ser ignorada. Se o tempo de deslocamento for maior do que zero, o exibidor de mídia deve obrigatoriamente colocar o evento principal no estado *occurring*, mas a transição do evento *starts* não é notificada. Se o tempo de deslocamento for zero, o

exibidor de mídia deve obrigatoriamente colocar o evento principal no estado *occurring* e notificar a ocorrência da transição *starts*.

Os eventos que teriam seu tempo final antes do tempo de início do evento principal e os eventos que teriam seu tempo de início após o final do evento principal não precisam ser monitorados pelo exibidor de mídia (é recomendado que o formatador faça essa verificação ao elaborar a lista de eventos monitorados).

Os eventos monitorados que teriam seu tempo de início antes do tempo de entrada do evento principal e o tempo final após o tempo de entrada do evento principal serão colocados no estado *occurring*, mas suas transições *starts* não serão notificadas (os relacionamentos que dependem dessa transição obrigatoriamente não podem ser acionados).

Os eventos monitorados que teriam seu tempo final após o tempo de início do evento principal, mas antes do tempo de entrada (tempo de início + tempo de deslocamento), devem obrigatoriamente ter seu atributo *occurrences* incrementado, mas as transições *starts* e *stops* não serão notificadas.

O tempo de retardo também é um parâmetro opcional e seu valor *default* é “zero”. Se for maior que zero, esse parâmetro contém um tempo a ser aguardado pelo exibidor de mídia antes de começar a apresentação.

Se um exibidor de mídia receber uma instrução *start* para um objeto que já está sendo apresentado (pausado ou não), ele obrigatoriamente deve ignorar a instrução e continuar a controlar a apresentação em andamento. Nesse caso, o elemento <simpleAction> que causou a instrução *start* não causa qualquer transição na máquina de estados do evento correspondente.

6.2.3 Instrução stop para eventos de apresentação

A instrução *stop* precisa apenas identificar o objeto de mídia que já esteja sendo controlado (*representationObjectId*). Identificar o objeto de mídia significa identificar o elemento <media> e os descritores correspondentes. Portanto, se um elemento <simpleAction> com atributo *actionType* igual a “stop” for vinculado por meio de um relacionamento a uma interface do nó, a interface deve obrigatoriamente ser ignorada quando a ação for executada.

Se o objeto não estiver sendo apresentado (nenhum dos eventos na lista de eventos do objeto estiver no estado *occurring* ou *paused*) e o exibidor de mídia não estiver aguardando, devido a uma instrução *start* retardada, a instrução *stop* é ignorada. Se o objeto estiver sendo apresentado, seu evento principal (o evento transmitido como parâmetro quando o objeto de mídia foi iniciado) e todos os eventos no estado *occurring* ou *paused* com o tempo final igual ou anterior ao tempo final do evento principal passam para o estado *sleeping* e suas transições *stop* são notificadas. Os eventos monitorados que estejam no estado *occurring* ou *paused* com tempo final posterior ao tempo final do evento principal são colocados no estado *sleeping*, mas suas transições *stop* não são notificadas e seus atributos *occurrences* não são incrementados. A apresentação do conteúdo do objeto deve obrigatoriamente ser interrompida. Se o valor do atributo *repetitions* do evento for maior do que zero, ele é decrementado e a apresentação do evento principal é reiniciada após o tempo entre repetições (o tempo entre repetições é passado para o exibidor de mídia como o parâmetro de retardo inicial). Se o objeto de mídia estiver aguardando para ser apresentado após uma instrução *start* retardada e uma instrução *stop* for emitida, a instrução *start* anterior é eliminada.

A instrução *stop* deve obrigatoriamente mudar os eventos monitorados do objeto para o estado *sleeping*, não importando se um efeito de transição está sendo aplicado ao objeto de mídia. Em outras palavras, o efeito de transição é interrompido. Os efeitos de transição não podem ser aplicados depois que um objeto sofre uma instrução *stop*.

Quando todos os objetos de mídia que referem a fluxos elementares de vídeo direcionados ao plano de vídeo estão no estado sleeping, o vídeo principal deve obrigatoriamente ser dimensionado para 100 % da tela. Um fluxo de vídeo elementar pode ser redimensionado somente usando um objeto de mídia (a ele referindo) em apresentação. O mesmo acontece com o áudio principal. Quando todos os objetos de mídia que referem a um fluxo elementar de áudio estão no estado *sleeping*, o áudio principal deve obrigatoriamente ser dimensionado para 100 % de seu volume.

6.2.4 Instrução *abort* para eventos de apresentação

A instrução *abort* precisa apenas identificar o objeto de mídia que já esteja sendo controlado (*representationObejctId*). Portanto, se um elemento <simpleAction> com atributo *actionType* igual a “abort” for vinculado por meio de um relacionamento a uma interface de nó, a interface deve obrigatoriamente ser ignorada quando a ação for executada.

Se o objeto não estiver sendo apresentado e não estiver aguardando para ser apresentado após uma instrução *start* retardada, a instrução *abort* é ignorada. Se o objeto estiver sendo apresentado, o evento principal e todos os eventos no estado *occurring* ou *paused* passarão para o estado *sleeping* e suas transições *aborts* serão notificadas. Qualquer apresentação do conteúdo deve obrigatoriamente parar. Se o atributo *repetitions* do evento for maior que zero, ele deve obrigatoriamente ser configurado para zero e a apresentação do objeto de mídia não é reiniciada. Se o objeto de mídia estiver aguardando para ser apresentado após uma instrução *start* retardada e uma instrução *abort* emitida, a instrução anterior *start* deve obrigatoriamente ser eliminada.

6.2.5 Instrução *pause* para eventos de apresentação

A instrução *pause* precisa apenas identificar o objeto de mídia que já esteja sendo controlado (*representationObejctId*). Portanto, se um elemento <simpleAction> com atributo *actionType* igual a “pause” for vinculado por meio de um relacionamento a uma interface de nó, a interface deve obrigatoriamente ser ignorada quando a ação for executada.

Se o objeto não estiver sendo apresentado (isto é, se o evento principal, passado como parâmetro quando o objeto de mídia foi iniciado, não estiver no estado *occurring*) e não estiver aguardando para ser apresentado após uma instrução *start* retardada, a instrução é ignorada. Se o objeto estiver sendo apresentado, o evento principal e todos os eventos monitorados no estado *occurring* mudarão para o estado *paused* e suas transições *pauses* serão notificadas. A apresentação do objeto deve obrigatoriamente ser pausada e o tempo decorrido da pausa não pode ser considerado como parte da duração do objeto. Como exemplo, se um objeto tiver uma duração explícita de 30 s e, após 25 s for pausado, mesmo que o objeto fique em pausa, por exemplo, por 7 min, após retomar, o evento principal do objeto permanecerá ocorrendo por mais 5 s. Se o evento principal não estiver ocorrendo devido ao exibidor de mídia estar aguardando o tempo de exibição devido a uma instrução *start* retardada, o objeto de mídia deve obrigatoriamente esperar por uma instrução *resume* para continuar aguardando o atraso restante para a partida.

6.2.6 Instrução *resume* para eventos de apresentação

A instrução *resume* precisa apenas identificar o objeto de mídia que já esteja sendo controlado (*representationObejctId*). Portanto, se um elemento <simpleAction> com atributo *actionType* igual a “resume” for vinculado por meio de um relacionamento a uma interface de nó, a interface deve obrigatoriamente ser ignorada quando a ação for executada.

Se o objeto não estiver pausado (isto é, se o evento principal, transmitido como parâmetro quando o objeto de mídia foi iniciado, não estiver no estado *paused*) e não estiver aguardando para ser apresentado após uma instrução *start* retardada, a instrução é ignorada. Se o exibidor de mídia estiver aguardando o tempo de exibição devido a uma instrução *start* retardada, ele deve obrigatoriamente retomar a espera a partir do instante em que foi pausado. Se o evento principal estiver no estado *paused*, o evento principal e todos os eventos monitorados no estado *paused* serão colocados no estado *occurring* e suas transições *resumes* serão notificadas.

6.2.7 Instrução *start* para eventos de atribuição

A instrução *start* pode ser aplicada a uma propriedade do objeto independentemente do fato de o objeto estar sendo apresentado ou não (nesse último caso, embora o objeto não esteja sendo apresentado, o seu exibidor de mídia já deve obrigatoriamente estar instanciado). A instrução *start* precisa identificar o objeto de mídia que está sendo controlado (*representationObjectId*) e um evento de atribuição monitorado, e determinar um valor a ser designado à propriedade associada ao evento, a duração da atribuição e o passo da atribuição. Ao atribuir um valor para a propriedade, o exibidor de mídia coloca a máquina de estado do evento correspondente no estado *occurring* e, após terminar a atribuição, novamente para o estado *sleeping*, gerando a transição *start* e depois a transição *stop*.

Para cada evento de atribuição monitorado, se o exibidor de mídia mudar por si só o valor da propriedade correspondente, ele deve obrigatoriamente proceder como se tivesse recebido uma instrução *start* externa.

6.2.8 Instrução *stop, abort, pause* e *resume* para eventos de atribuição

As instruções *stop, abort, pause* e *resume* precisam identificar o objeto de mídia que está sendo controlado (*representationObjectId*) e o evento de atribuição monitorado.

A instrução *stop* para o procedimento de atribuição, trazendo a máquina de estado do evento correspondente para o estado *sleeping*. Ocorrendo a transição da máquina de estado, a transição *stop* deve ser gerada.

A instrução *abort* para o procedimento de atribuição, trazendo a máquina de estado do evento correspondente para o estado *sleeping* e o valor da propriedade para seu valor inicial. Ocorrendo a transição da máquina de estado, a transição *abort* deve ser gerada.

A instrução *pause* pausa o procedimento de atribuição, trazendo a máquina de estado do evento correspondente para o estado *paused*. Ocorrendo a transição da máquina de estado, a transição *pause* deve ser gerada.

Finalmente, a instrução *resume* retoma o procedimento de atribuição, trazendo a máquina de estado do evento correspondente para o estado *occurring*. Ocorrendo a transição da máquina de estado, a transição *resumes* deve ser gerada.

