NORMA
BRASILEÑA

ABNT NBR 15606-7

Primeira edición
21.03.2011

Válida a partir de
21.04.2011

Televisión digital terrestre — Codificación de datos y especificaciones de transmisión para radiodifusión digital
Parte 7: Ginga-NCL — Directrices operacionales para las ABNT NBR 15606-2 y 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 978-85-07-02784-3

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS

Número de referencia
ABNT NBR 15606-7:2011
55 páginas

ABNT
Av.Treze de Maio, 13 - 28º andar
20031-901 - Rio de Janeiro - RJ
Tel.: + 55 21 3974-2300
Fax: + 55 21 2220-1762
abnt@abnt.org.br
www.abnt.org.br

Índice

Página

Prefacio

La Associação Brasileira de Normas Técnicas (ABNT) es el Fórum Nacional de Normalización. Las Normas Brasileñas, cuyo contenido es responsabilidad de los Comités Brasileños (ABNT/CB), de los Organismos de Normalización Sectorial (ABNT/ONS) y de las Comisiones de Estudios Especiales (ABNT/CEE), son elaboradas por Comisiones de Estudio (CE), formadas por representantes de sus sectores implicados de los que forman parte: productores, consumidores y neutrales (universidades, laboratorios y otros).

Los Documentos Técnicos ABNT se elaboran de acuerdo con las reglas de Directivas ABNT, Parte 2.

La Associação Brasileira de Normas Técnicas (ABNT) llama la atención sobre la posibilidad de que algunos de los elementos de este documento pueden ser objeto de derechos de patente. La ABNT no debe ser considerada responsable por la identificación de cualesquiera derechos de patente.

La ABNT NBR 15606-7 fue elaborada por la Comisión de Estudio Especial de Televisión Digital (ABNT/CEE-85). El Proyecto circuló en Consulta Nacional según Edicto nº 12, de 21.12.2010 a 18.02.2011, con el número de Proyecto 00:001.85-006/7.

En caso que surja cualquier duda con relación a la interpretación de la versión en español siempre deben prevalecer las prescripciones de la versión en portugués

Esta Norma se basa en los trabajos del Foro del Sistema Brasileño de Televisión Digital Terrestre, según lo establecido en el decreto presidencial n° 5.820, del 29.06.2006.

La ABNT NBR 15606, bajo el título general “Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital”, está previsto que contenga las siguientes partes:

— Parte 1: Codificación de datos;

— Parte 2: Ginga-NCL para receptores fijos y móviles – Lenguaje de aplicación XML para codificación de aplicaciones;

— Parte 3: Especificación de transmisión de datos;

— Parte 4: Ginga-J – Ambiente para la ejecución de aplicaciones procedurales;

— Parte 5: Ginga-NCL para receptores portátiles – Lenguaje de aplicación XML para codificación de aplicaciones;

— Parte 6: Java DTV 1.3;

— Parte 7: Ginga-NCL – Directrices operacionales para las ABNT NBR 15606-2 y ABNT NBR 15606-5.

Esta versión en español es equivalente a la ABNT NBR 15606-7:2011, de 21.03.2011.

Esta versión en español fue publicada en 19.05.2011.

El Alcance de esta Norma Brasileña en inglés es el siguiente:

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.

Introducción

La Asociación Brasileña de Normas Técnicas (ABNT) alerta que la exigencia de conformidad con este documento ABNT puede involucrar el uso de un derecho de propiedad relativo a NCL.

ABNT no toma posición alguna sobre evidencias, validez y alcance de ese derecho de propiedad.

El propietario de ese derecho de propiedad aseguró a ABNT que está preparado para negociar licencias sobre términos y condiciones razonables y no discriminatorias con los solicitantes. Sobre esto, una declaración del propietario de ese derecho está registrada en la ABNT. Informaciones se pueden obtener a través de:

Pontificia Universidad Católica de Rio de Janeiro, Departamento de Transferencia de Tecnología

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

ABNT alerta sobre la posibilidad de que algunos de los elementos de este documento ABNT pueden ser objeto de otras patentes y derechos de propiedad además de los identificados arriba. ABNT no puede ser considerada responsable por la identificación de cualesquier derechos de patente.

Esta parte de la ABNT 15606 complementa y aclara la estandarización del lenguaje NCL, que permite la especificación de presentaciones multimedia interactivas. Esta parte de la ABNT NBR 15606 forma parte de las especificaciones de codificación de datos para el Sistema Brasileño de Televisión Digital Terrestre, y provee las directrices operacionales relativas a la máquina de presentación Ginga-NCL del *middleware* llamado Ginga.

Esta parte de la ABNT 15606 se destina, primordialmente, a las entidades que están implementando terminales y/o estándares basados en el Ginga. También se destina a los desarrolladores de aplicaciones que utilizan las funcionalidades del Ginga y de sus API. El *middleware* Ginga tiene como objetivo asegurar la interoperabilidad de las aplicaciones en diferentes implementaciones de plataformas que lo soportan.

Las aplicaciones Ginga se clasifican bajo dos categorías, dependiendo si la aplicación inicialmente procesada posee contenido de naturaleza declarativa o imperativa. Estas categorías de aplicaciones son llamadas aplicaciones declarativas y aplicaciones imperativas, respectivamente. Los ambientes de aplicación son igualmente clasificados en dos categorías, dependiendo de si procesan aplicaciones declarativas o imperativas, siendo denominados Ginga-NCL y Ginga-J, respectivamente, en el Sistema Brasileño de Televisión Digital Terrestre. Solamente el ambiente Ginga-NCL es obligatorio en receptores portátiles. Usando el Ginga-NCL, el *middleware* Ginga da soporte a código imperativo a través del lenguaje Lua.

En la Sección 5 se discute como la semántica de los elementos y atributos NCL 3.0 son interpretados, cuáles son sus valores posibles y predeterminados y las directrices operacionales para un formateador NCL (agente del usuario) al gestionar estos elementos y atributos. En la sección 6 se establecen los procedimientos de los exhibidores de los objetos de media NCL. La Sección 7 presenta las directrices para la gestión del ciclo de vida de una aplicación NCL.

NOTA TAGinga es una marca comercial de la PUC-Rio y de la UFPB y, NCL es una marca comercial de la PUC-Rio.

Televisión digital terrestre — Codificación de datos y especificaciones de transmisión para radiodifusión digital
Parte 7: Ginga-NCL — Directrices operacionales para las ABNT NBR 15606-2 y ABNT NBR 15606-5

1 Alcance

Esta parte de la ABNT NBR 15606 detalla y explica la especificación del lenguaje de aplicación XML denominada NCL (Nested Context Language), el lenguaje declarativo del middleware Ginga, la codificación de datos y la transmisión de datos para radiodifusión digital, proveyendo las directrices operacionales para una implementación de acuerdo con las ABNT NBR 15606-2 y ABNT NBR 15606-5.

2 Referencias normativas

Los documentos relacionados a continuación son indispensables a la aplicación de esta Norma. Para referencias fechadas, se aplican apenas las ediciones citadas. Para referencias no fechadas, se aplican las ediciones más recientes de dicho documento (incluyendo enmiendas).

ABNT NBR 15604, Televisión digital terrestre – Receptores

ABNT NBR 15606-1, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 1: Codificación de datos*

ABNT NBR 15606-2:2007, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 2: Ginga-NCL para receptores fijos y móviles – Lenguaje de aplicación XML para codificación de aplicaciones*

ABNT NBR 15606-3, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 3: Especificación de transmisión de datos*

ABNT NBR 15606-5:2008, *Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 5: Ginga-NCL para receptores portátiles – Lenguaje de aplicación XML para codificación de aplicaciones*

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*

Guía Postal Brasileña de la Empresa Brasileña de Correos y Telégrafos, disponible en http://www.correios.com.br/cep/cep_estrutura.cfm

3 Términos y definiciones

A los efectos de esta parte de la ABNT NBR 15606, se aplican los siguientes términos y definiciones.

3.1
agente del usuario
cualquier programa que interprete un documento escrito en el lenguaje NCL según los términos presentes en las ABNT NBR 15606-2 y ABNT NBR 15606-5

NOTA Un agente del usuario puede exhibir un documento, procurando garantizar que las relaciones especificadas por el autor entre los objetos se respeten, leer en voz alta, imprimir, convertirlo para otro formato etc.

3.2
ambiente de la aplicación
contexto o ambiente del *software* en que se procesa una aplicación

3.3
ambiente de aplicación declarativa
ambiente que soporta el procesamiento de aplicaciones declarativas

NOTA El agente de usuario NCL (formateador) es un ejemplo de ambiente de aplicación declarativa.

3.4
aplicación
informaciones que expresan un conjunto específico de procedimientos observables

3.5
aplicación declarativa
aplicación que se inicia con, y utiliza principalmente, informaciones declarativas para expresar su comportamiento

NOTA Una instancia de documento NCL es un ejemplo de aplicación declarativa.

3.6
aplicación nativa
función intrínseca implementada por una plataforma de receptor

NOTA La exhibición de subtítulos es un ejemplo de una aplicación nativa.

3.7
atributo
parámetro que representa una propiedad

3.8
atributo de un elemento
propiedad de un elemento XML

3.9
audio principal
audio básico
flujo básico de audio cuya component_tag es igual a 0x10 para el receptor *full-seg* y 0x83 ó 0x85 para el receptor *one-seg*

3.10
autor
persona que especifica documentos NCL

3.11
canal de interactividad
canal de retorno
mecanismo de comunicación que provee la conexión entre el receptor y un servidor remoto

3.12
carácter
letra específica u otro símbolo de identificación

EJEMPLO "A"

3.13
ciclo de vida de una aplicación
periodo de tiempo, desde el momento en que la aplicación se carga hasta el momento en que se destruye

3.14
codificación de caracteres
mapeado entre un valor de entrada entero y el carácter textual que se representa por ese mapeado

3.15
comando y control del almacenamiento de media digital
DSM-CC
método de control que provee acceso a un archivo o flujo de servicios digitales interactivos

NOTA Para más informaciones sobre DSM-CC, se puede consultar la ISO/IEC 13818-6.

3.16
contenido NCL
conjunto de informaciones que consiste en un documento NCL y un grupo de datos, incluyendo objetos (objetos de media o de ejecución), acompañando al documento NCL

3.17
elemento
unidad de estructuración del documento delimitada por delimitadores XML

NOTA Un elemento, generalmente, se delimita por una marca de inicio y una de final, con excepción de un elemento vacío, que es delimitado por una marca de elemento vacío.

3.18
elemento *property*
elemento NCL que define un nombre de propiedad y su valor asociado

3.19
entidad de la aplicación
unidad de información que expresa parte de una aplicación

3.20
evento
acontecimiento en el tiempo, que puede ser instantáneo o tener una duración mensurable

3.21
exhibidor de media
componente de un ambiente de aplicación que decodifica o ejecuta un tipo específico de contenido

3.22
flujo elemental
ES
flujo básico que contiene datos de vídeo, audio o datos privados

NOTA Un flujo elemental se transmite en una secuencia de paquetes PES con una única *stream_id*.

3.23
flujo de transporte
flujo de transporte MPEG-2 conteniendo el empaquetado y multiplexación de video, audio y señales de datos para los sistemas de transmisión digital

3.24
fuente
mecanismo que permite la interpretación específica de un carácter especial

EJEMPLO Tiresias, 12 puntos.

NOTA En la práctica, un formato de fuente incorpora algunos aspectos de una codificación de caracteres.

3.25
formateador NCL
componente de *software* que es responsable de recibir la especificación de un documento NCL y controlar su presentación con el objetivo de asegurar que se respeten las relaciones entre los objetos de media especificados por el autor.

NOTA Los términos renderizador de documento, agente de usuario y exhibidor se utilizan en el mismo sentido de formateador de documento.

3.26
eXtensible HTML
XHTML
versión extendida del HTML

NOTA En la especificación del XHTML, un documento HTML es reconocido como aplicación XML.

3.27
identificador de recurso uniforme
URI
método de direccionamiento que permite el acceso a objetos en una red

3.28
informaciones sobre servicios
SI
datos que describen programas y servicios

3.29
interfaz de programación de aplicaciones
API
bibliotecas de *software* que ofrecen acceso uniforme al sistema de servicios

3.30
lenguaje de *script*
lenguaje utilizado para describir un contenido de objeto activo, integrado en documentos NCL y HTML

3.31
locator
identificador que provee la referencia para una aplicación o recurso

3.32
máquina de presentación
subsistema de un receptor que evalúa y presenta aplicaciones declarativas compuestas de contenidos de media, como audio, vídeo, gráficos y texto, esencialmente con base en reglas de presentación definidas por la máquina de presentación

NOTA La máquina de presentación es responsable de controlar el procedimiento de la presentación e iniciar otros procedimientos, en respuesta a las requisiciones del usuario y a otros eventos.

EJEMPLO navegador HTML y formateador NCL.

3.33
nudo NCL
elementos NCL de <media>, <context>, <body> o <switch>

3.34
objeto de media
conjunto de fragmentos de datos que puede representar un contenido de media o un programa escrito en lenguaje específico

3.35
perfil
especificación de una clase de recursos que ofrece diferentes niveles de funcionalidad en un receptor

3.36
perfil *one-seg*
servicio que se puede recibir por un sintonizador de banda estrecha (430 KHz), por lo tanto, con ahorro de energía

NOTA El perfil *one-seg* también se conoce como perfil portátil.

3.37
perfil *full-seg*
servicio que necesita un demodulador de banda ancha (5,7 MHz) para ser recibido

NOTA Dependiendo de la configuración de transmisión y de las funcionalidades específicas del receptor, el servicio se puede recibir por receptores móviles o apenas por receptores fijos, sin embargo, sin los beneficios del ahorro de energía. La resolución de vídeo puede ser de alta definición o de definición *default*.

3.38
plataforma del receptor
hardware del receptor, sistema operativo y bibliotecas de *software* nativo elegidos por el fabricante

3.39
recurso
objeto o servicio de red identificado unívocamente en una red

3.40
sintonización
acto de transferencia entre dos flujos de transporte MPEG

NOTA La transferencia entre dos servicios SBTVD, realizada en el mismo flujo de transporte, no es sintonización.

3.41
tiempo normal de exhibición
NPT
coordenada temporal absoluta que representa la posición en un flujo

3.42
usuario
persona que interactúa con un agente de usuario para ver, oír o utilizar un documento NCL

3.43
video principal

video básico
flujo básico de vídeo cuya component_tag es igual a 0x00 para el receptor *full-seg* y 0x81 para el receptor *one-seg*

4 Símbolos y abreviaturas

A los efectos de esta parte de la ABNT NBR 15606, se aplican los siguientes símbolos y abreviaturas.

API	*Application Programming Interface* (Interfaz de programación de aplicaciones)
BML	*Broadcast Markup Language*
CSS	*Cascading Style Sheets*
DSM-CC	*Digital Storage Media Command and Control*
DTV	*Digital Television (*televisión 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 Brasileño de Televisión Terrestre Digital
SMIL	*Synchronized Multimedia Integration Language*
TS	*Transport Stream* (flujo 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 en los perfiles EDTV y BDTV

5.1 Consideraciones generales

Lo descrito en 5.2 a 5.26 está relacionado y complementa la ABNT NBR 15606-2:2007, Sección 7.

La definición completa de los módulos NCL 3.0, utilizando esquemas XML, se presenta en la ABNT NBR 15606-2.