6.2.9 Instrução addEvent

A instrução *addEvent* é emitida em caso de recebimento de um comando de edição NCL *addInterface* (ver Seção 7). A instrução precisa identificar um objeto de mídia que já esteja sendo controlado e um novo evento a ser incluído no conjunto de eventos sendo monitorados. Todas as regras aplicadas à interseção de eventos monitorados com o evento principal serão aplicadas ao novo evento. Se o tempo de início do novo evento for anterior ao tempo atual de exibição do objeto e o tempo final do novo evento for posterior ao tempo atual de exibição do objeto, o novo evento deve obrigatoriamente ser colocado no mesmo estado do evento principal (*occurring* ou *paused*), sem notificar a transição correspondente.

6.2.10 Instrução de removeEvent

A instrução *removeEvent* é emitida em caso de recebimento de um comando de edição NCL *removeInterface*. A instrução precisa identificar um objeto de mídia que já esteja sendo controlado e um evento monitorado que não pode ser mais controlado. O estado do evento deve obrigatoriamente ser colocado no estado *sleeping* sem gerar qualquer transição.

6.2.11 Final natural de uma apresentação

Os eventos de um objeto com duração explícita ou duração intrínseca normalmente finalizam suas apresentações naturalmente, sem precisar de instruções externas. Nesse caso, o exibidor de mídia deve obrigatoriamente mudar o evento para o estado *sleeping* e notificar a transição *stops*. O mesmo deve ser obrigatoriamente feito para os eventos monitorados que estejam no estado *occurring* e que tenham o mesmo tempo final do evento principal ou que tenham um tempo de término desconhecido, quando o evento principal termina. Eventos no estado *occurring* com tempo final posterior ao tempo final do evento principal devem obrigatoriamente ser colocados no estado sleeping, mas sem gerar a transição *stops* e sem incrementar o atributo *occurrences*. É importante ressaltar que, quando o evento principal termina, toda a apresentação do objeto de mídia deve obrigatoriamente terminar.

NOTA O autor do aplicativo deve considerar que o final natural dos conteúdos recebidos como fluxos elementares pode ser notificado apenas algum tempo depois, devido ao atraso dos descritores que transmitem essa informação.

6.3 Funcionamento esperado dos exibidores hipermídia em aplicações NCL

Os objetos de mídia NCL (elementos <media> do tipo “application/x-ginga-NCL” ou “application/x-ncl-NCL”)

e objetos baseados em HTML (elementos <media> do tipo“text/html”), incorporados em aplicações NCL, seguem as diretrizes estabelecidas em Nested Context Language 3.0, Part 11.

6.4 Funcionamento esperado dos exibidores imperativos em aplicações NCL

6.4.1 Considerações gerais

Objetos imperativos podem ser inseridos em documentos NCL, trazendo capacidades computacionais adicionais a aplicações declarativas. A forma de inserir objetos imperativos em documentos NCL é definir um elemento <media> cujo conteúdo (localizado pelo atributo *src*) é o código imperativo a ser executado. Os perfis EDTV e BDTV da NCL 3.0 permitem que dois tipos de mídia sejam associados com o elemento <media>: application/x-ginga-NCLua (ou “application/x-ncl-NCLua”), para códigos imperativos Lua (extensão de arquivo .lua); e application/x-ginga-NCLet (ou “application/x-ncl-NCLet”), para códigos imperativos Java (Xlet) (extensão de arquivo .class ou .jar), sendo esse último opcional no Ginga para receptores portáteis.

Os exibidores imperativos para objetos NCL dos tipos “application/x-ginga-NCLua” (ou “application/x-ncl-NCLua”) e “application/x-ginga-NCLet” (ou “application/x-ncl-NCLet”) seguem as diretrizes estabelecidas em Nested Context Language 3.0, Part 10, Seção 5. Essa seção é uma tradução autorizada dessa referência, devido à sua importância na definição da ponte entre o ambiente declarativo e um ambiente imperativo.

Em um objeto imperativo, trechos de código imperativo podem ser associados com um elemento <área> filho por meio do atributo *label*. Um elemento <área> pode também ser usado apenas como interface para ser usada como codição de elos para disparo de ações em outros objetos NCL.

Um objeto imperativo tem uma âncora de conteúdo total (*whole content anchor*) definida por *default*. Essa âncora de conteúdo tem, no entanto, um significado especial. Ela representa a execução de qualquer trecho de código do objeto imperativo. Uma outra âncora é também definida por *default*, a âncora de conteúdo principal (*main content anchor*). Todas as vezes que um objeto imperativo for iniciado sem a especificação de uma de suas âncoras de conteúdo, a âncora de conteúdo principal deve obrigatoriamente ser assumida. Em qualquer outra referência ao objeto imperativo sem especificar uma de suas âncoras ou propriedades, que não a referência para partida do objejto imperativo, a âncora de conteúdo total deve obrigatoriamente ser assumida.

Autores de um documento NCL podem definir elos para iniciar, parar, pausar, retomar ou abortar a execução de um código imperativo. Um exibidor imperativo (máquina de execução da linguagem) deve obrigatoriamente fazer a interface entre o ambiente de execução imperativo e o formatador NCL.

Similarmente aos exibidores de conteúdo de mídia perceptual (vídeo, áudio, imagem etc.), os exibidores de código imperativo devem obrigatoriamente controlar as máquinas de estado de eventos associados ao objeto imperativo. Como exemplo, se o código terminar sua execução, o exibidor deve obrigatoriamente gerar a transição *stop* na máquina de estado do evento de apresentação correspondente à execução do código. No entanto, diferentemente dos exibidores de conteúdo de mídia, um exibidor de código imperativo não tem informações suficientes para controlar, por si só, todas as máquinas de estado de evento e deve então recorrer ao conteúdo da aplicação imperativa para comandar esses controles.

Elos NCL podem ser ligados às interfaces de um objeto imperativo (elementos <area> e <property> e as âncoras de conteúdo definidas por *default*).

Se um relacionamento iniciar, parar, pausar, retomar ou abortar a apresentação de uma âncora que representa um elemento <area> ou a âncora de conteúdo principal (*main content anchor*), *callbacks* no código imperativo devem obrigatoriamente ser acionadas. A forma como as *callbacks* são definidas é de responsabilidade de cada código imperativo associado ao objeto imperativo.

Por outro lado, um código imperativo também pode comandar ações para iniciar, parar, pausar ou retomar eventos de apresentação associados às suas âncoras de conteúdo por meio de uma API oferecida pela linguagem. As transições decorrentes podem ser usadas como condições nos elos NCL para desencadear ações em outros objetos do mesmo documento NCL. Desse modo, pode-se estabelecer uma sincronização de duas vias entre o código imperativo e o restante do documento NCL.

Um código imperativo também pode ser sincronizado com outros objetos por meio dos elementos <property>. Quando o elemento <property> é mapeado para um trecho de código (função, método etc.) por meio do seu atributo *name*, uma ação "start" aplicada à propriedade causa a execução do código, com os valores da atribuição sendo interpretados como parâmetros passados ao trecho do código. Quando o elemento <property> é mapeado para um atributo do código imperativo, a ação "start" deve obrigatoriamente determinar o valor ao atributo. Como de costume, a máquina de estado do evento associado à propriedade deve obrigatoriamente ser controlada pelo exibidor do objeto imperativo, mas, às vezes, comandado pela aplicação imperativa.

Um elemento <property> definido como filho de um elemento <media> representando um objeto imperativo também pode ser associado a um papel de *assessment* de um elo NCL. Nesse caso, o formatador NCL deve consultar o valor da propriedade para avaliar a expressão do elo. Se o elemento <property> for mapeado para um atributo do código, o valor do atributo é devolvido pelo exibidor do objeto imperativo para o formatador NCL. Se o elemento <property> for mapeado para um trecho de código, o código deve obrigatoriamente ser executado e seu valor de saída deve ser devolvido pelo exibidor do objeto imperativo para o formatador NCL.

6.4.2 Modelo de execução de um objeto imperativo

O ciclo de vida de um objeto imperativo é controlado pelo formatador NCL. O formatador é responsável por desencadear a execução de um objeto imperativo e mediar a comunicação entre esse objeto e outros objetos em um documento NCL.

Tal como acontece com todos os exibidores de objetos de mídia, uma vez instanciado, o exibidor do objeto imperativo executa um procedimento de inicialização. Porém, diferentemente de outros exibidores de mídia, esse código de inicialização deve obrigatoriamente ser especificado pelo autor do código imperativo. O procedimento de inicialização é executado apenas uma vez, para cada instância do objeto, criando todos os trechos de código e dados que podem ser utilizados durante a execução do objeto imperativo e, em particular, registrando um (ou mais) tratadores de eventos para a comunicação com o formatador NCL. Observar que pelo menos o trecho de código associado com a âncora de conteúdo principal (*main content anchor*) é criado durante o processo de inicialização.

Após a inicialização, a execução do objeto imperativo passa a ser orientada a evento, em ambas as direções. Ou seja, qualquer ação comandada pelo formatador NCL atinge os tratadores de eventos registrados e qualquer notificação de alteração de estado de evento NCL é enviada como um evento ao formatador NCL (como, por exemplo, o fim natural da execução de um trecho de código). O exibidor do objeto imperativo estará então preparado para executar qualquer instrução, conforme discutido nas próximas seções.

6.4.3 Instruções para eventos de apresentação

6.4.3.1 Considerações gerais

Os formatadores NCL podem controlar os exibidores de objetos imperativos emitindo instruções que podem causar alterações nas máquinas de estado dos eventos de apresentação (execuções de trechos de código). Por outro lado, quaisquer mudanças de estado nessas máquinas de estado de evento de apresentação são notificadas ao formatador NCL.

6.4.3.2 Instrução start

A instrução *start* emitida por um formatador deve obrigatoriamente informar os seguintes parâmetros para o exibidor de objeto imperativo: a localização do conteúdo do objeto imperativo a ser controlado; a lista de todas as propriedades associadas ao objeto de mídia, a identificação *representationObjectId* do objeto em exibição, uma lista de eventos (definidos pelos elementos <area> e <property> filhos do elemento <media> e pelas âncoras de conteúdo definidas por *default*) que precisam ser monitorados pelo exibidor do objeto imperativo; a âncora de conteúdo identificada por *label*, ou, por *default*, a âncora de conteúdo principal, identificando o código imperativo associado a ser iniciado; e um tempo de retardo, opcional. A partir do atributo *src* do objeto de mídia, o exibidor de objeto imperativo tenta localizar o código imperativo e iniciar sua execução. Se o conteúdo não puder ser localizado, é recomendado que o exibidor termine a operação de início sem executar qualquer ação.