Para cada perfil NCL (perfil avanzado EDTV y perfil básico BDTV) se presenta una tabla indicando los elementos del módulo y sus atributos. Para un determinado perfil, los atributos y contenidos (elementos hijos) de los elementos se pueden definir en el propio módulo o en el perfil del lenguaje que agrupa los módulos. Por lo tanto, las Tablas 1 a 37 muestran los atributos y contenidos procedentes de los perfiles, además de los definidos en los propios módulos. Los atributos necesarios de cada elemento están subrayados. En las tablas se utilizan los siguientes símbolos: (?) opcional (cero o un acontecimiento), (|) o (*) cero o más acontecimientos, (+) un o más acontecimientos. El orden de los elementos hijos no se especifica en las tablas.

Lo descrito en 5.4 a 5.26 discute valores y directrices de operación para cada módulo.

5.2 Perfil EDTV NCL 3.0

Las Tablas 1 a 20 describen los elementos y atributos definidos por el perfil EDTV.

Tabela 1 — Elementos y atributos del módulo Structure extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
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 y atributos del módulo *Layout* extendido utilizados en el perfil EDTV

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

Tabela 3 — Elementos y atributos del módulo *Media* extendido utilizados en el perfil EDTV

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

Tabela 4 — Elementos y atributos del módulo *Context* extendido utilizados en el perfil EDTV

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

Tabela 5 — Elementos y atributos del módulo *MediaContentAnchor* extendido utilizados en el perfil EDTV

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

Tabela 6 — Elementos y atributos del módulo *CompositeNodeInterface* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
port	*id, component, interface*	vacío

Tabela 7 — Elementos y atributos del módulo *PropertyAnchor* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
property	*name, value*	vacío

Tabela 8 —Elementos y atributos del módulo *SwitchInterface* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
switchPort	*id*	mapping+
mapping	*component, interface*	vacío

Tabela 9 — Elementos y atributos del módulo *Descriptor* extendido utilizados en el perfil EDTV

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

Tabela 10 — Elementos y atributos del módulo *Connector* extendido utilizados en el perfil EDTV

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

Tabela 11 — Elementos y atributos del módulo *CausalConnectorFunctionality* extendido utilizados en el perfil EDTV

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

Tabela 12 — Elementos y atributos del módulo *ConnectorBase* extendido utilizados en el perfil EDTV

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

Tabela 13 — Elementos y atributos del módulo *TestRule* extendido utilizados en el perfil EDTV

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

Tabela 14 — Elementos y atributos del módulo *TestRuleUse* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
bindRule	*constituent, rule*	vacío

Tabela 15 — Elementos y atributos del módulo *ContentControl* extendido utilizados en el perfil EDTV

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

Tabela 16 — Elementos y atributos del módulo DescriptorControl extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
descriptorSwitch	*id*	(defaultDescriptor?, (bindRule | descriptor)*)
defaultDescriptor	*descriptor*	vacío

Tabela 17 — Elementos y atributos del módulo *Import* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
importBase	*alias, documentURI, region, baseId*	vacío
importedDocumentBase	*id*	(importNCL)+
importNCL	*alias, documentURI*	vacío

Tabela 18 —Elementos y atributos del módulo *TransitionBase* extendido utilizados en el perfil EDTV

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

Tabela 19 — Elementos y atributos del módulo *BasicTransition* extendido utilizados en el perfil EDTV

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

Tabela 20 — Elementos y atributos del módulo *Metainformation* extendido utilizados en el perfil EDTV

Elementos	Atributos	Contenido
meta	*name, content*	vacío
metadata	*empty*	RDF tree

5.3 Perfil BDTV NCL 3.0

Las Tablas 21 a 37 describen los elementos y atributos definidos por el perfil BDTV.

Tabela 21 — Elementos y atributos del módulo *Structure* extendido utilizados en el perfil BDTV

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

Tabela 22 — Elementos y atributos del módulo *Layout* extendido utilizados en el perfil BDTV

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

Tabela 23 — Elementos y atributos del módulo *Media* extendido utilizados en el perfil BDTV

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

Tabela 24 — Elementos y atributos del módulo *Context* extendido utilizados en el perfil BDTV

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

Tabela 25 — Elementos y atributos del módulo *MediaContentAnchor* extendido utilizados en el perfil BDTV

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

Tabela 26 — Elementos y atributos del módulo *CompositeNodeInterface* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
port	*id, component, interface*	vacío

Tabela 27 — Elementos y atributos del módulo *PropertyAnchor* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
property	*name, value*	vacío

Tabela 28 — Elementos y atributos del módulo *SwitchInterface* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
switchPort	*id*	mapping+
mapping	*component, interface*	vacío

Tabela 29 — Elementos y atributos del módulo *Descriptor* extendido utilizados en el perfil BDTV

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

Tabela 30 — Elementos y atributos del módulo *Connector* extendido utilizados en el perfil BDTV

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

Tabela 31 — Elementos y atributos del módulo *CausalConnectorFunctionality* extendido utilizados en el perfil BDTV

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

Tabela 32 — Elementos y atributos del módulo *ConnectorBase* extendido utilizados en el perfil BDTV

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

Tabela 33 — Elementos y atributos del módulo *TestRule* extendido utilizados en el perfil BDTV

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

Tabela 34 — Elementos y atributos del módulo *TestRuleUse* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
bindRule	*constituent, rule*	vacío

Tabela 35 — Elementos y atributos del módulo *ContentControl* extendido utilizados en el perfil BDTV

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

Tabela 36 — Elementos y atributos del módulo *DescriptorControl* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
descriptorSwitch	*id*	(defaultDescriptor?, (bindRule | descriptor)*)
defaultDescriptor	*descriptor*	vacío

Tabela 37 — Elementos y atributos del módulo de *Import* extendido utilizados en el perfil BDTV

Elementos	Atributos	Contenido
importBase	*alias, documentURI, region*	vacío
importedDocumentBase	*id*	(importNCL)+
importNCL	*alias, documentURI,*	vacío

5.4 Módulo Structure

5.4.1 Consideraciones generales

El atributo *xmlns* del elemento <ncl> declara un *namespace* XML, es decir, declara el grupo primario de construcciones XML que el documento utiliza. Se permiten tres valores para el atributo *xmlns*: http://www.ncl.org.br/NCL3.0/EDTVProfile, http://www.ncl.org.br/NCL3.0/BDTVProfile y http://www.ncl.org.br/NCL3.0/CausalConnectorProfile, para los perfiles TVD avanzado, TVD básico y conector causal, respectivamente. Un formateador NCL debe obligatoriamente saber que el schemaLocation para esos namespaces son, por *default*, respectivamente:

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

5.4.2 Valores *default*

No hay ningún valor *default*.

5.4.3 Tratamiento de excepciones

Se recomienda que, los documentos con atributo *xmlns* diferente de los tres valores mencionados anteriormente, se ignoren en una implementación, de acuerdo con las ABNT NBR 15606-2 y ABNT NBR 15606-5.

Se recomienda que los documentos con atributos id, cuyos valores no son secuencias de caracteres compatibles con la_producción NCName [Namespaces in XML: 1999], se ignoren en una implementación, de acuerdo con las ABNT NBR 15606-2 y ABNT NBR 15606-5.

5.5 Módulo Layout

5.5.1 Consideraciones generales

Cada elemento <regionBase> está asociado a una clase de dispositivos donde la presentación acontece. Con la intención de identificar la asociación, el elemento <regionBase> define el atributo *device*, que puede tener valores: “systemScreen (i)” o “systemAudio(i)”, donde *i* es un número entero mayor o igual a 1.

El soporte a múltiples dispositivos se define en las directrices establecidas en la Nested Context Language 3.0, Part 12.

En el SBTVD, el systemScreen (1) y systemAudio (1) son reservados para las clases pasivas y el systemScreen (2) y systemAudio (2) son reservados para las clases activas.

5.5.2 Valores default

El elemento <regionBase> que define una clase pasiva también puede tener un atributo *region*. Si el atributo no se especifica, la exhibición acontece apenas en los dispositivos de la clase pasiva.

Cuando una región anidada no especifica un valor de posicionamiento o de tamaño, y ese valor no pudiera ser calculado a partir de otros atributos, se debe obligatoriamente asumir el mismo valor absoluto del atributo de la geometría padre correspondiente. Concretamente, cuando una región de primer nivel no especifica ningún valor de posicionamiento o de tamaño, se adopta el valor de toda el área de presentación del dispositivo.

Cuando no se especifica, el atributo *zIndex* se define como cero.

5.5.3 Tratamiento de excepciones

Cuando el usuario especifica los valores para *top*, *bottom* y *height* para la misma <region>, puede haber inconsistencias espaciales. En ese caso, los valores *top* y *height* deben obligatoriamente tener prioridad sobre el valor *bottom.* Igualmente, cuando el usuario especifica valores inconsistentes para los atributos *left*, *right* y *width* del elemento <region>, los valores *left* y *width* se utilizarán para calcular un nuevo valor *right*.

Las regiones hijas deben obligatoriamente estar totalmente contenidas en el área establecida por sus regiones padre. Cuando parte de la región hija se encuentra fuera de su región padre, la región hija debe ser ignorada (considerada como si no fuese especificada).

5.6 Módulo Media

5.6.1 Consideraciones generales

El módulo Media define tipos básicos de objetos de media. Para ello, este módulo define el elemento <media>. Cada objeto de media tiene dos atributos principales, además de su atributo *id*: *src*, que define la URI del contenido del objeto, y *type*, que define el tipo de objeto.

Obsérvese que los objetos de media con el mismo valor para *src* y con el esquema URI diferente de “ncl-mirror” tienen el mismo contenido que se presentará, sin embargo, son dos instancias de objetos de media. Como consecuencia, el contenido de cada uno de los objetos puede tener su presentación iniciada en momentos diferentes, dependiendo de cuándo los objetos de media se inicien, y sus exhibiciones son totalmente independientes. Por otro lado, si el esquema de URI es igual al “ncl-mirror”, el objeto de media cuyo atributo *src* define ese esquema y el objeto de media referenciado por el esquema deben obligatoriamente tener el mismo contenido en presentación y en el mismo instante del tiempo, si los dos objetos de media se estuvieran presentando, independientemente de cuando se iniciaron. Por ser objetos de media diferentes, sus propiedades pueden tener valores diferentes, por ejemplo, las que especifican la localización de la presentación. Se debe destacar que la relación de espejado es reflexiva, simétrica y transitiva.

El sistema “ncl-mirror” no puede referirse a un elemento de <media> de los tipos application/x-ncl-NCL, application/x-ncl-NCLet y application/x-ncl-NCLua.

5.6.2 Objeto de media continua en flujos elementales TS

Si más de un objeto de media que tienen el atributo *src* con un mismo valor referente a un contenido transportado en el flujo de transporte (TS) fuera iniciado, se debe obligatoriamente iniciar más de una presentación. Además, como de costumbre, esos objetos pueden tener diferentes regiones de presentación, que pueden ser redimensionadas por una aplicación NCL. Sin embargo, se debe observar que el número de objetos de media que se pueden presentar en el plan de vídeo SBTVD depende de la implementación del receptor. De los exhibidores H.264 implementados en el *hardware*, se exige apenas una presentación de contenido de cada vez. El permiso de más de un objeto de media de vídeo en el plan de vídeo es opcional. Si el número de objetos de media de un determinado tipo excede el número máximo permitido, la exhibición de los objetos de media excedentes deben obligatoriamente ignorarse.

Con relación al plan de vídeo del dispositivo base de exhibición, si, y exclusivamente si, no hubiera ningún objeto de media siendo presentado en ese plan, refiriéndose (por medio de su atributo *src*) a un flujo elemental de vídeo de un servicio sintonizado (no importando en qué aplicación de la base privada que representa ese canal de TV), se debe obligatoriamente presentar un vídeo ES en pantalla completa, que no está en el momento siendo referenciado por ningún objeto de media en exhibición. Ese vídeo ES es el referenciado por el último objeto de media que tuvo su presentación interrumpida en el plan de vídeo o, si no, el vídeo principal ES del flujo de transporte sintonizado, según lo definido en la ABNT NBR 15604.

Si, y solamente si, no hubiera ningún objeto de media de audio, siendo presentado en el dispositivo base de exhibición, refiriéndose (por medio de su atributo *src*) un flujo de audio de un servicio sintonizado (objeto éste en cualquier aplicación de la base privada que representa un canal de TV), un audio ES, que no está en el momento siendo referenciado por ningún objeto de media en exhibición, debe obligatoriamente ser presentado. Ese audio ES es el referenciado por el último objeto de media que tuvo su presentación interrumpida o, si no, el audio principal ES del flujo de transporte sintonizado, según lo definido en la ABNT NBR 15604.

5.6.3 Tipos especiales de objetos NCL

5.6.3.1 Consideraciones generales

Cinco tipos especiales de objetos de media NCL son definidos: application/x-ginga-NCL (o application/x-ncl-NCL); application/x-ginga-NCLua (o application/x-ncl-NCLua); application/x-ginga-NCLet (o application/x-ncl-NCLet), que es opcional en el Ginga para receptores portátiles; application/x-ginga-settings (o application/x-ncl-settings); y application/x-ginga-time (o application/x-ncl-time).

Cuatro objetos importantes son los que permiten códigos imperativos o hipermedia declarativos integrados en un documento NCL: text/html; application/x-ginga-NCL (o application/x-ncl-NCL); application/x-ginga-NCLua (o application/x-ncl-NCLua); application/x-ginga-NCLet (o application/x-ncl-NCLet), que es opcional en el Ginga para receptores portátiles. Como tal, se discuten en 5.6.3.2 a 5.6.3.4. La presentación de los objetos de media se presenta en la Sección 6.

5.6.3.2 Objetos hipermedia declarativos integrados en presentaciones NCL

5.6.3.2.1 Consideraciones generales

Objetos hipermedia declarativos (con código NCL o codificados con otro lenguaje declarativo) se pueden insertar en documentos NCL. La manera de adicionar un objeto hipermedia declarativo a un documento NCL es definir un elemento <media>, cuyo contenido (localizado por medio del atributo *src*) es el código declarativo que se ejecutará. Ambos perfiles EDTV y BDTV de la NCL 3.0 permiten que el elemento <media> de los tipos text/html y application/x-Ginga-NCL (o application/x-ncl-NCL) se anide en un documento NCL.

5.6.3.2.2 Objetos NCL integrados en aplicaciones NCL

Los objetos NCL anidados en aplicaciones NCL siguen las directrices establecidas en Nested Context Language 3.0, Part 11.

5.6.3.2.3 Objetos XHTML integrados en aplicaciones NCL

Cualquier implementación de objeto de media basado en XHTML, de acuerdo con la ABNT NBR 15606, debe obligatoriamente ofrecer soporte a todas las marcaciones (markups) XML y propiedades de hojas de estilo (stylesheets) comunes a la BML para servicios básicos ("fixed terminal profile"), ACAP-X e DVB-HTML, según lo definido en la ITU-T J.201. Las funcionalidades comunes de los objetos ECMAScript nativos y API DOM son opcionales.

Aunque un *browser* basado en XHTML deba ser soportado, la utilización de elementos XHTML para definir relaciones (incluso enlaces XHTML y eventos de flujo) se desaconseja en la autoría de documentos NCL.

Los objetos basados en HTML anidados en aplicaciones NCL siguen las directrices establecidas en la Nested Context Language 3.0, Part 11.

5.6.3.3 Objetos codificados en Lua integrados en aplicaciones NCL

Los objetos NCLua integrados en aplicaciones NCL siguen las directrices establecidas en la Nested Context Language 3.0, Part 10.