Os descritores são selecionados pelo formatador seguindo as diretrizes definidas no documento NCL. Se a instrução *start* resultar de uma ação de um elo que tem um descritor explicitamente declarado em seu elemento <bind> (atributo *descriptor* do elemento <bind> filho do elemento <link>), o descritor resultante, calculado pelo formatador, deve obrigatoriamente unificar os atributos do descritor do elo com os atributos do descritor especificado no elemento <media> correspondente, se esse atributo tiver sido especificado. Para os atributos comuns, a informação do descritor especificado no elemento <bind> deve obrigatoriamente sobrepor a informação do descritor especificado no elemento <media> correspondente. Se o elemento <bind> não contiver um descritor explícito, o descritor calculado pelo formatador é aquele especificado no elemento <media>, se esse descritor tiver sido especificado. Caso contrário, um descritor *default* para o tipo específico desse objeto imperativo é escolhido pelo formatador. Com base nesse procedimento, a lista dos atributos *id* dos decritores que são associados ao objeto é construída e, unificando as propriedades do descritor resultande com as propriedades definidas no elemento <media>, a lista de propriedades associadas ao objeto de mídia é definida.

É recomendado que a lista de eventos a ser monitorada por um exibidor de objeto imperativo seja computada pelo formatador, levando-se em consideração a especificação do documento NCL. O formatador deve verificar todos os relacionamentos em que o objeto imperativo, com o descritor resultante, participa. Ao computar os eventos a serem monitorados, o formatador deve considerar a perspectiva do objeto de mídia, ou seja, o caminho de elemento <body> e elementos <context> até atingir o elemento <media>. É recomendado que apenas os relacionamentos contidos no elemento <body> e nos elementos <context> sejam considerados no cálculo dos eventos monitorados.

Assim como em qualquer outro elemento de <media>, o tempo de retardo é um parâmetro opcional e seu valor *default* é “zero”. Se for maior que zero, esse parâmetro contém um tempo a ser aguardado pelo exibidor de objeto imperativo antes de começar a execução do código.

Diferentemente do que é realizado em outros elementos <media>, se um exibidor de objeto imperativo receber uma instrução *start* para um evento associado com a âncora de conteúdo e esse evento estiver no estado *sleeping*, é iniciada a execução do código imperativo associado ao elemento, mesmo que outra parte do código imperativo do objeto esteja sendo executada (pausada ou não). No entanto, se o evento associado com a âncora de conteúdo/alvo estiver no estado *occurring* ou *paused*, a instrução *start* deve obrigatoriamente ser ignorada pelo exibidor de código imperativo, que continuará a controlar a execução em andamento. Como consequência, diferentemente do que acontece em outros elementos <media>, um elemento <simpleAction> com um atributo *actionType* igual a “stop”, “pause”, “resume” ou ”abort” deve obrigatoriamente ser vinculado por meio de um elo a uma interface do objeto imperativo, que não pode ser ignorada quando a ação for executada.

Uma vez que nem o formatador nem o exibidor de código imperativo possuem qualquer outro conhecimento sobre as âncoras de conteúdo do objeto imperativo, exceto seus atributos *id* e *label*, eles não sabem quais outras âncoras de conteúdo devem ter seus eventos associados colocados no estado *occurring* quando uma âncora de conteúdo for iniciada ou estiver em execução. Portanto, exceto para o evento associado a *whole content anchor*, é de responsabilidade do trecho de código imperativo, assim que for iniciado, comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado de evento que esteja relacionada à máquina de estado de evento associada ao código iniciado e informar se uma transição associada com uma alteração deve ser notificada. De forma similar, é de responsabilidade do trecho de código imperativo comandar qualquer alteração de estado de evento e informar se a transição associada deve ser notificada, caso a execução do trecho de código inicie outro trecho de código associado a uma âncora de conteúdo.

Diferentemente de outros elementos <media>, se qualquer âncora de conteúdo for iniciada e o evento associado a *whole content anchor* estiver em estado *sleeping* ou *paused*, ele deve obrigatoriamente ser colocado no estado *occurring* e a transição correspondente deve ser notificada.

6.4.3.3 Instrução stop

A instrução *stop* precisa apenas identificar um trecho de código imperativo que já esteja sendo controlado. Identificar o trecho de código imperativo significa identificar o objeto de mídia em execução correspondente (*representationObejctId*) e a interface do elemento de <media>.

É recomendado que a instrução *stop* emitida por um formatador NCL seja ignorada por um exibidor de objeto imperativo se o trecho do código imperativo associado com a interface especificada não estiver sendo executado

(se o evento correspondente não estiver no estado *occurring* ou *paused*) e se o exibidor de objeto imperativo não estiver aguardando devido a uma instrução *start* retardada. Se a interface do objeto imperativo estiver sendo executada, seu evento de apresentação correspondente deve obrigatoriamente passar para o estado *sleeping* e sua transição *stop* correspondente deve obrigatoriamente ser notificada. A execução do código imperativo associado com a interface deve obrigatoriamente ser interrompida. Se o atributo *repetition* do evento for maior do que zero, ele deve ser decrementado e o código imperativo associado à interface reiniciado após o tempo de retardo de repetição (o retardo de repetição deve ter sido passado para o exibidor de mídia como parâmetro de retardo de partida). Se o objeto de mídia estiver aguardando para ser apresentado após uma instrução *start* retardada e uma instrução *stop* for emitida, a instrução *start* anterior deve obrigatoriamente ser eliminada.

Pelo mesmo motivo discutido na instrução *start*, exceto para o evento associado a *whole content anchor*, é de responsabilidade do trecho de código interrompido, antes da interrupção total, comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado de evento que esteja relacionada à máquina de estado de evento associada ao código interrompido e informar se a transição associada a uma alteração deve ser notificada.

Diferentemente de outros elementos <media>, se qualquer âncora de conteúdo for interrompida e todos os outros eventos de apresentação estiverem no estado *sleeping,* a *whole content anchor* deve obrigatoriamente ser colocada no estado *sleeping*. Se uma âncora de conteúdo for interrompida e pelo menos outro evento de apresentação estiver no estado *occurring,* a *whole content anchor* deve obrigatoriamente permanecer no estado *occurring*. Em todos os outros casos, se uma âncora de conteúdo for interrompida, a *whole content anchor* deve obrigatoriamente ser colocada no estado *paused*. Se a instrução *stop* for aplicada a um objeto imperativo sem especificar a interface do nó, a *whole content anchor* deve ser assumida. Nesse caso, instruções *stop* devem ser emitidas para todas as outras âncoras de conteúdo.

6.4.3.4 Instrução abort

A instrução *abort* precisa identificar um trecho de código imperativo que já esteja sendo controlado. Identificar o trecho de código imperativo significa identificar o objeto de mídia em execução correspondente (*representationObejctId*) e a interface do elemento de <media>.

É recomendado que, se o código imperativo associado à interface do objeto não estiver sendo executado e não estiver aguardando para ser executado após uma instrução *start* retardada, a instrução *abort* seja ignorada. Se o código imperativo associado à interface do objeto estiver sendo executado, seu evento associado, no estado *occurring* ou *paused*, deve obrigatoriamente passar para o estado *sleeping*, e sua transição *abort* correspondente deve obrigatoriamente ser notificada. Se o atributo *repetition* do evento for maior que zero, ele deve ser colocado em zero e a execução do código imperativo não pode ser reiniciada. Se o código imperativo associado à interface do objeto estiver aguardando para ser executado após uma instrução *start* retardada e uma instrução *abort* for emitida, a instrução *start* anterior deve obrigatoriamente ser eliminada.

Pelo mesmo motivo discutido na instrução *start*, exceto para o evento associado a *whole content anchor*, é de responsabilidade do trecho de código abortado, antes de ser abortado, comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado do evento que esteja relacionada à máquina de estado de evento associada ao código abortado e informar se a transição associada a uma alteração deve ser notificada.

Diferentemente de outros elementos <media>, se qualquer âncora de conteúdo for abortada e todos os outros eventos de apresentação estiverem no estado *sleeping*, a *whole content anchor* deve obrigatoriamente ser colocada no estado *sleeping*. Se uma âncora de conteúdo for abortada e pelo menos outro evento de apresentação estiver no estado *occurring*, a *whole content anchor* deve obrigatoriamente permanecer no estado *occurring*. Em todos os outros casos, se uma âncora de conteúdo for abortada, a *whole content anchor* deve obrigatoriamente ser colocada no estado *paused*. Se a instrução *abort* for aplicada a um objeto imperativo sem especificar a interface do nó, a *whole content anchor* é assumida. Nesse caso, instruções *abort* devem ser emitidas para todas as outras âncoras de conteúdo.

6.4.3.5 Instrução pause

A instrução *pause* precisa identificar um trecho de código imperativo que já esteja sendo controlado. Identificar o

trecho de código imperativo significa identificar o objeto de mídia em execução correspondente (*representationObejctId*) e a interface do elemento de <media>.

É recomendado que se o código imperativo associado à interface do objeto não estiver sendo executado (e não estiver no estado *paused*) e não estiver aguardando para ser executado após uma instrução *start* retardada, a instrução seja ignorada. Se o código imperativo associado à interface do objeto estiver sendo executado, seu evento associado no estado *occurring* deve obrigatoriamente passar para o estado *paused*, sua transição *pause* correspondente deve obrigatoriamente ser notificada e o tempo decorrido da pausa não pode ser considerado parte da duração do objeto. Se o código imperativo associado à interface do objeto estiver aguardando para ser executado após uma instrução *start* retardada, a interface do objeto imperativo deve esperar obrigatoriamente por uma instrução de reinício para continuar aguardando o atraso restante.

Pelo mesmo motivo discutido na instrução *start*, exceto para o evento associado a *whole content anchor*, é de responsabilidade do trecho de código pausado, antes da pausa completa, comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado de evento que esteja relacionada à máquina de estado de evento associada ao código pausado e informar se a transição associada a uma alteração deve ser notificada.

Diferentemente de outros elementos <media>, se qualquer âncora de conteúdo for pausada e todos os outros eventos de apresentação estiverem no estado *sleeping* ou *paused*, a *whole content anchor* deve obrigatoriamente ser colocada no estado *paused*. Se uma âncora de conteúdo for pausada e pelo menos outro evento de apresentação estiver no estado *occurring*, a *whole content anchor* deve obrigatoriamente permanecer no estado *occurring*. Se a instrução *pause* for aplicada a um objeto imperativo sem especificar a interface do nó, a *whole content anchor* é assumida. Nesse caso, instruções *pause* devem ser emitidas para todas as outras âncoras de conteúdo que estiverem no estado occurring.

6.4.3.6 Instrução resume

A instrução *resume* precisa identificar um trecho de código imperativo que já esteja sendo controlado. Identificar o trecho de código imperativo significa identificar o objeto de mídia em execução correspondente (*representationObejctId*) e a interface do elemento de <media>.

É recomendado que, se o código imperativo associado à interface do objeto não estiver em pausa ou o exibidor de objeto imperativo não estiver aguardando para ser executado após uma instrução *start* retardada, a instrução seja ignorada. Se o exibidor do objeto imperativo estiver pausado aguardando o atraso do início, ele deve obrigatoriamente retomar a espera a partir do instante em que foi pausado. Se o código imperativo associado à interface do objeto estiver pausado, seu evento associado deve obrigatoriamente passar para o estado *occurring* e sua transição *resume* correspondente deve obrigatoriamente ser notificada.

Pelo mesmo motivo discutido na instrução *start*, exceto para o evento associado a *whole content anchor*, é de responsabilidade do trecho de código a ser retomado, antes de retomar a execução, comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado de evento que esteja relacionada à máquina de estado de evento associada ao código retomado e informar se a transição associada a uma alteração deve ser notificada.

Diferentemente de outros elementos <media>, se qualquer conteúdo âncora for retomado, a *whole content anchor* deve obrigatoriamente ser configurada para o estado *occurring*. Se a instrução *resume* for aplicada a um objeto imperativo sem especificar a interface do nó, a *whole content anchor* é assumida. Se a *whole content anchor* não estiver no estado *paused* por ter recebido uma instrução *pause* anterior, a instrução *resume* deve obrigatoriamente ser ignorada. Caso contrário, instruções *resume* devem ser emitidas para todas as outras âncoras de conteúdo que estejam em estado *paused*, exceto aquelas que já estavam pausadas antes da *whole content anchor* receber a instrução *pause*.

6.4.3.7 Fim natural da execução de um código

Os eventos de um objeto imperativo normalmente terminam sua execução naturalmente, sem precisar de instruções externas. Nesse caso, imediatamente antes do término, um trecho do código deve comandar o exibidor de código imperativo para alterar o estado de qualquer outra máquina de estado de evento que esteja relacionada à máquina de estado de evento associada ao código que terminou e informar se a transição associada a uma

alteração deve ser notificada. O evento de término da apresentação deve obrigatoriamente passar para o estado *sleeping* e a transição *stop* deve ser notificada. Se o atributo *repetition* do evento for maior do que zero, ele deve ser decrementado e o código imperativo associado à interface reiniciado, após o tempo de retardo de repetição (o retardo de repetição deve ter sido passado para o exibidor de mídia como parâmetro de retardo inicial).

Diferentemente de outros elementos <media>, se qualquer execução da âncora de conteúdo terminar e todos os outros eventos de apresentação estiverem no estado *sleeping*, a *whole content anchor* deve obrigatoriamente ser colocada no estado *sleeping*. Se a execução de uma âncora de conteúdo terminar e pelo menos outro evento de apresentação estiver no estado *occurring*, a *whole content anchor* deve obrigatoriamente permanecer no estado *occurring*. Em todos os outros casos, se a execução de uma âncora de conteúdo terminar, a *whole content anchor* deve obrigatoriamente ser colocada no estado *paused*.

6.4.4 Instruções para eventos de atribuição

6.4.4.1 Considerações gerais

Formatadores NCL também podem enviar instruções que causem alterações em máquinas de estado de eventos de atribuição (execuções de trechos de código). Semelhantemente aos eventos de apresentação, quaisquer alterações de estado nas máquinas de estado de evento de atribuição são notificadas para o formatador NCL.

Embora propriedades do nó imperativo possam ser associadas a trechos de código, a execução desses trechos não altera qualquer máquina de estado associada a âncoras de conteúdo do objeto imperativo.

6.4.4.2 Instrução start

A instrução *start* emitida por um formatador pode ser aplicada a uma propriedade de um objeto imperativo, independentemente do fato de o objeto estar em execução (isto é, a *whole content anchor* estar no estado *occurring*) ou não (nesse último caso, embora o objeto não esteja sendo executado, seu exibidor de objeto imperativo deve obrigatoriamente ter sido instanciado). Em ambos os casos, a instrução *start* precisa identificar o objeto imperativo (*representationObejectId*), um evento de atribuição monitorado e, se for o caso, um valor a ser passado para o código imperativo envolvido com o evento. Ao receber a ação *start*, o exibidor de objeto imperativo deve obrigatoriamente colocar a máquina de estado do evento no estado *occurring* e, após terminar a atribuição, novamente para o estado *sleeping*, gerando a transição *start* e depois a transição *stop*.

Observar que, se uma instrução *start* for aplicada a um elemento <property> que solicita a execução de um trecho de código, nenhum estado de âncora de conteúdo pode ser afetado.

Para cada evento de atribuição monitorado, se o trecho do código do objeto imperativo mudar por si só o valor do atributo correspondente, ele também deve obrigatoriamente comandar o exibidor de código imperativo, que se comportará como se tivesse recebido uma instrução *start* externa.

6.4.4.3 Instrução stop, abort, pause e resume

Com exceção da instrução *start*, discutida na seção anterior, todas as outras instruções têm o mesmo efeito na atribuição da propriedade que têm na atribuição de propriedades de qualquer outro tipo de objeto.

As instruções *stop, abort, pause* e *resume* precisam identificar o objeto de mídia que está sendo controlado (*representationObjectId*) e o evento de atribuição monitorado.

A instrução *stop* apenas interrompe o processo de atribuição da propriedade, trazendo a máquina de estado do evento de atribuição para o estado *sleeping*.

A instrução *abort* interrompe o processo de atribuição da propriedade, trazendo a máquina de estado do evento de atribuição para o estado *sleeping* e restabelecendo o valor original da propriedade.

A instrução *pause* apenas pausa o processo de atribuição da propriedade, trazendo a máquina de estado do evento de atribuição para o estado *paused*.

A instrução *resume* apenas retoma o processo de atribuição da propriedade, trazendo a máquina de estado do evento de atribuição para o estado *occurring*.

6.5 Funcionamento esperado dos exibidores de mídias após instruções aplicadas a objetos compostos

6.5.1 Considerações gerais

O descrito em 6.5.2 a 6.5.6 se aplica somente a ações aplicadas a objetos representados por elementos <context>, <body> e <switch>.

Uma <simpleCondition> ou <simpleAction> com o valor do atributo *eventType* igual a “presentation” pode ser ligada por um elo a um nó de composição (representado por um elemento <context>, <switch> ou <body>) como um todo (ou seja, sem a informação de uma interface). De forma similar, uma <attributeAssessment> com o valor do atributo *eventType* igual a “presentation” e o *attributeType* igual a “state”, “occurrences” ou “repetitions” pode ser ligada por um elo a um nó de composição (representado por um elemento <context>, <switch> ou <body>) como um todo, com o valor do atributo vindo da máquina de estado do evento de apresentação definida sobre o nó de composição. Se uma <simpleAction> com valor do atributo *eventType* igual a “presentation” for vinculada por um elo a um nó de composição (representado por um elemento <context> ou <body>) como um todo (ou seja, sem a informação de uma interface), a instrução também se refletirá nas máquinas de estado dos eventos dos nós filhos da composição, conforme explicado nas subseções seguintes.

6.5.2 Iniciando uma apresentação de contexto

Se um elemento <context> ou <body> participar de um papel ação, com o tipo da ação igual a “start”, quando a ação for acionada sem a especificação de uma interface, a instrução *start* também deve ser obrigatoriamente aplicada a todos os eventos de apresentação mapeados pelas portas dos elementos <context>/<body>. Se não existir elemento <port> definido, a instrução deve ser ignorada.

Se o autor pretender iniciar a apresentação usando uma porta específica, ele deve obrigatoriamente indicar o *id* do elemento <port> como o valor do atributo *interface* do elemento <bind> correspondente. Nesse caso, a instrução *start* deve obrigatoriamente ser aplicada apenas ao evento de apresentação mapeado pela porta do elemento <context>/<body>.

6.5.3 Interrompendo uma apresentação de contexto

Se um elemento <context> ou <body> participar de um papel-ação, com o tipo de ação igual a “stop”, quando essa ação for acionada sem a especificação de uma interface, a instrução *stop* também será aplicada a todos os eventos de apresentação dos nós filhos da composição.

Se o nó de composição contiver relacionamentos sendo avaliados (ou com sua avaliação em pausa), a avaliação deve obrigatoriamente ser suspensa e nenhuma ação deve ser acionada.

6.5.4 Abortando uma apresentação de contexto

Se um elemento <context> ou <body> participar de um papel-ação, com o tipo de ação igual a “abort”, quando essa ação for acionada sem a especificação de uma interface, a instrução *abort* também deve obrigatoriamente ser aplicada a todos os eventos de apresentação dos nós-filhos da composição.

Se o nó de composição contiver relacionamentos sendo avaliados (ou com sua avaliação em pausa), a avaliação deve obrigatoriamente ser suspensa e nenhuma ação deve ser acionada.

6.5.5 Pausando uma apresentação de contexto

Se um elemento <context> ou <body> participar de um papel-ação, com o tipo de ação igual a “pause”, quando

essa ação for acionada sem a especificação de uma interface, a instrução *pause* também deve obrigatoriamente ser aplicada a todos os eventos de apresentação dos nós-filhos da composição que estão no estado *occurring*.