5.6.3.4 Objetos codificados en Java integrados en aplicaciones NCL

Los objetos NCLet integrados en aplicaciones NCL siguen las directrices establecidas en la Nested Context Language 3.0, Part 10.

Cuando un Xlet es referido como un archivo ".class" en un elemento <media>, un *classpath* alternativo se puede definir por medio de un elemento <property> del objeto, con el atributo *name* con el valor "x-classpath" y el atributo *value* teniendo el camino relativo para el *classpath*.

NOTA 1 Los objetos NCLet son opcionales en receptores *one-seg*.

NOTA 2 Las aplicaciones Xlet se pueden encapsular en archivos JAR, si viniesen por el canal de interactividad. En ese caso, un archivo *manifest* tiene que estar presente (camino "META-INF/MANIFEST.MF") con el parámetro "Main-Class" refiriéndose al Xlet principal.

5.6.3.5 Objeto del tipo application/x-ncl-settings

El tipo application/x-ginga-settings (o application/x-ncl-settings) se aplica a un elemento <media> especial, cuyas propiedades son variables globales definidas por el autor del documento o variables de ambiente reservadas, que se pueden manipular por medio del procesamiento del documento NCL.

La propiedad *user.location* depende de qué país adopte el Ginga. Ésta debe obligatoriamente ser el código del país concatenado con el código postal del país. La especificación del código del país debe obligatoriamente seguir el formato ISO 3166-1 alfa 3. En el caso de una implementación para Brasil, el número del código postal del país (CEP) debe obligatoriamente ser especificado sin caracteres de guión, subrayado (_) o barra (/) y seguir la Guía Telefónica Brasileña de la Empresa Brasileña de Correos y Telégrafos.

Una nueva propiedad reservada es adicionada al grupo *system* del objeto de media *settings* en receptores portátiles: la propiedad *system.screenOrientation*. Esta propiedad recibe el valor "portrait" o "landscape", definiendo la orientación de la pantalla.

5.6.4 Valores *default*

El atributo *type* es opcional (excepto para los elementos <media> sin atributo *src* definido) y se usa para orientar la elección del exhibidor (herramienta de presentación) por el formateador. Cuando el atributo *type* no se especifica, el formateador debe obligatoriamente utilizar la especificación de extensión de contenido en el atributo *src* para elegir el exhibidor.

Para objetos de media con el atributo *src,* cuyo valor identifica el esquema “sbtvd-ts”, la parte específica del esquema, más concretamente, el “program_number.component_tag”, se puede sustituir por las palabras reservadas dadas en la Tabla 38.

Tabela 38 — Palabras reservadas para la parte específica del esquema sbtvd-ts

Palabra reservada	Descripción
video	Correspondiente al vídeo ES principal que se está presentando en el plan de vídeo, según lo definido por la ABNT NBR 15604
audio	Correspondiente al audio ES principal, según lo definido por la ABNT NBR 15604
text	Correspondiente al texto ES principal, según lo definido por la ABNT NBR 15604
video(i)	Correspondiente al i-ésimo menor vídeo ES *component_tag* listado en la PMT de los servicios sintonizados
audio(i)	Correspondiente al i-ésimo menor audio ES *component_tag* listado en la PMT de los servicios sintonizados
text(i)	Correspondiente al i-ésimo menor texto ES *component_tag* listado en la PMT de los servicios sintonizados

En el objeto de media del tipo application/x-ginga-settings (el application/x-ncl-settings), las propiedades system.ncl.version, system.GingaNCL.version y system.GingaJ.version, reciben como default los valores 3.0, 1.0 y 1.0, respectivamente.

En la definición del teclado virtual en la propiedad channel.keyboardBounds, se permite el uso de un teclado virtual de tamaño prefijado por el fabricante del receptor.

5.6.5 Tratamiento de excepciones

Las referencias a recursos de *streaming* de vídeo o de audio no pueden causar sintonización. Las referencias que implican en sintonización para acceso a un recurso deben obligatoriamente ser tratadas como si los recursos no estuvieran disponibles.

Se recomienda que toda acción sobre un elemento <media> representando un recurso indisponible sea ignorada por el formateador NCL. Se recomienda que cualquier *condition* o *assessment* basado en un elemento <media> representando un recurso indisponible se considere como falsa.

Si el número de objetos de media de un determinado tipo excede el número máximo permitido para aquel tipo, en un dispositivo de exhibición específico, se recomienda que se ignore la exhibición de los objetos de media excedentes.

Si el archivo de origen asociado a un nudo de media es actualizado en un carrusel de objetos, y si el nudo de media no se está presentando cuando el archivo actualizado llegue, el nuevo archivo sustituye al anterior. Por lo tanto, cuando el nudo de media se inicia después que la actualización se concluya, éste carga los contenidos del archivo actualizado. Si el nudo de media es iniciado durante la actualización, carga los contenidos de la última versión del archivo completo recibido. Esa versión se mantiene durante todo el proceso de actualización, siendo sustituida solamente al final. Si el archivo asociado a un nudo de media es actualizado en un carrusel de objetos, y si el nudo de media ya se está presentando cuando el archivo actualizado llegue, la presentación del nudo de media no es afectada. El archivo actualizado solamente es utilizado en presentaciones futuras.

Si un elemento <media> tiene un atributo *src* que especifica el esquema “ncl-mirror” y ese esquema hace referencia a un elemento <media> de application/x-ncl-NCL, o application/x-ncl-NCLet, o application/x-ncl-NCLua, se recomienda que se ignore el elemento <media>.

5.7 Módulo Context

No hay comentarios que complementen o aclaren la ABNT NBR 15606-2.

5.8 Módulo MediaContentAnchor

5.8.1 Consideraciones generales

El módulo MediaContentAnchor define el elemento <area>.

Para objetos de media del tipo texto, los atributos *beginText* y *beginPosition* especifican el inicio del texto áncora. El final del texto áncora se puede especificar utilizando los atributos *endTex* y *endPosition*, que también definen una secuencia y el acontecimiento de la secuencia en el texto, respectivamente.

EJEMPLO Suponiendo el contenido del texto a continuación: AAA AA AA AAA y los atributos *beginText*=”AA”.

Para *beginPosition* = 1, las anclas comienzan en la secuencia subrayada y en negrita: **AA**A AA AA AAA

Para *beginPosition* = 2, las anclas comienzan en la secuencia subrayada y en negrita: A**AA** AA AA AAA

Para *beginPosition* = 3, las anclas comienzan en la secuencia subrayada y en negrita: AAA **AA** AA AAA

Para elementos <area> con atributos *first* y *last*, cuando sus valores se especifiquen en NPT, éstos se refieren a la base temporal especificada por el atributo *contentId* del mismo elemento <media> que contiene el elemento <area>.

Para objetos de media del tipo application/x-ginga-NCL (o application/x-ncl-NCL), los valores de los atributos *clip* y *label* siguen las directrices establecidas en la Nested Context Language 3.0, Part 11.

Para objetos de media del tipo application/x-ginga-NCLua (o application/x-ncl-NCLua), los valores del atributo *label* siguen las directrices establecidas en la Nested Context Language 3.0, Part 10.

5.8.2 Valores *default*

En la NCL, cada nudo (elementos de <media>, <body> o <context>) tiene un ancla con una región que representa todo su contenido. Este ancla se denomina *whole content anchor* y es declarada por *default* en la NCL. Con excepción del elemento de media con código imperativo (por ejemplo, <media type=“application/x-ginga-NCLua” ...>), cada vez que un componente NCL es referenciado sin especificar una de sus anclas, se debe obligatoriamente asumir la *whole content anchor.*

En anclas de contenido temporal, si el atributo *begin* de un elemento <area> es definido, mas el atributo *end* no es especificado, el final de toda la presentación del contenido de media debe obligatoriamente ser considerado como el final del ancla. Por otro lado, si el atributo *end* es definido, pero sin una definición explícita de *begin*, el inicio de toda la presentación del contenido de media debe obligatoriamente ser considerado como el inicio del ancla. Se espera un funcionamiento semejante de los atributos *first* y *last*. En el caso de un elemento <media> del tipo application/x-ginga-time, los atributos *begin* y *end* deben obligatoriamente ser definidos y deben obligatoriamente asumir un valor absoluto del Tiempo Universal Coordenado (UTC).

En anclas de contenido textual, si el final de la región del ancla no se define, se debe obligatoriamente asumir el final del contenido del texto.

En anclas de contenido textual, si el inicio de la región del ancla no se define, se debe obligatoriamente asumir el inicio del contenido del texto.

5.8.3 Tratamiento de excepciones

No hay comentarios que complementen o aclaren la ABNT NBR 15606-2.

5.9 Módulo PropertyAnchor

5.9.1 Consideraciones generales

El elemento <property> define el atributo *name*, que indica el nombre de la propiedad o grupo de propiedades.

Los elementos <body>, <context> y <media> pueden tener varias propiedades (ver ABNT NBR 15606-2) que no son explícitamente declaradas. Sin embargo, cuando una propiedad es utilizada en una relación, ésta debe obligatoriamente ser explícitamente declarada en un elemento <property> (interface). Por lo tanto, cada propiedad tiene un atributo booleano denominado *externable,* que se define como “false” por *default* y como “true” cuando la propiedad es explícitamente declarada.

Las propiedades que tienen como valor strings con nombres de colores reservados (“white”, “black”, “silver”, “gray”, red”, “maroon”, fuchsia”, “purple”, “lime”, “green”, “yellow”, “olive”, “blue”, “navy”, “aqua”, o “teal”) siguen el estándar de color CSS1, según lo definido en la Tabla 39:

Tabela 39 — Definición de las palabras reservadas para colores

	Nombre	Hexadecimal	R	G	B	Matiz	Saturación	Luz	Saturación	Valor
	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 %

El valor de la propiedad *contentId* (asociada a un objeto de media continua cuyo contenido se define haciéndose referencia a un flujo elemental) es definido por el *middleware*. En principio, tiene que tener obligatoriamente el valor "null" y debe obligatoriamente asumir el valor del identificador transportado en el descriptor de referencia NPT (en un campo identificado con el mismo nombre: contentId), así que el objeto de media continúe asociado es iniciado.

El valor de la propiedad *standby* también es definido por el *middleware*. Éste tiene que tener obligatoriamente "true" como valor, mientras un objeto de media continúa ya iniciado y que hace referencia a un flujo elemental esté temporalmente interrumpido por otro contenido intercalado en el mismo flujo elemental.

NOTA La propiedad *standby* recibe el valor "true" automáticamente por el *middleware*, cuando el valor del identificador transportado en el descriptor de referencia NPT (en un campo identificado con el mismo nombre: contentId) señalado como no pausado sea diferente del valor de la propiedad *contentId*.

La propiedad *standby* puede ser usada para pausar una aplicación, cuando el contenido del objeto de media continua que hace referencia a un flujo elemental transportando el vídeo principal de un programa de TV sea temporalmente interrumpido por otro contenido intercalado, por ejemplo, una propaganda (comercial de TV). La misma propiedad puede ser utilizada para retomar la aplicación.

Cuando la propiedad *visible,* asociada a un elemento <context> o <body>, es igual a "true", la propiedad *visible* de cada elemento hijo de la composición debe obligatoriamente ser considerada para definir la forma como cada elemento hijo se exhibirá.

Cuando la propiedad *visible,* asociada a un elemento <context> o <body> es igual a "false", todos los elementos hijos de la composición son exhibidos de forma oculta. Específicamente, cuando un documento tiene su elemento <body> con la propiedad *visible* configurada como "false" y su evento de presentación está en el estado *paused*, se dice que el documento está en espera (*stand-by).* Cuando hay una única aplicación en ejecución y esa aplicación está en espera, el vídeo principal del servicio debe obligatoriamente ser dimensionado para el 100 % de la pantalla y el audio principal debe obligatoriamente ser configurado para el 100 % de su volumen.

Se debe observar que un objeto con propiedad *visible* igual a "false", es decir, un objeto en exhibición pero oculto, no puede transicionar las máquinas de estados de eventos de selección definidas por sus anclas de contenido para el estado "occurring", mientras el valor de la propiedad *visible* siga como "false".

5.9.2 Valores *default*

El atributo *value* de un elemento <property> es opcional y define un valor inicial para la propiedad declarada en *name*. Cuando el valor no es especificado, la propiedad asume como valor inicial el definido en los atributos homónimos del descriptor o región asociados al nudo (objeto) donde la propiedad fue definida o, si no, un valor *default*. Cuando el atributo *value* es especificado, tiene prioridad sobre el valor definido en los atributos homónimos del descriptor o región asociados al nudo.

NOTA Todas las propiedades (y sus valores iniciales) de un objeto NCL se pueden definir usando solamente elementos <property>. Los elementos <descriptor>, <descriptorParam> y <region> son apenas una opción más (opción de reutilización) para definir valores iniciales para las propiedades.

Si las propiedades *left, right, top, bottom, width* o *height* no están definidas y no pueden ser calculadas a partir de los valores de propiedad definidos en <property>, <descriptor> y sus elementos hijos, o en los elementos <region>, deben obligatoriamente asumir el valor "0".

5.9.3 Tratamiento de excepciones

Cuando el formateador trata un cambio en un grupo de propiedades, debe comprobar la consistencia del proceso solamente al final.

Cuando el usuario especifica las informaciones de *top*, *bottom* y *height* para el mismo elemento <media>, pueden acontecer inconsistencias espaciales. En ese caso, los valores de *top* y *height* deben obligatoriamente tener prioridad sobre el valor de *bottom*. De la misma forma, cuando el usuario especifica valores inconsistentes para las propiedades de *left*, *right* y *width*, los valores de *left* y *width* son utilizados para calcular un nuevo valor de *right*.

Cuando las propiedades de *left, right, top, bottom, width* o *height* excedan la dimensión del dispositivo de exhibición, solamente será exhibida una parte del contenido dentro de la dimensión del dispositivo.

Si dos o más elementos <property> con el mismo atributo *name* se definen como elementos hijos del mismo elemento <media>, se debe obligatoriamente considerar solamente el último *value* definido.

5.10 Módulo CompositeNodeInterface

El módulo CompositeNodeInterface define el elemento <port>, que especifica la puerta de un nudo de composición, con su respectivo mapeado para un interfaz (atributo *interface*) de uno, y solamente uno, de sus componentes hijos (especificado por el atributo *component*).

5.11 Módulo SwitchInterface

5.11.1 Consideraciones generales

El módulo SwitchInterface permite la creación de interfaces del elemento <switch>, que se pueden mapear para un conjunto de interfaces alternativas de los objetos internos del switch, permitiendo que un *link* se ancle en la interfaz elegida, cuando el <switch> es procesado. Este módulo introduce el elemento <switchPort>, que contiene un conjunto de elementos *mapping*. Un elemento *mapping* define un camino de la <switchPort> para una interfaz (atributo *interface*) de uno de los componentes del switch (especificado por su atributo *component*).

NOTA Un elemento <switchPort> puede definir un mapeado para un subconjunto de los componentes del switch. Cuando un eslabón se refiere a una <switchPort> y ninguna de las reglas, asociadas a los componentes definidos por sus elementos hijo <mapping>, se define como verdadera, es elegido el elemento <defaultComponent>. Si el elemento <defaultComponent> no se define, ningún componente es seleccionado para la presentación.

5.11.2 Valores *default*

La referencia a un componente interno a un switch debe obligatoriamente hacerse por medio de un elemento <switchPort> o, por *default*, al elemento <switch> sin especificar ningún <switchPort>. En ese último caso, se considera como si la referencia se hiciese a un <switchPort> *default* que contiene elementos <mapping> para cada objeto hijo del switch y refiriéndose a su *whole content anchor*.

5.12 Módulo Descriptor

5.12.1 Consideraciones generales

Un elemento <descriptor> puede tener elementos hijos <descriptorParam>, que son utilizados para establecer parámetros de control de la presentación del objeto asociado al elemento descriptor.

Un atributo *descriptor* puede ser asociado a cualquier objeto de media por medio de los propios elementos <media> o por medio de los puntos terminales (elementos <bind>) de los eslabones (elementos <link>).

El parámetro *plan* de un elemento <descriptorParam> define en qué plano de una pantalla estructurada se posiciona el objeto. El valor de ese atributo puede ser: “background”, “video” o “graphic”, siguiendo la definición de plano de la ABNT NBR 15606-1.

Los atributos *reusePlayer* y *playerLife* ofrecen soporte adicional para la gestión del exhibidor de los objetos de media y se pueden usar o ignorar por un determinado implementador del *middleware*. El atributo *playerLife* especifica lo que pasará con una instancia del exhibidor al final de una presentación del objeto de media. Mantener una instancia del exhibidor exige espacio en la memoria, sin embargo, disminuye el tiempo de carga del exhibidor y, como consecuencia, la probabilidad de desparejados en la sincronización. El atributo *reusePlayer* permite usar la misma instancia del exhibidor en la presentación de más de un objeto de media, incluyendo el uso de un exhibidor mantenido en el espacio de memoria por el atributo *playerLife*.

5.12.2 Valores *default*

La Tabla 40 resume algunos valores *default* para parámetros-de-descriptor/nombres-de-propiedades reservados.

Tabela 40 — Algunos parámetros/propiedades reservados y sus valores *default*

Nombre del parámetro/propiedad	*Default*
top, left, bottom, right, width, height	Si cualquier valor de esas propiedades no se define y no se puede inferir de las reglas definidas por la especificación NCL, debe obligatoriamente asumir el valor “0”.
location	Ver primera línea de esta tabla
size	Ver primera línea de esta tabla
bounds	Ver primera línea de esta tabla
plan	“video”, para objeto de media con el atributo *src* refiriendo un PES de un flujo TS, ”graphics”, para todos los otros casos.
baseDeviceRegion	
deviceClass	0
explicitDur	Para medias continuas debe asumir el valor de la duración de la presentación natural del contenido, caso contrario debe asumir el 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;	El valor definido por *default.focusBorderColor*
selBorderColor	El valor definido por *default.selBorderColor*
focusBorderWidth	El valor definido por *default.focusBorderWidth*
focusBorderTransparency	El valor definido por *default.focusTransparency*
focusSrc, focusSelSrc	nill
freeze	false
transIn, transOut	string vacía

5.12.3 Tratamiento de excepciones

Si varios valores son especificados para una misma propiedad, el valor definido en un elemento <property> tiene prioridad sobre el definido en un elemento <descriptorParam>, que, a su vez, tiene prioridad sobre el valor definido en un atributo del elemento <descriptor> correspondiente (incluso el atributo *region*).

Se recomienda que las propiedades definidas utilizándose CSS que no son exigidas por la ABNT NBR 15606-2 sean ignoradas por el exhibidor de media correspondiente.

5.13 Módulo Linking

5.13.1 Valores *default*

El atributo *interface* de un elemento <bind> puede referirse a cualquier interfaz de un nudo, es decir, un ancla, una propiedad, una puerta (en el supuesto de un nudo de composición) o una switchPort (en el supuesto de un switch). El atributo es opcional y si no es especificado la *whole content anchor* se asume como la interfaz, excepto en el caso de objetos de media con código imperativo, según lo detallado en 6.4.1.

5.13.2 Tratamiento de excepciones

Se recomienda ignorar un elemento <link> si el atributo *xconnector* se refiere a un conector hipermedia inexistente.

Si un eslabón define un mismo parámetro usando los elementos <linkParam> y <bindParam>, la definición usando <bindParam> tiene preferencia.

5.14 Funcionalidad Connectors

5.14.1 Consideraciones generales

La funcionalidad Connectors define el módulo CausalConnectorFunctionality que agrupa un conjunto de módulos básicos de la Funcionalidad Connectors con el fin de facilitar la definición de un perfil de lenguaje. Ese es el caso de los perfiles EDTV y BDTV.

El elemento <compoundAction> de un conector tiene el atributo *operator* (“par” o “seq”) relacionando sus elementos hijos. Es importante mencionar que al utilizar el operador secuencial, las acciones deben obligatoriamente ser disparadas en el orden especificado. Sin embargo, una acción no tiene que esperar la conclusión de la anterior para ser iniciada.

5.14.2 Valores *default*

Si el valor *eventType* de un elemento <simpleCondition> es “selection”, el papel también puede definir a qué dispositivo de selección (por ejemplo, teclado o teclas del mando a distancia) se refiere, por medio de su atributo *key*. Si este atributo no es especificado, la selección por medio de un dispositivo apuntador (ratón, pantalla de toque, teclas de navegación, según 5.23, etc.) debe asumirse obligatoriamente.

NOTA Cuando un dispositivo de selección es presionado, más de una <simple condition> puede ser considerada como satisfecha si ese dispositivo de selección se define en el atributo *key* de las <simpleCondition> y las interfaces vinculadas por los elementos <link> refiriéndose a la <simpleCondition> (por medio de los atributos *role* de sus elementos <bind>) se estuvieran presentando.

En un elemento <simpleCondition> y en un elemento <simpleAction>, la cardinalidad del rol especifica el número mínimo (atributo *min*) y máximo (atributo *max*) de participantes que pueden desempeñar el papel (número de *binds*), cuando el <causalConnector> se use para crear un <link>. Si las cardinalidades mínimas y máximas no son informadas, se debe obligatoriamente asumir “1” como valor *default* para ambos parámetros.

Un atributo *qualifier* informa la relación lógica entre los actores de la misma condición simple. Si no se especifica, se debe obligatoriamente asumir el valor *default* “or”.

Un atributo *qualifier* informa la relación lógica entre los actores de la misma acción simple. Si no se especifica, se debe obligatoriamente asumir el valor *default* “par”.

Si el valor *eventType* de un elemento <attributeAssessment> es “selection”, la función también puede definir a qué dispositivo de selección (por ejemplo, teclado o teclas de mando a distancia) se refiere, por medio de su atributo *key*. Si este atributo no se especifica, la selección por medio de un dispositivo apuntador (ratón, pantalla de toque, etc) debe asumirse obligatoriamente.

Si el valor de *eventType* de un elemento <attributeAssessment> es “attribution”, el *attributeType* es opcional y tiene el valor “nodeProperty” como *default*.

5.14.3 Tratamiento de excepciones

El valor mínimo de la cardinalidad del papel es siempre un valor positivo finito, mayor que cero y menor o igual al valor máximo de la cardinalidad, en caso contrario, se recomienda que el *link* sea ignorado.

Si el valor de *eventType* es “attribution”, la <simpleAction> también debe obligatoriamente definir el valor que se atribuirá, por medio de su atributo *value*. Si *value* es especificado como “$anyName” (donde $ es un símbolo reservado y anyName es cualquier secuencia de caracteres, excepto nombres de papeles reservados), el valor atribuido debe obligatoriamente ser recuperado a partir de la propiedad asociada con *role*=“anyName”, definida por un elemento <bind> hijo del elemento <link> que hace referencia al conector. Si ese valor no pudiera ser recuperado, no se hace ninguna atribución.

Si el valor de *eventType* es “attribution”, y la <simpleAction> define el valor que se atribuirá como “$anyName”, el valor que se atribuirá debe obligatoriamente ser el valor de una propiedad (elemento <property>) de un componente de la misma composición donde la relación (elemento <link>) que se refiere al evento es definido, o de una propiedad de la composición, donde la relación se define, o de una propiedad de un elemento que puede ser alcanzada por medio de un elemento <port> de la composición, donde la relación se define, o incluso una propiedad de un elemento que puede ser alcanzada por medio de una puerta (elementos <port> o <switchPort>) de una composición anidada en la misma composición donde la relación se define. Caso contrario, se puede hacer ninguna atribución.

Un elemento <attributeAssessment> define un atributo *offset* cuyo valor puede adicionarse al valor de la variable referida por el atributo *attributeType* correspondiente. El valor *offset* debe obligatoriamente tener el mismo tipo y ser especificado con la misma unidad del valor al cual se adicionará, caso contrario, se recomienda que el valor *offset* sea ignorado.

El valor del atributo *comparator* del elemento <asessmentStatement> debe obligatoriamente asumir los valores: “eq”, “ne”, “gt”, “lt”, “gte”, o “lte”. Se recomienda que si cualquier otro valor fuera especificado para ese atributo, el elemento sea ignorado.

Los eventos de selección solamente se pueden definir sobre unidades de información de un objeto de media que se está presentando, es decir, sobre unidades de información cuyo evento de presentación asociado esté en el status “occurring”.

5.15 Módulo TestRule

5.15.1 Consideraciones generales

El módulo TestRule permite la definición de reglas que, cuando son cumplidas, seleccionan alternativas para la presentación del documento. Estas reglas pueden ser simples, definidas por el elemento <rule>, o compuestas, definidas por el elemento <compositeRule>.

Las comparaciones en reglas simples se hacen con base en la representación binaria del valor de la variable que se va a comparar y en la representación binaria del otro valor utilizado en la comparación.

Un elemento <rule> definido como elemento hijo de <compositeRule> puede tener su atributo *id* omitido.

5.15.2 Tratamiento de excepciones

El valor del atributo *comparator* del elemento <rule> debe obligatoriamente asumir los valores: “eq”, “ne”, “gt”, “lt”, “gte”, o “lte”. Se recomienda que si cualquier otro valor fuera especificado para este atributo, el elemento sea ignorado.

En reglas simples, el atributo *comparator* relaciona la variable a un valor. El tipo de la variable y del valor deben obligatoriamente ser iguales; caso contrario, se recomienda que la definición de la regla sea ignorada.

5.16 Módulo TestRuleUse

No hay comentarios que complementen o aclaren la ABNT NBR 15606-2.

5.17 Módulo ContentControl

5.17.1 Consideraciones generales

El módulo ContentControl especifica el elemento <switch>. Los formateadores NCL deben obligatoriamente postergar la evaluación del switch para el momento en el cual una relación (eslabón) anclada en el switch necesite ser evaluado. Las reglas de prueba utilizadas para elegir el componente del switch que se presentará deben obligatoriamente ser definidas por el módulo TestRule.

Durante la presentación del documento, a partir del momento en que un <switch> se evalúa en adelante, éste se considera solucionado hasta el final de su actual presentación, es decir, mientras su evento de presentación correspondiente esté en el estado “occurring” o “paused”. La alternativa elegida puede ser referenciada por medio de un elemento <switchPort> mapeado para una de sus interfaces.

Cuando un <context> se define como un hijo de un elemento <switch>, los elementos <link> recursivamente contenidos en <context> se deben ser considerados por un exhibidor NCL solamente si el <context> es seleccionado después de la evaluación del switch. En caso contrario, los elementos <link> deben obligatoriamente considerarse desactivados y no pueden interferir en la presentación del documento.

5.17.2 Tratamiento de excepciones

Si el elemento <defaultComponent> no es definido en un elemento <switch> y si ninguna de las reglas bindRule es evaluada como verdadera para un componente vinculado a un elemento <mapping>, hijo del <switchPort> a través del cual el elemento <switch> es referenciado, ningún componente es seleccionado para la presentación y el formateador NCL funciona como si el componente no existiera.

5.18 Módulo DescriptorControl

5.18.1 Consideraciones generales

El módulo DescriptorControl especifica el elemento <descriptorSwitch>. De manera análoga al elemento <switch>, la selección del <descriptorSwitch> se hace utilizando reglas de prueba definidas en el módulo TestRule. La evaluación del descriptorSwitch debe obligatoriamente ser postergada hasta que el objeto que hace referencia al descriptorSwitch tenga que prepararse para la presentación.

Durante la presentación del documento, a partir del momento en que el <descriptorSwitch> es evaluado por un elemento <media> específico, es considerado resuelto para aquél elemento <media> hasta el final de la presentación de ese elemento <media>, es decir, mientras cualquier evento de presentación asociado a ese elemento <media> esté en el estado “occurring” o “paused”.

5.18.2 Tratamiento de excepciones

Si el elemento <defaultDescriptor> no es definido en un elemento <descriptorSwitch> y si ninguna de las reglas bindRule es evaluada como verdadera, ningún descriptor puede ser seleccionado para presentación y el formateador NCL funciona como si el descriptor no existiera.

5.19 Módulo Timing

5.19.1 Consideraciones generales

Este módulo define el atributo *freeze* para especificar qué le pasa a un elemento <media> al final de su presentación. Cuando el *freeze* es especificado con un valor igual a “true”, el último mapa de imagen del objeto debe obligatoriamente ser congelado, hasta que su final sea determinado por un evento externo (por ejemplo, procedente de la evaluación de un <link>), o por medio de un valor *explicitDur* definido para aquel objeto.

El módulo Timing también define el atributo *explicitDur* especificando la duración de la presentación de un objeto representado por un elemento <media>. Obsérvese que el atributo *explicitDur* provee la duración de la presentación del objeto y no la duración de la presentación del contenido del objeto. Si el valor *explicitDur* es mayor que la duración de la presentación del contenido, lo que pasará al final de la presentación de ese contenido depende del atributo *freeze* mencionado anteriormente. Si el valor *explicitDur* es menor que la duración de la presentación del contenido, la presentación del contenido es interrumpida. Obsérvese que un exhibidor puede, eventualmente, hacer ajustes de tiempo flexibles en el contenido de la media con el fin de hacer la duración de la presentación del contenido lo más próximo posible del valor de *explicitDur*.

5.19.2 Valores *default*

Cuando no se especifica, el valor del atributo *freeze* debe obligatoriamente ser considerado como “false”.

Cuando no se especifica, el valor del atributo *explicitDur* debe obligatoriamente ser considerado igual a la duración normal de la presentación del contenido.

5.20 Módulo Import

5.20.1 Consideraciones generales

El elemento <importNCL> no incluye el documento NCL referenciado, solamente hace visible el documento referenciado para que tenga sus componentes reutilizados por el documento que definió el elemento <importNCL>. Se pueden definir nuevas relaciones entre nuevos nudos creados por medio de la reutilización, pero no es posible definir nuevas relaciones dentro de los nudos de contexto importados, ya que, cuando es reutilizado, el nudo no puede tener su contenido modificado, pudiendo solamente ser relacionado a otros nudos.

Cuando se importa un documento, su elemento <media> del tipo application/x-ginga-settings (o application/x-ncl-settings) no tiene ninguna influencia sobre el elemento del mismo tipo del documento importador, cuyas propiedades son las que tienen validez para el documento importador.

5.20.2 Tratamiento de excepciones

La operación <importBase> es transitiva, es decir, si la *baseA* importa la *baseB*, que importa la *baseC,* la *baseA* importa la *baseC*. Sin embargo, el *alias* definido para la *baseC* dentro de la *baseB* no puede ser considerado por la *baseA.*

La operación del elemento <importNCL> también tiene la propiedad transitiva, es decir, si el *documentA* importa el *documentB*, que importa el *documentC,* el *documentA* importa el *documentC*. Sin embargo, el *alias* definido para el *documentC* dentro del *documentB* no puede ser considerado por el *documentA*.