Se o nó de composição contiver relacionamentos sendo avaliados, todas as avaliações devem obrigatoriamente ser suspensas até que uma ação de reiniciar, parar ou abortar seja emitida.

Se o nó de composição contiver nós-filhos com eventos de apresentação já no estado de pausa quando a ação de pausa na composição for emitida, esses nós devem ser identificados, porque se a composição receber uma instrução *resume*, esses eventos não podem ser retomados.

6.5.6 Retomando uma apresentação de contexto

Se um elemento <context> ou <body> participar de um papel-ação, com o tipo de ação igual a “resume”, quando essa ação for acionada sem a especificação de uma interface, a instrução *resume* também deve obrigatoriamente ser aplicada a todos os eventos de apresentação dos nós-filhos da composição que estão no estado *paused*, exceto aqueles que já estavam pausados antes da composição ser pausada.

Se a composição contiver relacionamentos com avaliações pausadas, elas devem obrigatoriamente ser retomadas.

6.6 Relação entre a máquina de estado do evento de apresentação de um objeto NCL e a máquina de estado do evento de apresentação de seu nó (objeto) pai

Esta subseção aplica-se a objetos-pai representados por elementos <context>, <body>, <switch> e elementos de <media> do tipo “application/x-ginga-NCL” (ou “application/x-ncl-NCL”).

Sempre que um evento de apresentação de um objeto-filho (de mídia ou de composição) vai para o estado *occurring*, o evento de apresentação do objeto de composição pai ou do objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) que contém o objeto-filho também entra no estado *occuring*.

Quando todos os objetos-filhos de um objeto de composição ou de um objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) tiverem seus eventos de apresentação no estado *sleeping*, o evento de apresentação do objeto de composição pai ou do objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) também entra no estado *sleeping*.

Objetos de composição ou objetos NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) não precisam inferir nas transições *aborts* de seus objetos-filhos. Essas transições em eventos de apresentação de objeto de composição ou objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) ocorrem apenas quando as instruções *abort* são aplicadas diretamente ao evento de apresentação do objeto de composição ou do objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”).

Quando todos os objetos-filhos de um objeto de composição ou de um objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) tiverem seus eventos de apresentação em um estado diferente de *occurring* e pelo menos um objeto-filho tiver seu evento principal no estado *paused,* o evento de apresentação do objeto de composição ou do objeto NCL do tipo “application/x-ncl-NCL” (ou “application/x-ginga-NCL”) também deve estar no estado *paused*.

Se um elemento <switch> for iniciado, mas não definir um componente *default* e nenhuma das regras referidas pelos elementos <bindRule> for avaliada como verdadeira, a apresentação do switch não pode entrar no estado *occurring*.

7 Edição ao vivo e eventos de fluxo NCL

7.1 Considerações gerais

O núcleo da máquina de apresentação do Ginga-NCL é composto pelo formatador NCL e pelo módulo gerenciador de base privada.

O formatador NCL é responsável por receber um documento NCL e controlar sua apresentação, com o objetivo de garantir que os relacionamentos especificados entre os objetos de mídia sejam respeitados. O formatador NCL lida com documentos que são coletados dentro de uma estrutura de dados conhecida como *base privada*. O Ginga associa pelo menos uma base privada com cada canal de TV. Os documentos NCL em uma base privada podem ser iniciados, pausados, retomados, interrompidos e podem se referir um ao outro.

O gerenciador de base privada é responsável por receber Comandos de Edição NCL e manter os documentos ativos NCL (documentos sendo apresentados).

O protocolo DSM-CC é adotado no SBTVD-T para transportar Comandos de Edição em fluxos elementares MPEG-2 TS, quando os comandos são provenientes do canal de transmissão terrestre. Os eventos de fluxo DSM-CC e o protocolo carrossel de objetos DSM-CC (ver ABNT NBR 15606-3) constituem a base para o tratamento de documentos na máquina de apresentação do Ginga, de acordo com o SBTVD-T.

Os Comandos de Edição são codificados como eventos de fluxo DSM-CC. A máquina de apresentação do Ginga deve obrigatoriamente ser capaz de gerenciar pelo menos o evento de fluxo do Comando de Edição, cuja sintaxe é mostrada na Tabela 42.

Tabela 42 — Descritor do evento de fluxo do comando de edição

Sintaxe	Número de bits
StreamEventDescriptor ( ) {	
descriptorTag	8
descriptorLength	8
eventId	16
reserved	31
eventNPT	33
privateDataLength	8
commandTag	8
sequenceNumber	7
finalFlag	1
privateDataPayload	8 to 1928
FCS	8
}	

Os objetos de evento de fluxo de um carrossel de objetos são usados para mapear os nomes dos eventos de fluxo em identificações dos eventos de fluxo. Os objetos de evento de fluxo são usados para informar o Ginga sobre eventos de fluxo DSM-CC que podem ser recebidos. Os nomes de eventos permitem especificar os tipos de eventos, oferecendo um nível maior de abstração para aplicações do *middleware* ao se registrarem e tratarem eventos de fluxo DSM-CC. O gerenciador de base privada, assim como os objetos de execução NCL (por exemplo, objetos NCLua), devem obrigatoriamente se registrar como ouvintes de eventos de fluxos por eles tratados, usando nomes de evento.

Os arquivos de documentos NCL e os conteúdos dos objetos de mídia NCL são organizados em estruturas de

sistemas de arquivos. Os parâmetros do comando (*command parameters*) de edição, baseados em XML, podem ser transportados diretamente na carga (*payload*) de um descritor de eventos de fluxo ou, alternativamente, organizados em estruturas de sistemas de arquivos a serem transportadas, cada uma, em um carrossel de objetos. Um gerador de carrossel DSM-CC é usado para unir os sistemas de arquivos e os objetos de eventos de fluxo em um fluxo de dados elementar.

A ABNT NBR 15606-2:2007, Tabela 56, apresenta os comandos e os parâmetros carregados no campo de carga de um descritor de evento *nclEditingCommand*. Os comandos são divididos em três grupos: o primeiro, para operação de base privada (para abrir, fechar, ativar, desativar e salvar bases privadas), o segundo, para manipulação de documentos em uma base privada (para adicionar, remover e salvar um documento em uma base privada aberta, e iniciar, pausar, retomar e interromper as apresentações do documento em uma base privada ativa), e o último, para lidar com entidades NCL em uma base privada aberta. Para cada entidade NCL, são definidos os comandos *add* e *remove*. Se uma entidade já existir, o comando *add* tem a semântica de atualização (alteração).

NOTA Os parâmetros dos Comandos de Edição, baseados em XML, seguem o esquema NCL30EdCommand.xsd XML, conforme definido na ABNT NBR 15606-2. Informalmente, isso significa que a especificação de um parâmetro de edição baseado em XML é idêntica à especificação do elemento, a que se refere, nos perfis de EDTV (ou BDTV), exceto no caso do comando *addInterface*, no qual o atributo *begin* de um elemento <area> pode receber o valor "now", especificando o NPT atual do nodeId, que identifica o vídeo principal MPEG em apresentação pelo decodificador do *hardware*.

Quando um comando de edição do documento NCL precisa ser enviado por difusão, um objeto de evento de fluxo DSM-CC deve ser criado, mapeando a sequência "*nclEditingCommand*" em um *id* de fluxo de evento selecionado (ver ABNT NBR 15606-2) e colocado como objeto em um carrossel de objetos DSM-CC. Um ou mais descritores de eventos de fluxo DSM-CC, com a *id* selecionada anteriormente, são então criados e enviados em outro fluxo elementar MPEG-2 TS. Esses descritores de eventos de fluxo geralmente têm sua referência de tempo configurada para zero, mas podem ter a execução prorrogada para um momento específico. O gerenciador de base privada deve obrigatoriamente se registrar como um ouvinte *"nclEditingCommand",* sendo notificado quando esse tipo de evento de fluxo chegar. O *commandTag* recebido é então utilizado pelo gerenciador de base privada para interpretar a semântica completa do *comando*. Se o parâmetro do comando, baseado em XML, for suficientemente curto, ele pode ser transportado diretamente na carga dos descritores do evento de fluxo, juntamente com os outros parâmetros do comando. Caso contrário, o privateDataPayload carrega um conjunto de pares de referência. No caso de arquivos (documentos NCL ou conteúdos de objetos NCL) recebidos sem solicitação (*pushed data*), cada par relaciona um conjunto de caminhos (path) de arquivos com sua respectiva localização no sistema de transporte. No caso de arquivos recebidos por demanda (*pulled data*) de um canal de interatividade ou localizados no próprio receptor, não há necessidade de enviar nenhum par de referências, exceto o par (uri, "null"} associado ao documento NCL ou à especificação de nó XML a ser adicionado pelo comando.

Receptores que apenas implementam o perfil NCL BDTV não são obrigados a lidar com os seguintes comandos: *pauseDocument*, *resumeDocument*, *addTransition*, *removeTransition*, *addTransitionBase* e *removeTransitionBase*.

O Ginga associa pelo menos uma base privada a cada canal de TV (conjunto de serviços): chamada base privada *default* do canal. Quando um canal é sintonizado, sua base privada default é aberta e ativada pelo gerenciador de base privada. As outras bases privadas devem obrigatoriamente ser desativadas. Através do canal sintonizado outras bases privadas podem ser abertas (ou criadas), mas no máximo uma pode ser associada com cada serviço do canal sintonizado. Quando o canal tem apenas um serviço, como é o caso da recepção one-seg, uma e somente uma base privada é associada a cada canal de TV (a base privada *default* do canal). Nesse caso, qualquer comando de edição que venha pelo canal de difusão para abrir uma base privada deve ser ignorado.

A base privada associada ao canal de TV deve obrigatoriamente ter o identificador (parâmetro baseId) igual ao "valor do campo network_id da NIT (table_id 0x40)". As outras possíveis bases privadas associadas a um serviço do canal de TV devem obrigatoriamente ter seu identificador (*baseId parameter*) igual ao_"valor do campo network_id da NIT.valor do campo program_number da PMT". O valor do *program_number* é aquele associado ao serviço correspondente.

As aplicações NCL residentes são gerenciadas em uma base privada específica.