Por definición, no hay reajuste en la operación de importación.

5.21 Módulo EntityReuse

5.21.1 Consideraciones generales

El módulo EntityReuse permite la reutilización de un elemento NCL. Solamente <media>, <context>, <body> y <switch> se pueden reutilizar.

Cuando un elemento declara un atributo *refer*, todos los atributos y elementos hijos definidos por el elemento referenciado son heredados. Para los elementos <media>, nuevos elementos hijos <area> y <property> se pueden incrementar y un nuevo atributo, *instance*, también se puede definir. Todo elemento <media> se comporta igualmente con relación a la reutilización, incluyendo el elemento con tipo igual a application/x-ginga-settings (o application/x-ncl-settings).

El elemento referenciado y el elemento que se refiere a él deben obligatoriamente ser considerados el mismo en relación a la especificación de sus datos.

5.21.2 Tratamiento de excepciones

Un elemento que se refiere a otro elemento no puede ser reutilizado, es decir, su *id* no puede ser el valor de ningún atributo *refer*. Cuando un elemento tiene el atributo *refer* con un valor correspondiente a la *id* de un elemento que se refiere a otro, se recomienda que el elemento sea considerado como inexistente.

Si el nudo referenciado se define en un documento importado *D*, el valor del atributo *refer* debe tener el formato de "alias#id", donde “*alias*” es el valor del atributo *alias* asociado a la importación de *D*. En caso contrario, se recomienda que el elemento que contiene el atributo *refer* sea considerado como inexistente.

Un elemento <media> solamente se puede referir a otro elemento <media>; un elemento <switch> solamente se puede referir a otro elemento <switch>, un elemento <context> o <body> solamente se puede referir a otro elemento <context> o <body>. En todos los otros casos, se recomienda que el elemento que contiene el atributo *refer* se considere como inexistente.

Todos los atributos y elementos hijos definidos por un elemento que hace referencia a otro deben obligatoriamente ser ignorados por el formateador NCL, excepto el atributo *id*, que debe obligatoriamente ser definido. La otra única excepción es para los elementos <media>, donde nuevos elementos hijos <area> y <property> se pueden incrementar, y un nuevo atributo, *instance*, también se puede definir.

Si el nuevo elemento <property> incrementado tiene el mismo atributo *name* de un elemento <property> ya existente (definido en el elemento <media> reutilizado), se recomienda que la nueva <property> incrementada sea ignorada. De manera análoga, si el nuevo elemento <area> incrementado tiene el mismo atributo *id* de un elemento <area> ya existente (definido en el elemento <media> reutilizado), se recomienda que la nueva <area> incrementada sea ignorada.

Los elementos <body>, <context> o <switch> no pueden incluir más de un elemento del conjunto compuesto por el objeto referente y por los objetos referenciados correspondientes. Si éste fuera el caso, se recomienda que todos los objetos referenciados se consideren inexistentes.

5.22 Módulo ExtendedEntityReuse

5.22.1 Consideraciones generales

El módulo ExtendedEntityReuse define el atributo *instance*.

El elemento referenciado y el elemento que se refiere a él se consideran objetos independientes con relación a su presentación si el atributo *instance* recibiere el valor “new”.

El elemento referenciado y el elemento que se refiere a él se consideran el mismo objeto con relación a su presentación si el atributo *instance* recibiere el valor “instSame” o “gradSame”.

5.22.2 Valores *default*

El atributo *instance* tiene el valor “new” como su valor *default*.

5.23 Módulo KeyNavigation

5.23.1 Consideraciones generales

El módulo KeyNavigation provee las extensiones necesarias para describir las operaciones de movimiento de foco usando un dispositivo de control, como un mando a distancia.

El atributo *focusIndex* especifica un índice para el elemento <media> en el cual el foco se puede aplicar. El *focusIndex* se puede definir en un elemento <property> o en un elemento <descriptor>.

Cuando un elemento en foco se selecciona presionando la tecla de activación de la navegación “(select, enter etc.)”, si hay un elemento <simpleCondition> con el atributo *role*=“onSelection” sin estar especificado el atributo *key*, esa condición es considerada como satisfecha si el elemento en foco fuera el especificado en el atributo *component* del elemento <simpleCondition>. Véase que las teclas de navegación operan de manera similar a un dispositivo apuntador (como ratón etc.).

Cuando un elemento en foco es seleccionado presionando la tecla de activación “(select, enter etc.)”, el control de foco debe obligatoriamente ser pasado para el renderizador (exhibidor) del elemento <media>. El exhibidor puede entonces seguir sus propias reglas de navegación. El control del foco es retornado para el formateador NCL cuando se presiona la tecla “back”. En ese caso, el foco va para el elemento identificado por la propiedad *service.currentFocus* del nudo de *settings* (elemento <media> del tipo application/x-ginga-settings). En un ambiente con diversos dispositivos, las reglas jerárquicas para el control de la tecla de entrada siguen las directrices establecidas en la Nested Context Language 3.0, Part 12.

El control del foco también puede ser transmitido configurando la propiedad service.currentKeyMaster del nudo *settings* (elemento <media> del tipo application/x-ginga-settings). Eso se puede hacer por medio de una acción de una relación o por medio de un comando de edición NCL ejecutado por un nudo de código imperativo (objeto NCLua, por ejemplo). El exhibidor de un nudo que tenga el control del foco no puede alterar directamente la propiedad service.currentKeyMaster.

5.23.2 Valores *default*

Cuando la propiedad *focusIndex* no se define, se considera como si ningún foco se pudiera establecer en el objeto correspondiente.

Cuando los atributos *focusBorderColor, focusBorderWidth, focusBorderTransparency,* o *selBorderColor* no están definidos, se consideran los valores *default*. Estos valores son especificados en las propiedades del elemento <media> del tipo application/x-ginga-settings (o application/ x-NCL-settings): *default.focusBorderColor, default.focusBorderWidth, default.focusTransparency*, *default.selBorderColor*.

5.23.3 Tratamiento de excepciones

En un determinado momento de la presentación, si el foco aún no se hubiera definido, o estuviera perdido, el foco es inicialmente aplicado al elemento que se esté presentando que tenga el menor valor para index.

Los valores del atributo *focusIndex* deben obligatoriamente ser exclusivos en un documento NCL. Caso contrario, se recomienda que los atributos repetidos sean ignorados, si, en determinado momento, hubiera más de un elemento de <media> para obtener el foco.

Cuando un elemento de <media> hace referencia a otro elemento de <media> (usando el atributo *refer*), el *focusIndex*, asociado al elemento referenciado de <media>, se debe ignorar obligatoriamente.

Cuando el foco es aplicado a un elemento con la propiedad visible configurada como falsa, o a un elemento que no se está presentando, el foco actual no se mueve.

Cuando el foco es aplicado a un elemento con la propiedad visible configurada como falsa, cualquier selección en el elemento debe ignorarse obligatoriamente.

5.24 Módulo Animation

5.24.1 Consideraciones generales

Toda vez que una animación NCL puede exigir muchos recursos computacionales, ésta es soportada solamente en el perfil EDTV, y sólo las propiedades que definen valores numéricos y los colores pueden ser animados. Un formateador NCL siguiendo el perfil BDTV puede ignorar los atributos de animación.

5.24.2 Valores *default*

Al definir un nuevo valor para una propiedad, el cambio es inmediato, por *default* (*duration* = ″0 ″).

Al definir un nuevo valor para una propiedad, la alteración del valor antiguo para el nuevo, si no se especifica lo contrario, es lineal por *default* (*by* = ″indefinite″).

5.25 Módulo Transition

5.25.1 Consideraciones generales

Las transiciones son clasificadas de acuerdo con una taxonomía de dos niveles: tipos y subtipos. El atributo *type* se exige y utiliza para especificar la transición general. El atributo *subtype* provee el control, específico por transición.

Solamente el subtipo *default* para los cinco tipos de transiciones necesarias, listadas en la Tabla 41, debe ser soportado obligatoriamente, los otros, definidos en *SMIL* 2.1, son opcionales.

Tabela 41 — Tipos y subtipos de transición necesarios

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

El módulo de transición también define los atributos que se usarán en los elementos <descriptor> para utilizar los templates de transición definidos por los elementos <transition>: atributos *transIn* y *transOut*.

Todas las transiciones definidas en el módulo de transición aceptan cuatro atributos adicionales procedentes de *SMIL 2.1*, que se pueden utilizar para controlar el aspecto visual de las transiciones: *horRepeat; vertRepeat; borderWidth;* y *borderColor*.

5.25.2 Valores *default*

El atributo *subtype* es opcional. Si ese atributo no se especifica, la transición asume el subtipo *default* para el tipo especificado de transición, según lo presentado en la Tabla 41.

El valor *default* para el atributo *dur* es de 1 s.

El atributo *startProgress* especifica la progresividad en la cual comienza la ejecución de la transición. El valor *default* es 0.0.

El atributo *endProgress* especifica la progresividad en la cual termina la ejecución de la transición. El valor *default* es 1.0.

El atributo *direction* especifica la dirección en que la transición se ejecuta. El valor *default* es “forward”.

El valor *default* para el atributo *fadeColor* es “black”.

El valor *default* para los atributos *transIn* y *transOut* es una secuencia de caracteres (string) vacía, que indica que ninguna transición es realizada.

El atributo *horRepeat* especifica la cantidad de veces, que el estándar de transición se ejecuta en la línea del eje horizontal. El valor *default* es 1.

El atributo *vertRepeat* especifica la cantidad de veces, que el estándar de transición se ejecuta en la línea del eje vertical. El valor *default* es 1.

El atributo *borderWidth* especifica la anchura de un borde en una barra de barredura. El valor *default* es 0.

El atributo *borderColor* especifica el contenido de un borde en una barra de barredura. El valor *default* para ese atributo es el color "black".

5.25.3 Tratamiento de excepciones

Si el tipo de transición especificada no es soportado por el formateador NCL, se recomienda que la transición sea ignorada.

El atributo *subtype*, si se especifica, debe obligatoriamente ser uno de los subtipos apropiados para el tipo especificado, en caso contrario, se recomienda que la transición sea ignorada.

El valor del atributo *endProgress* es un número real en el intervalo [0.0 – 1.0] y debe obligatoriamente ser igual o mayor que el valor del atributo *startProgress.* Si el valor *endProgress* es especificado como inferior al valor de *startProgress*, se recomienda que el efecto de transición sea ignorado. Si *endProgress* es igual a *startProgress*, la transición permanece en una progresividad fija.

Obsérvese que no todas las transiciones tendrán interpretaciones inversas significativas. Por ejemplo, un crossfade no es una transición geométrica y, por lo tanto, no tiene interpretación de sentido inverso. Las transiciones que no presentan una interpretación inversa deben obligatoriamente tener el atributo *direction* ignorado y adoptar el valor *default* “forward”.

Si el valor del atributo *type* no es “fade”, o el valor del atributo *subtype* no es “fadeToColor” o “fadeFromColor”, entonces el atributo *fadeColor* debe obligatoriamente ser ignorado.

Si el valor del atributo *transIn* o del atributo *transOut* no corresponde al valor del identificador XML de cualquiera de los elementos de transición definidos, entonces se recomienda que el valor del atributo sea considerado como siendo la secuencia (string) vacía y, por lo tanto, ninguna transición se debe realizar.

5.26 Módulo Metainformation

No existen comentarios que complementen o aclaren la ABNT NBR 15606-2.

6 Objetos de media en presentaciones NCL

6.1 Consideraciones generales

Lo descrito en 6.2 a 6.6 está relacionado y complementa la ABNT NBR 15606-2:2007, Sección 8 y ABNT NBR 15606-5:2008, Sección 8.

Como la arquitectura y la implementación del Ginga-NCL es una opción de cada desarrollador del receptor, el objetivo de lo que está descrito en 6.2 a 6.6 es apenas definir el funcionamiento esperado de un exhibidor de media al controlar los objetos que forman parte de un documento (aplicación) NCL.

Un objeto de media en ejecución es identificado por el atributo *id* del elemento <media> correspondiente y por los atributos *id* de los descriptores que fueron asociados al objeto. Esa identificación del objeto de media en exhibición se denomina *representationObjectId* en esta Sección 6.

Un exhibidor de media (o su adaptador) debe obligatoriamente ser capaz de recibir comandos de presentación, controlar las máquinas de estado de los eventos del objeto de media y responder a requisiciones del formateador. Las disposiciones dadas en 6.2 a 6.6 describen cómo un exhibidor de media reacciona a cada instrucción emitida por el formateador y cuál es el comportamiento de un objeto de composición (representado por los elementos <context>, <body> y <switch>).

6.2 Funcionamiento esperado de los exhibidores básicos de media

6.2.1 Consideraciones generales

Lo descrito en 6.2.2 a 6.2.11 trata de los exhibidores de media para los elementos <media> cuyos tipos son diferentes de “application/x-ginga-NCL” (o “application/x-ncl-NCL”), “application/x-ginga-NCLua” (o “application/x-ncl-NCLua”), “application/x-ginga-NCLet” (o “application/x-ncl-NCLet”) y text/html.

Lo descrito en 6.2.2 a 6.2.11 es totalmente basado en la ABNT NBR 15606-2:2007, 8.2.

6.2.2 Instrucción *start* para eventos de presentación

Antes de enviar una instrucción *start*, se recomienda que el formateador encuentre el exhibidor de media más adecuado para ser accionado, con base en el tipo de contenido que se exhibirá. Para ello, el formateador lleva en cuenta el atributo *player* asociado al objeto de media que se exhibirá. Si ese atributo no se especifica, el formateador debe obligatoriamente considerar el atributo *type* del elemento <media>. Si ese atributo tampoco se especifica, el formateador debe obligatoriamente considerar la extensión de archivo especificada en el atributo *src* del elemento <media>.

La instrucción *start* emitida por un formateador debe obligatoriamente informar los siguientes parámetros para el exhibidor de media: la localización del contenido del objeto de media que se controlará, la lista de todas las propiedades asociadas al objeto de media, la identificación *representationObjectId* del objeto en exhibición, la lista de eventos (presentación, selección o atribución) que necesitan ser controlados por el exhibidor de media, el evento de presentación cuya exhibición tiene que ser iniciada (llamado aquí de evento principal), el desplazamiento en el tiempo *offset-time* (cuando se especifica) y el tiempo de retardo *delay-time* (cuando se especifica).

NOTA 1 Los parámetros *offset-time* y *delay-time* son computados por el formateador NCL, antes de desencadenar el inicio de la instrucción, con base en el control del ciclo de vida de la aplicación, como, por ejemplo, basándose en un NCLEdittingCommand recibido.

NOTA 2 Se recomienda que las propiedades de presentación que no son obligatorias en la ABNT NBR 15606-2 sean ignoradas por el exhibidor de media.

El atributo *src* del elemento <media> es utilizado por el exhibidor de media para localizar el contenido que se exhibirá e iniciar su presentación. Si el contenido no pudiera ser localizado, o si el exhibidor de media no supiera como tratar el tipo de contenido, el exhibidor de media concluye la operación de inicio sin hacer ninguna acción.

Los descriptores deben obligatoriamente ser seleccionados por el formateador siguiendo las directrices definidas en el documento NCL. Si la instrucción *start* es el resultado de la acción de una relación (eslabón) que tiene un descriptor explícitamente declarado en su elemento <bind> (atributo *descriptor* del elemento <bind> hijo del elemento <link>), el descriptor resultante debe unificar los atributos del descriptor especificado en el eslabón con los atributos del descriptor especificado en el elemento <media> correspondiente, si ese atributo se hubiera especificado. Para los atributos comunes a los dos descriptores, la información del descriptor especificado en el elemento <bind> debe obligatoriamente sobreponerse a la información especificada en el descriptor definido por el elemento <media>. Si el elemento <bind> no contiene un descriptor explícito, el descriptor calculado por el formateador es el especificado por el elemento <media>, si ese atributo se hubiera especificado. De lo contrario, es escogido por el formateador un descriptor *default* para el tipo del elemento <media>. Con base en ese procedimiento, es construida la lista de los atributos *id* de los descriptores que son asociados al objeto y, unificando las propiedades del descriptor resultante con las propiedades explícitamente definidas en el elemento <media>, se define la lista de propiedades asociadas al objeto de media.

Se recomienda que la lista de eventos que se monitorizará por un exhibidor de media también sea computada por el formateador, teniéndose en cuenta la especificación del documento NCL. En la computación son evaluadas todas las relaciones en que participan el objeto de media y el descriptor resultante. Al computar los eventos que se monitorizarán, el formateador debe considerar obligatoriamente la perspectiva del objeto de media, es decir, el camino de elementos <body> y <context> anidados hasta alcanzar el elemento <media>. Solamente las relaciones contenidas en el elemento <body> y en los elementos <context> del camino deben ser obligatoriamente considerados en el cálculo de los eventos monitorizados.

El parámetro *offset-time* es opcional, “cero” es su valor *default,* y es relevante sólo para medias continua o medias estáticas con duración explícita. En ese caso, ese parámetro define un tiempo de desplazamiento, del inicio (tiempo de inicio) del evento principal, a partir del cual la presentación del evento principal será inmediatamente iniciada (es decir, comanda el exhibidor para que avance para el *tiempo de entrada*, igual al *tiempo de inicio* más el *tiempo de desplazamiento*). Obviamente, el valor del tiempo de desplazamiento debe obligatoriamente ser inferior a la duración del evento principal, de lo contrario, la instrucción *start* debe obligatoriamente ignorarse. Si el tiempo de desplazamiento es mayor que cero, el exhibidor de media debe obligatoriamente poner el evento principal en estado *occurring*, pero la transición del evento *start* no es notificada. Si el tiempo de desplazamiento es cero, el exhibidor de media debe obligatoriamente poner el evento principal en estado *occurring* y notificar el acontecimiento de la transición *start*.

Los eventos que tendrían su tiempo final antes del tiempo de inicio del evento principal y los eventos que tendrían su tiempo de inicio después del final del evento principal no necesitan ser monitorizados por el exhibidor de media (se recomienda que el formateador haga esa verificación al elaborar la lista de eventos monitorizados).

Los eventos monitorizados que tendrían su tiempo de inicio antes del tiempo de entrada del evento principal y el tiempo final después del tiempo de entrada del evento principal serán colocados en estado *occurring*, pero sus transiciones *start* no serán notificadas (las relaciones que dependen de esta transición, obligatoriamente, no se pueden accionar).

Los eventos monitorizados que tendrían su tiempo final después del tiempo de inicio del evento principal, pero antes del tiempo de entrada (tiempo de inicio + tiempo de desplazamiento) deben obligatoriamente tener su atributo *occurrence* incrementado, pero las transiciones *start* y *stop* no se notificarán.

El tiempo de retardo también es un parámetro opcional y su valor *default* es “cero”. Si es mayor que cero, ese parámetro contiene un tiempo que será esperado por el exhibidor de media antes de comenzar la presentación.

Si un exhibidor de media recibe una instrucción *start* para un objeto que ya se está presentando (pausado o no), obligatoriamente debe ignorar la instrucción y seguir controlando la presentación en marcha. En ese caso, el elemento <simpleAction> que causó la instrucción *start* no causa ninguna transición en la máquina de estados del evento correspondiente.

6.2.3 Instrucción stop para eventos de presentación

La instrucción *stop* sólo tiene que identificar el objeto de media que ya esté siendo controlado (*representationObjectId*). Identificar el objeto de media significa identificar el elemento <media> y los descriptores correspondientes. Por lo tanto, si un elemento <simpleAction> con atributo *actionType* igual a “stop” es vinculado por medio de una relación a una interfaz del nudo, la interfaz debe obligatoriamente ser ignorada cuando se ejecute la acción.

Si el objeto no se estuviere presentando (ninguno de los eventos en la lista de eventos del objeto estuviera en el estado *occurring* o *paused*) y el exhibidor de media no estuviera esperando, debido a una instrucción *start* retardada, la instrucción *stop* es ignorada. Si el objeto se está presentando, su evento principal (el evento transmitido como parámetro cuando el objeto de media fue iniciado) y todos los eventos en el estado *occurring* o *paused* con el tiempo final igual o anterior al tiempo final del evento principal, pasan para el estado *sleeping* y sus transiciones *stops* son notificadas. Los eventos monitorizados que estén en el estado *occurring* o *paused* con tiempo final posterior al tiempo final del evento principal son pasados al estado *sleeping*, pero sus transiciones *stop* no son notificadas y sus atributos *occurrence* no son incrementados. La presentación del contenido del objeto debe obligatoriamente ser interrumpida. Si el valor del atributo *repetition* del evento es mayor que cero, éste es disminuido y la presentación del evento principal es reiniciada después del tiempo entre repeticiones (el tiempo entre repeticiones es pasado para el exhibidor de media como el parámetro de retardo inicial). Si el objeto de media está esperando ser presentado después de una instrucción *start* retardada y una instrucción *stop* es emitida, la instrucción *start* anterior es eliminada.

La instrucción *stop* debe obligatoriamente cambiar los eventos monitorizados del objeto para el estado *sleeping*, sin importar si un efecto de transición está siendo aplicado al objeto de media. En otras palabras, el efecto de transición es interrumpido. Los efectos de transición no pueden ser aplicados después que un objeto sufre una instrucción *stop*.

Cuando todos los objetos de media que se refieren a flujos elementales de vídeo direccionados al plano de vídeo están en el estado sleeping, el vídeo principal debe obligatoriamente ser dimensionado para el 100 % de la pantalla. Un flujo de vídeo elemental puede ser redimensionado solamente usando un objeto de media (referido a él) en presentación. Lo mismo sucede con el audio principal. Cuando todos los objetos de media que se refieren a un flujo elemental de audio están en estado *sleeping*, el audio principal debe obligatoriamente ser dimensionado para el 100 % de su volumen.

6.2.4 Instrucción *abort* para eventos de presentación

La instrucción *abort* solo tiene que identificar el objeto de media que ya se está controlando (*representationObejctId*). Por lo tanto, si un elemento <simpleAction> con atributo *actionType* igual a “abort” es vinculado por medio de una relación a un interfaz de nudo, la interfaz debe obligatoriamente ser ignorada cuando la acción se ejecuta.

Si el objeto no se está presentando y no está esperando para ser presentado después de una instrucción *start* retardada, la instrucción *abort* es ignorada. Si el objeto se está presentando, el evento principal y todos los eventos en el estado *occurring* o *paused* pasarán para el estado *sleeping* y sus transiciones *abort* serán notificadas. Cualquier presentación del contenido debe parar obligatoriamente. Si el atributo *repetition* del evento es mayor que cero, éste debe obligatoriamente ser configurado para cero y la presentación del objeto de media no es reiniciada. Si el objeto de media está aguardando para ser presentado después de una instrucción *start* retardada y una instrucción *abort* es emitida, la instrucción anterior *start* debe obligatoriamente ser eliminada.

6.2.5 Instrucción *pause* para eventos de presentación

La instrucción *pause* sólo tiene que identificar el objeto de media que ya se esté controlando (*representationObejctId*). Por lo tanto, si un elemento <simpleAction> con atributo *actionType* igual a “pause” fuera vinculado por medio de una relación a una interfaz de nudo, la interfaz debe obligatoriamente ser ignorada cuando la acción se ejecute.

Si el objeto no se está presentando (es decir, el evento principal, pasado como parámetro cuando el objeto de media se inició, no está en estado *occurring*) y no está esperando para ser presentado después de una instrucción

start retardada, la instrucción es ignorada. Si el objeto se está presentando, el evento principal y todos los eventos monitorizados en estado *occurring* cambiarán para el estado *paused* y sus transiciones *pauses* serán notificadas. La presentación del objeto debe obligatoriamente ser pausada y el tiempo transcurrido de la pausa no puede ser considerado como parte de la duración del objeto. Como ejemplo, si un objeto tiene una duración explícita de 30 s y, después de 25 s se pausa, aunque el objeto queda en pausa, por ejemplo, por 7 min, después de retomar, el evento principal del objeto permanecerá aconteciendo por 5 s más. Si el evento principal no acontece debido a que el exhibidor de media espera el tiempo de exhibición debido a una instrucción *start* retardada, el objeto de media debe obligatoriamente esperar por una instrucción *resume* para continuar aguardando el retraso restante para la puesta en marcha.

6.2.6 Instrucción *resume* para eventos de presentación

La instrucción *resume* sólo tiene que identificar el objeto de media que ya se esté controlando (*representationObejctId*). Por lo tanto, si un elemento <simpleAction> con atributo *actionType* igual a “resume” es vinculado, por medio de una relación, a una interfaz de nudo, la interfaz debe obligatoriamente ser ignorada cuando la acción se ejecute.

Si el objeto no estuviera pausado (es decir, el evento principal, transmitido como parámetro cuando el objeto de media se inició, no estuviera en estado *paused*) y no estuviera esperando ser presentado después de una instrucción *start* retardada, la instrucción es ignorada. Si el exhibidor de media está aguardando el tiempo de exhibición debido a una instrucción *start* retardada, éste debe obligatoriamente retomar la espera a partir del instante en que fue pausado. Si el evento principal estuviese en estado *paused*, el evento principal y todos los eventos monitorizados en el estado *paused* pasarán al estado *occurring* y sus transiciones *resumes* serán notificadas.

6.2.7 Instrucción *start* para eventos de atribución

La instrucción *start* puede ser aplicada a una propiedad del objeto independientemente de que el hecho del objeto se esté presentando o no (en ese último caso, aunque el objeto no se esté presentando, su exhibidor de media ya debe obligatoriamente estar instanciado). La instrucción *start* tiene que identificar el objeto de media que está siendo controlado (*representationObjectId*) y un evento de atribución monitorizado, y determinar un valor a ser designado a la propiedad asociada al evento, la duración de la atribución y el paso de la atribución. Al atribuir un valor a la propiedad, el exhibidor de media pone la máquina de estado del evento correspondiente en el estado *occurring* y, después de terminar la atribución, pasa nuevamente al estado *sleeping*, generando la transición *start* y después la transición *stops*.

Para cada evento de atribución monitorizado, si el exhibidor de media cambia por sí solo el valor de la propiedad correspondiente, él debe obligatoriamente proceder como si hubiese recibido una instrucción *start* externa.

6.2.8 Instrucción *stop, abort, pause y resume* para eventos de atribución

Las instrucciones *stop, abort, pause* y *resume* tienen que identificar el objeto de media que se está controlando (*representationObjectId*) y el evento de atribución monitorizado.

La instrucción *stop* detiene el procedimiento de atribución, poniendo la máquina de estado del evento correspondiente en estado *sleeping*. Efectuándose la transición de la máquina de estado, la transición *stop* debe generarse.

La instrucción *abort* detiene el procedimiento de atribución, llevando la máquina de estado del evento correspondiente al estado *sleeping* y el valor de la propiedad a su valor inicial. Efectuándose la transición de la máquina de estado, la transición *abort* debe generarse.

La instrucción *pause* pausa el procedimiento de atribución, situando la máquina de estado del evento correspondiente en estado *paused*. Efectuándose la transición de la máquina de estado, debe generarse la transición *pause.*

Finalmente, la instrucción *resume* retoma el procedimiento de atribución, poniendo la máquina de estado del evento correspondiente en estado *occurring*. Efectuándose la transición de la máquina de estado, la transición *resume* debe generarse.

6.2.9 Instrucción addEvent

La instrucción *addEvent* se emite en caso de recepción de un comando de edición NCL *addInterface* (ver Sección 7). La instrucción tiene que identificar un objeto de media que ya se esté controlando y un nuevo evento que se incluirá en el conjunto de eventos siendo monitorizados. Todas las reglas aplicadas a la intersección de eventos monitorizados con el evento principal se aplicarán al nuevo evento. Si el tiempo de inicio del nuevo evento es anterior al tiempo actual de exhibición del objeto y el tiempo final del nuevo evento es posterior al tiempo actual de exhibición del objeto, el nuevo evento debe obligatoriamente ponerse en el mismo estado del evento principal (*occurring* o *paused*), sin notificar la transición correspondiente.

6.2.10 Instrucción de removeEvent

La instrucción *removeEvent* es emitida en caso de recepción de un comando de edición NCL *removeInterface*. La instrucción tiene que identificar un objeto de media que ya se esté controlando y un evento monitorizado que no pueda ser más controlado. El estado del evento debe obligatoriamente ponerse en estado *sleeping* sin generar ninguna transición.

6.2.11 Final natural de una presentación

Los eventos de un objeto con duración explícita o duración intrínseca normalmente finalizan sus presentaciones naturalmente, sin necesidad de instrucciones externas. En ese caso, el exhibidor de media debe obligatoriamente cambiar el evento al estado *sleeping* y notificar la transición *stop*. Lo mismo se debe hacerse obligatoriamente para los eventos monitorizados que estén en el estado *occurring* y que tengan el mismo tiempo final del evento principal o que tengan un tiempo de finalización desconocido, cuando el evento principal termina. Los eventos en estado *occurring* con tiempo final posterior al tiempo final del evento principal deben obligatoriamente ponerse en estado *sleeping*, pero sin generar la transición *stop* y sin incrementar el atributo *occurrence*. Es importante destacar que, si el evento principal corresponde a un ancla temporal interna al objeto, cuando la presentación de ese ancla termina, toda la presentación del objeto de media debe obligatoriamente terminar.

NOTA El autor del aplicativo debe considerar que el final natural de los contenidos recibidos como flujos elementales puede ser notificado solamente algún tiempo después, debido al retraso de los descriptores que transmiten esa información.

6.3 Funcionamiento esperado de los exhibidores hipermedia en aplicaciones NCL

Los objetos de media NCL (elementos <media> del tipo “application/x-ginga-NCL” o “application/x-ncl-NCL”) y objetos basados en HTML (elementos <media> del tipo“text/html”) incorporados en aplicaciones NCL siguen las directrices establecidas en Nested Context Language 3.0, Part 11.

6.4 Funcionamiento esperado de los exhibidores imperativos en aplicaciones NCL

6.4.1 Consideraciones generales

Los objetos imperativos pueden ser insertados en documentos NCL, trayendo capacidades computacionales adicionales a aplicaciones declarativas. La manera de insertar objetos imperativos en documentos NCL es definir un elemento <media> cuyo contenido (localizado por el atributo *src*) es el código imperativo que se ejecutará. Los perfiles EDTV y BDTV de la NCL 3.0 permiten que dos tipos de media sean asociados con el elemento <media>: application/x-ginga-NCLua (o “application/x-ncl-NCLua”), para códigos imperativos Lua (extensión de archivo .lua); y application/x-ginga-NCLet (o “application/x-ncl-NCLet”), para códigos imperativos Java (Xlet) (extensión de archivo .class o .jar), siendo este último opcional en el Ginga para receptores portátiles.