Por razões de segurança, pode-se ativar apenas uma base privada por vez, entre aquelas controladas por meio do canal de transmissão sintonizado. A forma mais simples e restrita para gerenciar bases privadas é ter apenas

uma base privada aberta por vez, entre aquelas controladas por meio do canal de transmissão sintonizado. No entanto, o número de bases privadas que podem ser mantidas abertas é uma decisão da implementação específica do *middleware*.

NOTA 1 Apenas bases privadas criadas por comandos de edição NCL enviados pelo canal de difusão sintonizado a e a base privada default do canal de TV podem ser controladas por comandos de edição NCL provenientes do mesmo canal sintonizado. Assim, para receptores one-seg, apenas a base privada *default* associada ao canal de televisão sintonizado pode ser controlada pelos comandos de edição NCL enviados por meio do mesmo canal sintonizado.

NOTA 2 Os eventos NCLua e NCLet (eventos de comando de edição NCL) gerados por objetos imperativos de aplicações operando em uma base privada associada a um canal de TV podem controlar apenas as bases privadas associadas a esse canal.

Os comandos *add* sempre possuem entidades NCL como seus argumentos (parâmetros de comando baseados em XML). Se a entidade especificada já existe ou não, a consistência do documento deve obrigatoriamente ser mantida pelo formatador NCL, no sentido de que todos os atributos da entidade, classificados como necessários, sejam definidos. Há apenas uma exceção a essa regra - o atributo *interface* de um elemento <bind> filho de um elemento <link> pode ser deixado inconsistente, referindo-se a um elemento <area> a ser preenchido por um comando *addInterface* cujo atributo *begin* possui o valor “now”, conforme especificado na ABNT NBR 15606-2. Nesse caso, o elemento <link> deve obrigatoriamente ser avaliado assim que o comando *addInterface* for recebido.

É recomendado pautar o desenvolvimento de aplicações pelas boas práticas de usabilidade. Na interação através de controles remotos convencionais, recomenda-se utilizar uma tecla-padrão para habilitar e desabilitar aplicações no plano gráfico.

7.2 Identificação de recursos

Os comandos de edição “addDocument” e “addNode” devem obrigatoriamente ser usados para mapear as estruturas de dados do servidor para as estruturas de dados utilizadas no receptor, quando documentos XML referenciados por esses comandos (arquivos de documento NCL ou outros arquivos de documento XML, respectivamente) forem transmitidos por meio de um carrossel de objeto. Nesse caso, no mesmo carrossel de objetos que carrega a especificação do documento XML, um objeto de evento de fluxo deve obrigatoriamente ser transmitido, a fim de mapear o nome “*nclEditingCommand*” ao eventId do descritor de evento de fluxo DSM-CC, que carrega o comando de edição addDocument ou addNode. O privateDataPayload do descritor de evento de fluxo deve obrigatoriamente carregar um conjunto de pares de referência (uri, id). O parâmetro uri do primeiro par pode ter o esquema “x-sbtvd” (opcional) e o caminho absoluto do documento NCL ou da especificação de nó NCL (o caminho no servidor de dados). O parâmetro correspondente no par id deve obrigatoriamente se referir à IOR do documento NCL ou à IOR da especificação do nó NCL (carouselId, moduleId, objectKey; ver ABNT NBR 15606-3 e ISO/IEC 13818-6) no carrossel de objetos. Se outros sistemas de arquivos com conteúdo de mídias tiverem de ser transmitidos por meio de outros carrosséis de objetos para completar o comando de edição (como é usual no caso dos comandos addDocument ou addNode), outros pares (uri, id) devem obrigatoriamente estar presentes no comando. Nesse caso, o parâmetro uri pode ter o esquema “x-sbtvd” (opcional) e o caminho absoluto da raiz do sistema de arquivos (o caminho no servidor de datacast); já o parâmetro id correspondente no par deve obrigatoriamente se referir à IOR (carouselId, moduleId, objectKey; ver ABNT NBR 15606-3 e ISO/IEC 13818-6) de qualquer diretório ou arquivo secundário da raiz no carrossel de objetos (a IOR do *service gateway* do carrossel).

Geralmente, a transmissão de sistemas de arquivos por meio do uso de outros carrosséis de objeto, diferentes daqueles que carregam o objeto de evento, é necessária quando os sistemas de arquivos que representam a aplicação não podem ser modelados por uma única estrutura de dados. Um exemplo é apresentado em Nested Context Language 3.0, Part 9.

7.3 Comando startDocument

Com o intuito de ficar conforme a ITU-T H.761 para o Ginga-NCL, o comando startDocument é definido conforme Tabela 43.

Tabela 43 — Definição do comando startDocument

String de comando	Tag de comando	Descrição
startDocument (baseId, documentId, interfaceId, offset,nptBaseId, nptTrigger) [a, b]	0x07	Inicia a reprodução de um documento NCL em uma base privada ativa, iniciando a apresentação a partir de uma interface específica do documento. A referência do tempo transportada no campo nptTrigger estabelece o ponto de início do documento, com respeito à base de tempo NPT identificada pelo campo *nptBaseId*. Três casos podem ocorrer: a) Se nptTrigger for diferente de zero e for maior ou igual ao valor NPT corrente da base temporal NPT identificada por nptBaseId, espera-se até que NPT atinja o valor dado em nptTrigger e começa a exibição do documento do seu ponto inícial no tempo + *offset*; b) Se nptTrigger for diferente de zero e for menor que o valor de NPT corrente da base temporal NPT identificada por nptBaseId, o início da exibição do documento é imediata e deslocada no tempo de seu ponto incial do valor "*offset* + (NPT – nptTrigger)$_{segundos}$"; [c] c) Se nptTrigger for igual a 0, a exibição do documento é imediata e a partir de seu ponto inícial no tempo + *offset*

[a] O parâmetro *offset* especifica um valor de tempo.

[b] O nptTrigger é obrigatoriamente um valor de NPT, e o nptBaseId é obrigatoriamente um identificador de uma base de tempo NPT.

[c] Somente nesse caso, o parâmetro *offset* pode receber um valor negativo, mas *offset* + (NPT – nptTrigger)$_{segundos}$ deve ser um valor positivo.

7.4 Correspondência entre o controle do ciclo de vida usando AIT e *nclEditingCommand*

Para executar uma aplicação o receptor precisa saber algumas informações sobre ela: onde encontrar seus arquivos, como a aplicação é denominada e quando o receptor deve iniciá-la. Em aplicações vinculadas ao service sintonizado (*service-bound applications*), o Ginga pode usar uma tabela SI extra, denominada tabela de informações de aplicação (ou AIT) para inferir as informações mencionadas acima, além do oferecido pelo uso dos comandos de edição NCL. Cada serviço que tem uma aplicação associada a ele pode conter uma AIT, e cada AIT pode conter uma descrição de cada aplicação, que pode ser executada enquanto o serviço está sendo exibido.

A AIT é transmitida com o ID da tabela igual a 0x74. Cada AIT contém dois laços (*loops*) de nível superior. O primeiro deles é o laço comum, que contém um conjunto de descritores que se referem a todas as aplicações sinalizadas naquela AIT particular, ou que se aplicam ao serviço como um todo. O outro laço de nível superior é o laço de aplicações, que descreve as aplicações sinalizadas por aquela instância de AIT. Cada iteração no laço de aplicações descreve uma aplicação (ou seja, representa uma linha na tabela de aplicações sinalizadas). Entre essas informações, está o código de controle da aplicação que, no caso de aplicações NCL, deve obrigatoriamente adotar um dos valores mostrados na Tabela 44.

Tabela 44 — Controle do ciclo de vida de aplicações NCL usando AIT (campo AIT: application_type =“0x0009”)

campo AIT: application_ control_code	Descrição	*nclEditingCommand* equivalente	Restrição
AUTOSTART (0x01)	A aplicação é iniciada automaticamente quando o serviço for selecionado	addDocument (baseId, {uri, id}+); seguido de startDocument (baseId, documentId, interfaceId, offset, null, null) para adicionar (se já não estiver adicionada) e iniciar uma aplicação na base privada associada com o serviço sintonizado	a) A base privada é identificada pelo “valor do campo network_id da NIT”, ou o “valor do campo program_number da PMT que refere à AIT”; b) Para o comando addDocument equivalente: Há apenas um parâmetro uri que deve obrigatoriamente receber o valor definido no campo base_directory do ginga_ application_location_descriptor. Há apenas um parâmetro de id que deve obrigatoriamente receber o valor IOR correspondente ao documento definido no campo initial_entity do ginga_ application_location_descriptor. c) Para o comando addDocument equivalente: O atributo id do elemento <ncl> deve receber o mesmo valor do campo application_id da estrutura application_identifier da AIT; d) A aplicação NCL deve obrigatoriamente ser iniciada a partir de todas as interfaces do elemento <body>; e) A aplicação NCL deve obrigatoriamente ser iniciada imediatamente, já que não pode ter seu tempo inicial se referindo a um valor NPT, que supõe-se ser "do it now"; f) A aplicação NCL deve obrigatoriamente ser iniciada do seu começo, já que o parâmetro *offset* é assumido como "0".
PRESENT (0x02)	A aplicação não é iniciada automaticamente. Ela é colocada em uma lista de aplicações disponíveis no receptor, podendo ser selecionada por um usuário para ser então iniciada	Semelhante ao AUTOSTART, exceto que a aplicação é iniciada pelo usuário e não automaticamente	Ver AUTOSTART
DESTROY (0x03)	A aplicação é interrompida	stopDocument (baseId, documentId) para interromper uma aplicação na base privada associada ao serviço de TV sintonizado ou na base privada *default* associada com o canal de TV sintonizado O parâmetro baseId deve obrigatoriamente identificar a base privada associada ao serviço ou a base privada *default* associada ao canal de TV sintonizado	a) O parâmetro baseId nos comandos de edição NCL deve receber "o valor do campo network_id da NIT” ou “o valor do campo program_number da PMT que refere a AIT"; b) O atributo *id* do elemento <ncl> deve obrigatoriamente receber o mesmo valor do campo application_id da estrutura application_identifier da AIT; c) A aplicação NCL deve obrigatoriamente ser imediatamente interrompida, já que não pode ter seu tempo final se referindo a um valor NPT, que é assumido como sendo "do it now"