Los exhibidores imperativos para objetos NCL de los tipos “application/x-ginga-NCLua” (o “application/x-ncl-NCLua”) y “application/x-ginga-NCLet” (o “application/x-ncl-NCLet”) siguen las directrices establecidas en Nested

Context Language 3.0, Part 10, Sección 5. Esa sección es una traducción autorizada de esa referencia, debido a su importancia en la definición del puente entre el ambiente declarativo y un ambiente imperativo.

En un objeto imperativo, trechos de código imperativo se pueden asociar con un elemento <área> hijo por medio del atributo *label*. Un elemento <área> también se puede usar sólo como interfaz para ser usada como condición de eslabones para disparo de acciones en otros objetos NCL.

Un objeto imperativo tiene un ancla de contenido total (*whole content anchor*) definida por *default*. Ese ancla de contenido tiene, sin embargo, un significado especial. Representa la ejecución de cualquier trecho de código del objeto imperativo. Otro ancla también se define por *default*, el ancla de contenido principal (*main content anchor*). Siempre que un objeto imperativo sea iniciado sin la especificación de una de sus anclas de contenido, el ancla de contenido principal debe obligatoriamente ser asumida. En cualquier otra referencia al objeto imperativo sin especificar una de sus anclas o propiedades, que no sea la referencia para partida del objeto imperativo, el ancla de contenido total debe obligatoriamente ser asumida.

Los autores de un documento NCL pueden definir eslabones para iniciar, parar, pausar, retomar o abortar la ejecución de un código imperativo. Un exhibidor imperativo (máquina de ejecución del lenguaje) debe obligatoriamente hacer la interfaz entre el ambiente de ejecución imperativo y el formateador NCL.

Similarmente a los exhibidores de contenido de media perceptual (vídeo, audio, imagen etc.), los exhibidores de código imperativo deben obligatoriamente controlar las máquinas de estado de eventos asociados al objeto imperativo. Como ejemplo, si el código termina su ejecución, el exhibidor debe obligatoriamente generar la transición *stop* en la máquina de estado del evento de presentación correspondiente a la ejecución del código. Sin embargo, diferentemente de los exhibidores de contenido de media, un exhibidor de código imperativo no tiene informaciones suficientes para controlar, por sí solo, todas las máquinas de estado de evento y debe recurrir al contenido de la aplicación imperativa para comandar esos controles.

Eslabones NCL pueden ser conectados a las interfaces de un objeto imperativo (elementos <area> y <property> y las anclas de contenido definidas por *default*).

Si una relación inicia, parar, pausar, retomar o abortar la presentación de un ancla que representa un elemento <area> o el ancla de contenido principal (*main content anchor*), *callback* en el código imperativo deben obligatoriamente ser accionadas. La manera como las *callback* son definidas es de responsabilidad de cada código imperativo asociado al objeto imperativo.

Por otro lado, un código imperativo también puede comandar acciones para iniciar, parar, pausar o retomar eventos de presentación asociados a sus anclas de contenido por medio de una API ofrecida por el lenguaje. Las transiciones decurrentes pueden ser usadas como condiciones en los eslabones NCL para desencadenar acciones en otros objetos del mismo documento NCL. De esa manera, se puede establecer una sincronización de dos vías entre el código imperativo y el resto del documento NCL.

Un código imperativo también puede ser sincronizado con otros objetos por medio de los elementos <property>. Cuando el elemento <property> es mapeado para un trecho de código (función, método etc.) por medio de su atributo *name*, una acción “start”, aplicada a la propiedad, causa la ejecución del código, con los valores de la atribución siendo interpretados como parámetros pasados al trecho del código. Cuando el elemento <property> es mapeado para un atributo del código imperativo, la acción “start” debe obligatoriamente determinar el valor al atributo. Como de costumbre, la máquina de estado del evento asociado a la propiedad debe obligatoriamente ser controlada por el exhibidor del objeto imperativo, pero, a veces, comandado por la aplicación imperativa.

Un elemento <property> definido como hijo de un elemento <media> representando un objeto imperativo también puede ser asociado a un papel de *assessment* de un eslabón NCL. En ese caso, el formateador NCL debe consultar el valor de la propiedad para evaluar la expresión del eslabón. Si el elemento <property> es mapeado para un atributo del código, el valor del atributo es devuelto por el exhibidor del objeto imperativo para el formateador NCL. Si el elemento <property> es mapeado para un trecho de código, el código se debe ejecutar obligatoriamente y su valor de salida debe ser devuelto por el exhibidor del objeto imperativo para el formateador NCL.

6.4.2 Modelo de ejecución de un objeto imperativo

El ciclo de vida de un objeto imperativo es controlado por el formateador NCL. El formateador es responsable de desencadenar la ejecución de un objeto imperativo y mediar la comunicación entre ese objeto y otros objetos en un documento NCL.

Tal como sucede con todos los exhibidores de objetos de media, una vez instanciado, el exhibidor del objeto imperativo ejecuta un procedimiento de inicialización. Sin embargo, diferentemente de otros exhibidores de media, ese código de inicialización debe obligatoriamente ser especificado por el autor del código imperativo. El procedimiento de inicialización se ejecuta sólo una vez, para cada instancia del objeto, criando todos los trechos de código y datos que se pueden utilizar durante la ejecución del objeto imperativo y, en particular, registrando uno (o más) tratadores de eventos para la comunicación con el formateador NCL. Obsérvese que por lo menos el trecho de código asociado con el ancla de contenido principal (*main content anchor*) se crea durante el proceso de inicialización.

Tras la inicialización, la ejecución del objeto imperativo pasa a orientarse como evento, en ambas direcciones. Es decir, cualquier acción comandada por el formateador NCL llega a los tratadores de eventos registrados y cualquier notificación de alteración de estado de evento NCL es enviada como un evento al formateador NCL (por ejemplo, el fin natural de la ejecución de un trecho de código). El exhibidor del objeto imperativo estará entonces preparado para ejecutar cualquier instrucción según lo discutido en las próximas secciones.

6.4.3 Instrucciones para eventos de presentación

6.4.3.1 Consideraciones generales

Los formateadores NCL pueden controlar los exhibidores de objetos imperativos emitiendo instrucciones que pueden causar alteraciones en las máquinas de estado de los eventos de presentación (ejecuciones de trechos de código). Por otro lado, cualquier cambio de estado en esas máquinas de estado de evento de presentación es notificada al formateador NCL.

6.4.3.2 Instrucción start

La instrucción *start* emitida por un formateador debe obligatoriamente informar los siguientes parámetros para el exhibidor de objeto imperativo: la localización del contenido del objeto imperativo que se controlará; la lista de todas las propiedades asociadas al objeto de media, la identificación *representationObjectId* del objeto en exhibición, una lista de eventos (definidos por los elementos <area> y <property> hijos del elemento <media> y por las anclas de contenido definidas por *default*) que necesitan ser monitorizados por el exhibidor del objeto imperativo; el ancla de contenido identificada por *label*, o, por *default*, el ancla de contenido principal, identificando el código imperativo asociado que se iniciará; y un tiempo de retardo, opcional. A partir del atributo *src* del objeto de media, el exhibidor de objeto imperativo intenta localizar el código imperativo e iniciar su ejecución. Si el contenido no se pudiera localizar, se recomienda que el exhibidor termine la operación de inicio sin ejecutar ninguna acción.

Los descriptores son seleccionados por el formateador siguiendo las directrices definidas en el documento NCL. Si la instrucción *start* resulta de la acción de un eslabón que tiene un descriptor explícitamente declarado en su elemento <bind> (atributo *descriptor* del elemento <bind> hijo del elemento <link>), el descriptor resultante, calculado por el formateador, debe obligatoriamente unificar los atributos del descriptor del eslabón con los atributos del descriptor especificado en el elemento <media> correspondiente, si ese atributo hubiera sido especificado. Para los atributos comunes, la información del descriptor especificado en el elemento <bind> debe obligatoriamente sobreponer la información del descriptor especificado en el elemento <media> correspondiente. Si el elemento <bind> no contiene un descriptor explícito, el descriptor calculado por el formateador es el especificado en el elemento <media>, si ese descriptor se hubiera especificado. En caso contrario, un descriptor *default* para el tipo específico de ese objeto imperativo es elegido por el formateador. Con base en ese procedimiento, se construye la lista de los atributos *id* de los descriptores que son asociados al objeto y, unificando las propiedades del descriptor resultante con las propiedades definidas en el elemento <media>, se define la lista de propiedades del objeto de media.

Se recomienda que la lista de eventos que se monitorizará por un exhibidor de objeto imperativo sea computada por el formateador teniéndose en cuenta la especificación del documento NCL. El formateador debe verificar todas las relaciones en que el objeto imperativo, con el descriptor resultante, participa. Al computar los eventos que se monitorizarán, el formateador debe considerar la perspectiva del objeto de media, es decir, el camino de elemento <body> y elementos <context> hasta alcanzar el elemento <media>. Se recomienda que solamente las relaciones contenidas en el elemento <body> y en los elementos <context> se consideren en el cálculo de los eventos monitorizados.

Así como en cualquier otro elemento de <media>, el tiempo de retardo es un parámetro opcional y su valor *default* es “cero”. Si es mayor que cero, ese parámetro contiene un tiempo que esperará el exhibidor de objeto imperativo antes de comenzar la ejecución del código.

Diferentemente de lo que se realiza en otros elementos <media>, si un exhibidor de objeto imperativo recibe una instrucción *start* para un evento asociado con el ancla de contenido y ese evento está en estado *sleeping*, se inicia la ejecución del código imperativo asociado al elemento, aunque otra parte del código imperativo del objeto se esté ejecutando (pausada o no). Sin embargo, si el evento asociado con el ancla de contenido objetivo estuviese en estado *occurring* o *paused*, la instrucción *start* debe obligatoriamente ser ignorada por el exhibidor de código imperativo, que continuará controlando la ejecución en marcha. Como consecuencia, diferentemente de lo que pasa en otros elementos <media>, un elemento <simpleAction> con un atributo *actionType* igual a “stop”, “pause”, “resume” o ”abort” debe obligatoriamente ser vinculado por medio de un eslabón a una interfaz del objeto imperativo, que no puede ser ignorada cuando la acción se ejecute.

Toda vez que ni el formateador ni el exhibidor de código imperativo tienen ningún otro conocimiento sobre las anclas de contenido del objeto imperativo, excepto sus atributos *id* y *label*, ellos no saben qué otras anclas de contenido deben tener sus eventos asociados en el estado *occurring* cuando un ancla de contenido se inicie o esté en ejecución. Por lo tanto, excepto para el evento asociado a *whole content anchor*, es de responsabilidad del trecho de código imperativo, así que se inicie, comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado de evento que esté relacionada a la máquina de estado de evento asociada al código iniciado e informar si una transición asociada con una alteración se debe notificar. Similarmente, es de responsabilidad del trecho de código imperativo comandar cualquier alteración de estado de evento e informar si la transición asociada debe ser notificada, caso la ejecución del trecho de código inicie otro trecho de código asociado a un ancla de contenido.

Diferentemente de otros elementos <media>, si cualquier ancla de contenido se inicia y el evento asociado a la *whole content anchor* estuviera en estado *sleeping* o *paused*, éste debe obligatoriamente ser puesto en estado *occurring* y la transición correspondiente ser notificada.

6.4.3.3 Instrucción stop

La instrucción *stop* sólo tiene que identificar un trecho de código imperativo que ya se esté controlando. Identificar el trecho de código imperativo significa identificar el objeto de media en ejecución correspondiente (*representationObejctId*), y la interfaz del elemento de <media>.

Se recomienda que la instrucción *stop* emitida por un formateador NCL sea ignorada por un exhibidor de objeto imperativo si el trecho del código imperativo asociado con la interfaz especificada no estuviera siendo ejecutado (si el evento correspondiente no estuviese en el estado *occurring* o *paused*) y si el exhibidor de objeto imperativo no estuviera esperando debido a una instrucción *start* retardada. Si la interfaz del objeto imperativo está siendo ejecutada, su evento de presentación correspondiente debe obligatoriamente pasar para el estado *sleeping* y su transición *stop* correspondiente debe obligatoriamente ser notificada. La ejecución del código imperativo asociado con la interfaz debe obligatoriamente ser interrumpida. Si el atributo *repetition* del evento es mayor que cero, debe ser disminuido y el código imperativo asociado a la interfaz reiniciado después del tiempo de retardo de repetición (el retardo de repetición debe haber sido pasado para el exhibidor de media como parámetro de retardo de puesta en marcha). Si el objeto de media está aguardando para ser presentado después de una instrucción *start* retardada y se emite una instrucción *stop*, la instrucción *start* anterior debe obligatoriamente ser eliminada.

Por el mismo motivo discutido en la instrucción *start*, excepto para el evento asociado a la *whole content anchor*, es de responsabilidad del trecho de código interrumpido, antes de la interrupción total, comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado de evento que esté relacionada a la máquina de estado de evento asociada al código interrumpido e informar si la transición asociada a una alteración debe ser notificada.

Diferentemente de otros elementos <media>, si cualquier ancla de contenido se interrumpe y todos los otros eventos de presentación estuvieran en estado *sleeping,* la *whole content anchor* debe obligatoriamente colocarse en estado *sleeping*. Si un ancla de contenido es interrumpida y, por lo menos, otro evento de presentación estuviera en estado *occurring,* la *whole content anchor* debe obligatoriamente permanecer en estado *occurring*. En todos los otros casos, si un ancla de contenido es interrumpida, la *whole content anchor* debe obligatoriamente colocarse en estado *paused*. Si la instrucción *stop* es aplicada a un objeto imperativo sin especificar la interfaz del nudo, debe ser asumida la *whole content anchor*. En ese caso, se deben emitir instrucciones *stop* para todas las otras anclas de contenido.

6.4.3.4 Instrucción abort

La instrucción *abort* tiene que identificar un trecho de código imperativo que ya se esté controlando. Identificar el trecho de código imperativo significa identificar el objeto de media en ejecución correspondiente (*representationObejctId*), y la interfaz del elemento de <media>.

Se recomienda que si el código imperativo asociado a la interfaz del objeto no se está ejecutando y no está esperando ser ejecutado después de una instrucción *start* retardada, la instrucción *abort* sea ignorada. Si el código imperativo asociado a la interfaz del objeto se estuviese ejecutando, su evento asociado, en estado *occurring* o *paused*, debe obligatoriamente pasar al estado *sleeping*, y su transición *abort* correspondiente debe obligatoriamente ser notificada. Si el atributo *repetition* del evento es mayor que cero, debe ponerse en cero y la ejecución del código imperativo no podrá ser reiniciada. Si el código imperativo asociado a la interfaz del objeto estuviera aguardando para ser ejecutado después de una instrucción *start* retardada y se emite una instrucción *abort*, la instrucción *start* anterior debe obligatoriamente ser eliminada.

Por el mismo motivo discutido en la instrucción *start*, excepto para el evento asociado a la *whole content anchor*, es de responsabilidad del trecho de código abortado, antes de ser abortado, comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado del evento que esté relacionada a la máquina de estado de evento asociada al código abortado e informar si la transición asociada a una alteración debe ser notificada.