Tabela 44 (continuação)

campo AIT: application_ control_code	Descrição	*nclEditingCommand* equivalente	Restrição
KILL (0x04)	A aplicação é interrompida e removida do receptor	stopDocument (baseId, documentId) seguido de removeDocument(baseId, documentId) para interromper uma aplicação, na base privada associada ao serviço sintonizado ou na base privada *default* associada com o canal de TV sintonizado, e então removê-la da base privada O parâmetro baseId deve obrigatoriamente identificar a base privada associada ao serviço ou a base privada *default* associada ao canal de TV sintonizado	a) O parâmetro baseId nos comandos de edição NCL deve receber "o valor do campo network_id da NIT" ou "o valor do campo program_number da PMT que refere a AIT"; b) O atributo *id* do elemento <ncl> deve obrigatoriamente receber o mesmo valor do campo application_id da estrutura application_identifier da AIT; c) A aplicação NCL deve obrigatoriamente ser imediatamente interrompida, já que não pode ter seu tempo final se referindo a um valor NPT, que é assumido como sendo "do it now" d) Uma aplicação NCL não pode ser reiniciada após ter sido interrompida pelo controle KILL, sem ter todo o processo AUTOSTART refeito (exceto se ela for iniciada por PREFETCH ou UNBOUND)
PREFETCH (0x05)	O exibidor NCL é preparado. O gerente de base privada deve obrigatoriamente aguardar o comando addDocument (baseId, {uri, id}+) para adicionar a aplicação na base privada baseId; e o comando startDocument (baseId, DocumentId, interfaceId, offset, nptBaseId, nptTrigger) para disparar a aplicação		a) O parâmetro baseId nos comandos de edição NCL deve receber "o valor do campo network_id do NIT" ou "o valor do campo program_number da PMT que refere a AIT".
REMOTE (0x06)	A aplicação remota só estará disponível para ser executada após a seleção do serviço. O exibidor NCL é preparado e o gerente de base privada deve obrigatoriamente aguardar o comando addDocument (baseId, {uri, id}+) para adicionar a aplicação na base privada e o comando startDocument (baseId, documentId, interfaceId, offset, nptBaseId, nptTrigger) para disparar a aplicação		a) O parâmetro baseId nos comandos de edição NCL deve receber "o valor do campo network_id do NIT" ou "o valor do campo program_number da PMT que refere a AIT".
UNBOUND (0x07)	A aplicação não é iniciada automaticamente. Ela é colocada em uma lista de aplicações disponíveis no receptor, podendo ser selecionada por um usuário para ser iniciada. A aplicação pode ser armazenada para ser iniciada em um momento posterior	Semelhante ao PRESENT, exceto que a aplicação pode ser armazenada para exibição futura	Ver PRESENT
STORE (0x08)	O mesmo comportamento de PREFETCH	Ver PREFETCH	Ver PREFETCH

7.5 Valores *default*

Quando o parâmetro *baseId* de um *nclEditingCommand* transportado em um fluxo de transporte (TS) é especificado como "nulo", ele deve obrigatoriamente assumir o valor do campo network_id da NIT correspondente (table_id 0x40).

Quando o parâmetro *baseId* de um *nclEditingCommand* proveniente de um objeto NCLua executado em uma determinada base privada não é especificado, deve obrigatoriamente ser assumido o mesmo valor *baseId* dessa base privada.

Em um comando AddDocument, se todos os recursos da aplicação estiverem sob a mesma raiz, o parâmetro *id* do par {uri, id} pode ser omitido.

Em um *nclEditingCommand*, se o campo eventNPT de um descritor de eventos de fluxo for diferente de “0”, o valor do campo eventId do descritor de eventos de fluxo deve estar listado em um objeto de eventos de fluxo DSM-CC cujo campo *id* de seu STR_NPT_USE tap (campo use do tap) identifica a base de tempo NPT (o campo *id* do tap deve obrigatoriamente conter o contentId da base de tempo).

Em um comando startDocument, se o parâmetro nptBaseId for “null”, o valor do campo eventId do descritor de eventos de fluxo deve estar listado em um objeto de eventos de fluxo DSM-CC cujo campo *id* de seu STR_NPT_USE tap (campo use do tap) identifica a base de tempo NPT (o campo *id* do tap deve obrigatoriamente conter o contentId da base de tempo).

Em um comando startDocument, se o valor de nptTrigger for especificado como "nulo", o campo eventNPT do descritor de eventos de fluxo identifica o tempo NPT que define o tempo de início da exibição de uma aplicação.

Em um comando startDocument, se o parâmetro interfaceId for especificado como "nulo", todos os elementos <port> do elemento <body> devem obrigatoriamente ser disparados (iniciados).

Em um comando startDocument, se o parâmetro offset for especificado como "nulo", deve-se obrigatoriamente assumir "0" como valor.

Se o parâmetro compositeId for especificado como "nulo", o elemento <body> do documento NCL documentId deve obrigatoriamente ser considerado como a composição a ser editada.

7.6 Tratamento de exceções

Recomenda-se que, se o parâmetro *baseId* de um *nclEditingCommand* transportado em um fluxo TS com identificador network_id tiver um valor diferente do valor network_id, ou do valor network_id.program_number (do serviço), o último no caso apenas de receptores full-seg, ele seja ignorado.

Recomenda-se que, se o parâmetro *baseId* de um *nclEditingCommand* proveniente de um objeto NCLua em execução em uma determinada base privada tiver um valor diferente do valor *baseId* dessa base privada, ele seja ignorado.

Receptores que apenas implementam o perfil NCL BDTV não são obrigados a tratar os seguintes *nclEditingCommands*: *pauseDocument*, *resumeDocument*, *addTransition*, *removeTransition*, *addTransitionBase* e *removeTransitionBase*.

Recomenda-se que um *nclEditingCommand* que pode causar uma inconsistência no documento, ou que se refere a um elemento NCL inexistente, ou a qualquer outro identificador inexistente, seja ignorado. Há apenas uma exceção a essa regra – o atributo *interface* de um elemento <bind> filho de um elemento <link> pode ser deixado inconsistente, referindo-se a um elemento <area> a ser preenchido por um comando *addInterface* cujo atributo *begin* possui o valor “now”, conforme especificado na ABNT NBR 15606-2. Nesse caso, o <link> deve ser avaliado obrigatoriamente assim que o comando *addInterface* for recebido.

Recomenda-se que, se em um comando startDocument o parâmetro *nptBaseId* for diferente de “null” e se referir a uma base de tempo inexistente, o comando seja ignorado.

8 API NCLua

8.1 Considerações gerais

A linguagem de script adotada pelo Ginga-NCL para implementar objetos imperativos em documentos NCL é a linguagem *Lua* (elementos de <media> do tipo application/x-ginga-NCLua ou do tipo application/x-ncl-NCLua).

Além da biblioteca padrão Lua, os seguintes módulos devem obrigatoriamente ser implementados: canvas; event, settings, e persistent.

Recomenda-se que as funções desses módulos utilizem a seguinte política de tratamento de erros (nessa ordem):

1. Se o número de parâmetros da chamada for maior do que o esperado, a função ignora os parâmetros adicionais;

2. Se um parâmetro opcional *P* da chamada for omitido, a função assume o valor *default* de *P*, definido na ABNT NBR 15606-2 ou nesta parte da ABNT NBR 15606;

3. Se o tipo de um parâmetro *P* da chamada for diferente do esperado, a função converte *P* para o tipo esperado, caso a conversão seja possível. Caso contrário, a função gera um erro;

4. Se o valor de um parâmetro *P* da chamada for inválido, a função assume o valor recomendado de *P*, definido na Norma ou neste Guia. Se não existir valor recomendado, a função gera um erro;

5. Se os parâmetros da chamada forem válidos e mesmo assim a chamada falhar, a função gera um erro.

Nos itens 3-5 anteriores, por “gerar um erro” entende-se que:

1. Se a função possuir retorno de *status* de erro – por exemplo, event.post() –, então a função retorna o valor que indica erro seguido de uma mensagem (*string*) de erro.

2. Se a função não possuir retorno de status de erro, então a função dispara uma exceção Lua – chamada error() ou lua_error() – contendo uma mensagem (*string*) de erro. Se o *script* não capturar essa exceção – chamada pcall() – então o objeto de mídia associado é abortado.

8.2 Módulo *canvas*

8.2.1 Considerações gerais

O módulo canvas oferece uma API gráfica para ser utilizada em uma aplicação NCLua. Usando a API, é possível desenhar linhas, retângulos, fontes, imagens etc.

Por motivos de compatibilidade com as regras *default* apresentadas em outros sistemas e em procedimentos de aceleração por *hardware* usado nos receptores, a operação canvas:compose (x, y: number; src: canvas; [src_x, src_y, src_width, src_height: number]) deve satisfazer as seguintes equações:

$$\alpha_R = \alpha_A + \alpha_B \times (1 - \alpha_A)$$

$$C_R = \frac{C_A\alpha_A + C_B\alpha_B \times (1 - \alpha_A)}{\alpha_R}$$

onde

c_R cor resultante normalizada entre 0 e 1;

α_R component alfa resultante normalizado entre 0 e 1;

c_A cor do canvas fonte (src) normalizada entre 0 e 1;

α_A component alfa do canvas fonte (src) normalizado entre 0 e 1;

c_B cor do canvas de destino (canvas) normalizada entre 0 e 1;

α_B component alfa do canvas de destino (canvas) normalizado entre 0 e 1.

8.2.2 Valores *default*

Em todas as operações **canvas: drawXXX,** a largura da linha deve obrigatoriamente ser considerada como 1 pixel.

Na operação **canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number)**, as unidades de ângulo devem obrigatoriamente ser consideradas como graus. Se ang_start ou ang_end forem *nil* , uma condição de erro deve ser reportada.

Na operação **canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number)**, o ângulo de 0 grau está na coordenada Y mais elevada da elipse e a progressão do ângulo segue o movimento no sentido horário.

A operação **canvas:attrRotation (degrees: number)** segue o movimento no sentido horário.

8.2.3 Tratamento de exceções

Para as funções do módulo Canvas, recomenda-se o seguinte comportamento:

- **canvas:new (image_path:string)**. Se o caminho *image_path* for inválido, a função retorna nil e uma mensagem de erro.
- **canvas:new (width, height:number)**. Se *width, height* < 0, a função assume 0.
- **canvas:attrColor (r, g, b, a:number)**. Se *r, g, b, a* < 0, a função assume 0. Se r, g, b, a > 255, a função assume 255.
- **canvas:attrColor (clr_name:string)**. Se *clr_name* for diferente de todas as cores válidas, a função assume “black”.
- **canvas:attrFont (face:string, size:number, style:string)**. Se face não for suportada ou *face*=nil, a função assume “Tiresias” (*default*). Se *size* < 0, a função assume 0. Se *style* for diferente de “bold”, “italic” e “bold-italic”, a função assume *style*=nil.
- **canvas:attrClip (x, y, width, height:number)**. Se *x, y, width, height* < 0, a função assume 0. Se *x* for maior do que a largura do canvas (canvas.width), a função assume canvas.width. Se y for maior do que a altura do canvas (canvas.height), a função assume canvas.height. Se *x*+*width* > canvas.width, a função assume *width*=canvas.width–x. Se *y*+*height* > canvas.height, a função assume *height*=canvas.height–y.
- **canvas:attrCrop (x, y, width, height:number).** Similar à canvas:attrClip().
- **canvas:attrOpacity (opacity:number)**. Se *opacity* < 0, a função assume 0. Se *opacity* > 255, a função assume 255.

- **canvas:attrScale (width, height:number)**. Se *width, height* < 0, a função assume 0. Se o parâmetro implícito “canvas” for o canvas principal, a função gera um erro. Notar que “escalar” um canvas significa alterar a escala (*default* 1.0) utilizada no momento da composição desse canvas com outros objetos canvas. Logo, a função canvas:attrScale() não modifica a matriz de pixels do canvas. Ela apenas especifica os fatores (horizontal e vertical) utilizados para interpolar a matriz de pixels desse canvas no momento imediatamente anterior ao da operação de composição.

- **canvas:drawLine (x1, x2, y1, y2:number)**. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:drawRect (mode:string, x, y, width, height:number)**. Se *mode* for diferente de “frame” e “fill”, a função assume “fill”. Se *width, height* < 0, a função assume 0. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:drawRoundRect (mode:string, x, y, width, height, arc_width, arc_height:number)**. Se *arc_width, arc_height* < 0, a função assume 0. O tratamento dos demais parâmetros é similar ao da canvas:drawRect().

- **canvas:drawPolygon (mode:string)**. Se *mode* for diferente de “open”, “close” e “fill”, a função assume “fill”. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas. Se a função drawer(), retornada pela canvas:drawPolygon(), for chamada mais de uma vez com o parâmetro “nil”, a função gera um erro.

- **cavans:drawEllipse (mode:string, xc, yc, width, height, ang_start, ang_end:number)**. Se *mode* for diferente de “arc” e “fill”, a função assume “fill”. Se *width, height* < 0, a função assume 0. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:drawText (x, y:number, text:string)**. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:clear([x, y, width, height:number])**. Similar à canvas:drawRect().

- **canvas:composse (x, y:number, src:canvas, [src_x, src_y, src_width, src_height:number])**. Se *src_width, src_height* < 0, a função assume 0. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:pixel (x, y, r, g, b, a:number)**. Se *r, g, b, a* < 0, a função assume 0. Se *r, g, b, a* > 255, a função assume 255. Se a operação de desenho exceder o tamanho do canvas, a função desenha apenas os pontos que estão “dentro” do canvas.

- **canvas:measureText (text:string)**. Notar que as medidas do texto renderizado dependem apenas da fonte utilizada.

8.3 Módulo *event*

8.3.1 Considerações gerais

Esse módulo oferece uma API para manipulação de eventos. Usando a API, o formatador NCL pode se comunicar com uma aplicação NCLua de forma assíncrona.

Em **event.register ([pos: number]; f: function; [class: string]; [...: any]),** a posição de registro inicial é 1.

Em **event.post** da class='si' e type='epg', o subcampo hasInteractivity da tabela de dados deve obrigatoriamente considerar o carousel_descriptor compatível, os comandos de edição NCL e as tabelas de AIT, a fim de especificar se o evento EPG tem (ou não) uma aplicação.

A função `event.post()` e o tratador registrado em `event.register()` recebem eventos como parâmetros. Um evento é descrito por uma tabela Lua comum, onde o campo da classe é obrigatório e identifica a classe do evento.

A fim de dar suporte a eventos ponteiros, NCLua deve obrigatoriamente incluir a seguinte classe:

Classe `pointer`:

evt = { class='pointer', type: string, x=number, y=number }

* o tipo pode ser "press", "release" ou "move"

* X e Y se referem às coordenadas da ocorrência do evento ponteiro

NOTA Na classe pointer, o filtro dependente da classe só pode ser *type*.

EXEMPLO evt = { class='pointer', type='press', x=20, y=50 }

A classe *sms* é obrigatória apenas na implementação do Ginga para receptores *one-seg* e deve obrigatoriamente estar de acordo com as seguintes especificações:

---

classe sms:

Uma aplicação NCLua envia dados, utilizando SMS, por meio da postagem de eventos na seguinte forma:

evt = { class='sms', type='send', to='string', value=string [, id:string]}

O campo `to` contém o número de destino (número do telefone ou número da conta). Se não forem especificados, a região e os prefixos de código do país receberão a região e os códigos do país de onde a mensagem está sendo enviada.

O campo `value` contém o conteúdo da mensagem.

O campo `id` pode ser usado para identificar o SMS que será enviado. A aplicação é responsável por definir o valor de `id` e garantir sua unicidade.

Um evento de confirmação deve obrigatoriamente ser enviado de volta à aplicação NCLua, seguindo o formato:

evt = { class='sms', type='send', to:string, sent:Boolean [,error:string] [, id:string] }

Na mensagem de confirmação, o campo `to` deve obrigatoriamente ter o mesmo valor que no evento original postado pela aplicação NCLua. O campo `sent` notifica se o SMS foi despachado pelo dispositivo (verdadeiro) ou não. O campo `error` é opcional. Se o valor do campo `sent` for falso, ele pode conter uma mensagem de erro detalhada. Se o SMS original for postado com o campo `id` definido, o evento de confirmação deve chegar obrigatoriamente com o mesmo valor de id. Desse modo, a aplicação NCLua pode fazer uma associação entre ambos os eventos e lidar com várias mensagens SMS enviadas simultaneamente.

De forma similar, uma aplicação NCLua se registra para receber mensagens SMS, postando eventos na forma:

evt = { class='sms', type ='register', port:number }

O campo `port` deve receber obrigatoriamente um número válido de porta TCP. Para o cumprimento das normas GSM (3GPP TS 23.040 V6.8.1, de 2006-10), esse valor deve obrigatoriamente estar no intervalo [16000,16999].

Eventos recebidos pelo tratador possuem o seguinte formato:

evt = { class='sms', type='receive', from:string, port:number, value:string }

O campo `port` é definido como no tipo = 'register'. O campo `from` contém o número da mensagem de origem (número do telefone ou número da conta). O prefixo da região e o código do país podem ser omitidos se forem iguais aos do receptor. O campo `value` tem o conteúdo da mensagem.

A qualquer momento, a aplicação pode solicitar parar de receber mensagens SMS em uma determinada porta, registrando o evento:

evt = { class='sms', type='unregister', port:number }

O campo porta é definido como no tipo = 'register'.

No momento em que a apresentação da mídia NCLua parar, a implementação do *middleware* deve obrigatoriamente assegurar que todas as portas sejam desregistradas.

NOTA 1 É recomendado que uma implementação específica de *middleware* trate questões como autenticação etc.

NOTA 2 Na classe sms, o filtro dependente da classe só pode ser *from* e *port*, nessa ordem.

NOTA 3 A finalidade do número de porta é evitar conflitos entre as mensagens comuns SMS recebidas por um usuário e as mensagens SMS que devem obrigatoriamente ser tratadas apenas pela aplicação.

NOTA 4 Uma implementação do Ginga-NCL obrigatoriamente retorna *false* imediatamente a cada chamada para event.post () que usa uma classe de evento não suportada. É recomendado que a aplicação NCLua capture essa condição de erro para verificar se o envio do SMS falhou.

8.3.2 Valores *default*

Em *ncl class,* os eventos podem ser direcionados para âncoras específicas ou para o nó inteiro. Isso é identificado pelo campo *label*, que assume o nó inteiro quando ausente.

8.3.3 Tratamento de exceções

Para as funções do módulo Event, recomenda-se o seguinte comportamento.

- **event.post ([dst:string], evt:event)**. Se *dst* for diferente de "in" e "out", a função assume "out". Se *evt.class* for uma classe conhecida e *evt* não possui os campos obrigatórios dessa classe, ou se o valor de algum desses campos for inválido, a função gera um erro.

- **event.timer (time:number, f:function)**. Se *time* < 0, a função assume 0.

- **event.register ([pos:number], f:function, [class:string], [...:any])**. Se *pos* < 0 ou *pos* for maior do que o tamanho da fila de tratadores registrados, a função registra *f* no final da fila. Quando um tratador é registrado em uma posição ocupada por outro, cada posição de tratador, a partir dessa posição, deve obrigatoriamente ser incrementada para dar lugar à nova inserção. Quando um tratador é removido, todas as outras posições de tratador, a partir da posição de tratador eliminada, devem ser decrementadas.

- **event.unregister (f:function)**. Se *f* não estiver registrada, a chamada é ignorada.

- **Eventos da classe "key"**. Notar que o *script* NCLua pode postar eventos dessa classe.

- **Eventos da classe "edit"**. Se o campo *data* não for um XML válido, o evento é ignorado.
- **Eventos da classe "settings".** Se o *script* tentar alterar o valor de alguma variável do sistema, o objeto de mídia associado é abortado.

Bibliografia

[1] Soares, L.F.G. et al. *Nested Context Language 3.0 – Implementors Guide.* Relatório Técnico, Departamento de Informática, PUC-Rio, n ° 13/09. Rio de Janeiro. Janeiro de 2009. ISSN 0103-9741.