Diferentemente de otros elementos <media>, si cualquier ancla de contenido es abortada y todos los otros eventos de presentación estuvieran en estado *sleeping*, la *whole content anchor* debe obligatoriamente puesta en estado *sleeping*. Si un ancla de contenido es abortada y al menos otro evento de presentación está en estado *occurring*, la *whole content anchor* debe obligatoriamente permanecer en estado *occurring*. En todos los otros casos, si un ancla de contenido fuera abortada, la *whole content anchor* debe obligatoriamente colocarse en estado *paused*. Si la instrucción *abort* es aplicada a un objeto imperativo sin especificar la interfaz del nudo, la *whole content anchor* es asumida. En ese caso, se deben emitir instrucciones *abort* para todas las otras anclas de contenido.

6.4.3.5 Instrucción pause

La instrucción *pause* tiene que identificar un trecho de código imperativo que ya se esté controlando. Identificar el trecho de código imperativo significa identificar el objeto de media en ejecución correspondiente (*representationObejctId*), y la interfaz del elemento de <media>.

Se recomienda que si el código imperativo asociado a la interfaz del objeto no se está ejecutando (y no está en estado *paused*) y no está esperando ser ejecutado después de una instrucción *start* retardada, la instrucción sea ignorada. Si el código imperativo asociado a la interfaz del objeto se está ejecutando, su evento asociado en estado *occurring* debe obligatoriamente pasar al estado *paused*, su transición *pauses* correspondiente debe obligatoriamente ser notificada, y el tiempo transcurrido de la pausa no puede ser considerado como parte de la duración del objeto. Si el código imperativo asociado a la interfaz del objeto está esperando ser ejecutado después de una instrucción *start* retardada, la interfaz del objeto imperativo debe esperar obligatoriamente por una instrucción de reinicio para continuar aguardando el retraso restante.

Por el mismo motivo discutido en la instrucción *start*, excepto para el evento asociado a la *whole content anchor*, es de responsabilidad del trecho de código pausado, antes de la pausa completa, comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado de evento que esté relacionada a la máquina de estado de evento asociada al código pausado e informar si se debe notificar la transición asociada a una alteración.

Diferentemente de otros elementos <media>, si cualquier ancla de contenido fuera pausada y todos los otros eventos de presentación estuvieran en estado *sleeping* o *paused, la whole content anchor* debe obligatoriamente colocarse en estado *paused*. Si un ancla de contenido es pausada y al menos otro evento de presentación estuviera en estado *occurring*, la *whole content anchor* debe obligatoriamente permanecer en estado *occurring*. Si la instrucción *pause* fuese aplicada a un objeto imperativo sin especificar la interfaz del nudo, se asume la *whole content anchor*. En ese caso, se deben emitir instrucciones *pause* para todas las otras anclas de contenido que estén en estado occurring.

6.4.3.6 Instrucción resume

La instrucción *resume* tiene que identificar un trecho de código imperativo que ya se esté controlando. Identificar el trecho de código imperativo significa identificar el objeto de media en ejecución correspondiente (*representationObejctId*), y la interfaz del elemento de <media>.

Se recomienda que si el código imperativo asociado a la interfaz del objeto no está en pausa o el exhibidor de objeto imperativo no está esperando para ser ejecutado después de una instrucción *start* retardada, la instrucción sea ignorada. Si el exhibidor del objeto imperativo estuviera pausado aguardando el retraso del inicio, debe obligatoriamente retomar la espera a partir del instante en que fue pausado. Si el código imperativo asociado a la interfaz del objeto está pausado, su evento asociado debe obligatoriamente pasar al estado *occurring* y su transición *resume* correspondiente debe obligatoriamente ser notificada.

Por el mismo motivo discutido en la instrucción *start*, excepto para el evento asociado a la *whole content anchor*, es de responsabilidad del trecho de código a ser retomado, antes de retomar la ejecución, comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado de evento que esté relacionada a la máquina de estado de evento asociada al código retomado e informar si la transición asociada a una alteración debe ser notificada.

Diferentemente de otros elementos <media>, si cualquier contenido ancla es retomado, la *whole content anchor* debe obligatoriamente ser configurada para el estado *occurring*. Si la instrucción *resume* fuera aplicada a un objeto imperativo sin especificar la interfaz del nudo, es asumida la *whole content anchor*. Si la *whole content anchor* no estuviera en el estado *paused* por haber recibido una instrucción *pause* anterior, la instrucción *resume* debe obligatoriamente ser ignorada. Caso contrario, se deben emitir instrucciones *resume* para todas las otras anclas de contenido que estén en estado *paused*, excepto las que ya estaban pausadas antes que la *whole content anchor* recibiera la instrucción *pause*.

6.4.3.7 Fin natural de la ejecución de un código

Los eventos de un objeto imperativo normalmente terminan su ejecución naturalmente, sin necesidad de instrucciones externas. En ese caso, inmediatamente antes de la finalización, un trecho del código debe comandar el exhibidor de código imperativo para alterar el estado de cualquier otra máquina de estado de evento que esté relacionada a la máquina de estado de evento asociada al código que terminó e informar si la transición asociada a una alteración debe ser notificada. El evento de finalización de la presentación debe obligatoriamente pasar para el estado *sleeping* y se debe notificar la transición *stop*. Si el atributo *repetition* del evento es mayor que cero, debe ser disminuido y el código imperativo asociado a la interfaz reiniciado, después del tiempo de retardo de repetición (el retardo de repetición debe haber sido pasado para el exhibidor de media como parámetro de retardo inicial).

Diferentemente de otros elementos <media>, si cualquier ejecución del ancla de contenido termina y todos los otros eventos de presentación están en el estado *sleeping*, la *whole content anchor* debe obligatoriamente ponerse en el estado *sleeping*. Si la ejecución de un ancla de contenido termina y, por lo menos, otro evento de presentación está en el estado *occurring*, la *whole content anchor* debe obligatoriamente permanecer en el estado *occurring*. En todos los otros casos, si la ejecución de un ancla de contenido termina, la *whole content anchor* debe obligatoriamente ser colocada en el estado *paused*.

6.4.4 Instrucciones para eventos de atribución

6.4.4.1 Consideraciones generales

Formateadores NCL también pueden enviar instrucciones que causen alteraciones en máquinas de estado de eventos de atribución (ejecuciones de trechos de código). De forma similar a los eventos de presentación, cualquier alteración de estado en las máquinas de estado de evento de atribución es notificada al formateador NCL.

Aunque propiedades del nudo imperativo se puedan asociar a trechos de código, la ejecución de esos trechos no altera ninguna máquina de estado asociada a anclas de contenido del objeto imperativo.

6.4.4.2 Instrucción start

La instrucción *start* emitida por un formateador puede ser aplicada a una propiedad de un objeto imperativo, independientemente de que el objeto esté en ejecución (es decir, la *whole content anchor* esté en el estado *occurring*) o no (en ese último caso, aunque el objeto no se esté ejecutando, su exhibidor de objeto imperativo debe obligatoriamente haber sido instanciado). En ambos casos, la instrucción *start* tiene que identificar el objeto imperativo (*representationObejectId*), un evento de atribución monitorizado y, en su caso, un valor que pasará para el código imperativo relacionado con el evento. Al recibir la acción *start*, el exhibidor de objeto imperativo debe obligatoriamente colocar la máquina de estado del evento en el estado *occurring* y, después de terminar la atribución, nuevamente en estado *sleeping*, generando la transición *start* y después la transición *stop*.

Obsérvese que, si una instrucción *start* es aplicada a un elemento <property> que solicita la ejecución de un trecho de código, ningún estado de ancla de contenido puede ser afectado.

Para cada evento de atribución monitorizado, si el trecho del código del objeto imperativo cambia por sí solo el valor del atributo correspondiente, él también debe obligatoriamente comandar el exhibidor de código imperativo, que se comportará como si hubiera recibido una instrucción *start* externa.

6.4.4.3 Instrucción stop, abort, pause y resume

Con excepción de la instrucción *start*, discutida en la sección anterior, todas las otras instrucciones tienen el mismo efecto en la atribución de la propiedad que tienen en la atribución de propiedades de cualquier otro tipo de objeto.

Las instrucciones *stop, abort, pause* y *resume* tienen que identificar el objeto de media que se está controlando (*representationObjectId*) y el evento de atribución monitorizado.

La instrucción *stop* solamente interrumpe el proceso de atribución de la propiedad, llevando la máquina de estado del evento de atribución al estado *sleeping*.

La instrucción *abort* interrumpe el proceso de atribución de la propiedad, poniendo la máquina de estado del evento de atribución en el estado *sleeping* y restableciendo el valor original de la propiedad.

La instrucción *pause* solamente pausa el proceso de atribución de la propiedad, llevando la máquina de estado del evento de atribución al estado *paused*.

La instrucción *resume* sólo retoma el proceso de atribución de la propiedad, poniendo la máquina de estado del evento de atribución en el estado *occurring*.

6.5 Funcionamiento esperado de los exhibidores de medias tras las instrucciones aplicadas a objetos compuestos

6.5.1 Consideraciones generales

Lo descrito en 6.5.2 a 6.5.6 se aplica solamente a acciones aplicadas a objetos representados por elementos <context>, <body> y <switch>.

Una <simpleCondition> o <simpleAction> con el valor del atributo *eventType* igual a “presentation” puede ser conectada por un eslabón a un nudo de composición (representado por un elemento <context>, <switch> o <body>) como un todo (es decir, sin la información de una interfaz). De forma similar, una <attributeAssessment> con el valor del atributo *eventType* igual a “presentation” y el *attributeType* igual a “state”, “occurrences” o “repetitions” puede ser conectada por un eslabón a un nudo de composición (representado por un elemento <context>, <switch> o <body>) como un todo, con el valor del atributo procedente de la máquina de estado del evento de presentación definida sobre el nudo de composición. Si una <simpleAction> con valor del atributo *eventType* igual a “presentation” es vinculada por un eslabón a un nudo de composición (representado por un elemento <context> o <body>) como un todo (es decir, sin la información de una interfaz), la instrucción también se reflejará en las máquinas de estado de los eventos de los nudos hijos de la composición, según se explica en las subsecciones siguientes.

6.5.2 Iniciando una presentación de contexto

Si un elemento <context> o <body> participa de un papel acción, con el tipo de la acción igual a “*start*”, cuando la acción es accionada sin la especificación de una interfaz, la instrucción *start* también debe ser obligatoriamente aplicada a todos los eventos de presentación mapeados por las puertas de los elementos <context>/<body>.

Si el autor pretende inicia la presentación usando una puerta específica, debe obligatoriamente indicar el *id* del elemento <port> como el valor del atributo *interface* del elemento <bind> correspondiente. En ese caso, la instrucción *start* debe obligatoriamente ser aplicada sólo al evento de presentación mapeado por la puerta del elemento <context>/<body>.

6.5.3 Interrumpiendo una presentación de contexto

Si un elemento <context> o <body> participa de un papel acción, con el tipo de acción igual a “stop”, cuando esa acción es accionada sin la especificación de una interfaz, la instrucción *stop* también será aplicada a todos los eventos de presentación de los nudos hijos de la composición.

Si el nudo de composición contiene relaciones siendo evaluadas (o con su evaluación en pausa), la evaluación debe obligatoriamente ser suspendida y ninguna acción debe ser accionada.

6.5.4 Abortando una presentación de contexto

Si un elemento <context> o <body> participa de un papel acción, con el tipo de acción igual a “abort”, cuando esa acción es accionada sin la especificación de una interfaz, la instrucción *abort* también debe obligatoriamente ser aplicada a todos los eventos de presentación de los nudos hijos de la composición.

Si el nudo de composición contiene relaciones siendo evaluadas (o con su evaluación en pausa), la evaluación debe obligatoriamente ser suspendida y ninguna acción debe ser accionada.

6.5.5 Pausando una presentación de contexto

Si un elemento <context> o <body> participa de un papel acción, con el tipo de acción igual a “pause”, cuando esa acción es accionada sin la especificación de una interfaz, la instrucción *pause* también debe obligatoriamente ser aplicada a todos los eventos de presentación de los nudos hijos de la composición que están en el estado *occurring*.

Si el nudo de composición contiene relaciones siendo evaluadas, todas las evaluaciones deben obligatoriamente ser suspendidas hasta que sea emitida una acción de reiniciar, parar o abortar.

Si el nudo de composición contiene nudos hijos con eventos de presentación ya en el estado de pausa cuando la acción de pausa en la composición sea emitida, esos nudos deben ser identificados porque si la composición recibe una instrucción *resume*, esos eventos no pueden ser retomados.

6.5.6 Retomando una presentación de contexto

Si un elemento <context> o <body> participa de un papel acción, con el tipo de acción igual a “resume”, cuando esa acción es accionada sin la especificación de una interfaz, la instrucción *resume* también debe obligatoriamente ser aplicada a todos los eventos de presentación de los nudos hijos de la composición que están en el estado *paused*, excepto los que ya estaban pausados antes de la composición ser pausada.

Si la composición contiene relaciones con evaluaciones pausadas, éstas deben obligatoriamente ser retomadas.

6.6 Relación entre la máquina de estado del evento de presentación de un objeto NCL y la máquina de estado del evento de presentación de su nudo (objeto) padre

Esta subsección se aplica a objetos padre representados por elementos <context>, <body>, <switch>, y elementos de <media> del tipo “application/x-ginga-NCL” (o “application/x-ncl-NCL”).

Siempre que un evento de presentación de un objeto hijo (de media o de composición) va para el estado *occurring*, el evento de presentación del objeto de composición padre o del objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) que contiene el objeto hijo también entra en el estado *occuring*.

Cuando todos los objetos hijos de un objeto de composición o de un objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) tengan sus eventos de presentación en el estado *sleeping*, el evento de presentación del objeto de composición padre o del objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) también entra en el estado *sleeping*.

Los objetos de composición u objetos NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) no tienen que inferir en las transiciones *abort* de sus objetos hijos. Esas transiciones en eventos de presentación de objeto de composición u objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) acontecen solamente cuando las instrucciones *abort* son aplicadas directamente al evento de presentación del objeto de composición o del objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”).

Cuando todos los objetos hijos de un objeto de composición o de un objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) tengan sus eventos de presentación en un estado diferente a *occurring* y, por lo menos, un objeto hijo tenga su evento principal en el estado *paused,* el evento de presentación del objeto de composición o del objeto NCL del tipo “application/x-ncl-NCL” (o “application/x-ginga-NCL”) también debe estar en el estado *paused*.

Si un elemento <switch> es iniciado, pero no define un componente *default* y ninguna de las reglas referidas por los elementos <bindRule> es evaluada como verdadera, la presentación del switch no puede entrar en el estado *occurring*.

7 Edición en vivo y eventos de flujo NCL

7.1 Consideraciones generales

El núcleo de la máquina de presentación del Ginga-NCL se compone de: el formateador NCL y el módulo gestor de base privada.

El formateador NCL es responsable de recibir un documento NCL y controlar su presentación con el objetivo de asegurar que se respeten las relaciones especificadas entre los objetos de media. El formateador NCL trata con documentos que son colectados dentro de una estructura de datos conocida como *base privada*. El Ginga asocia al menos una base privada con cada canal de TV. Los documentos NCL en una base privada pueden ser iniciados, pausados, retomados, interrumpidos y pueden referirse uno al otro.

El gestor de base privada es responsable de recibir Comandos de Edición NCL y mantener los documentos activos NCL (documentos que se presentan).

El protocolo DSM-CC es adoptado en el SBTVD-T para transportar Comandos de Edición en flujos elementales MPEG-2 TS, cuando los comandos son procedentes del canal de transmisión terrestre. Los eventos de flujo DSM-CC y el protocolo carrusel de objetos DSM-CC (ver ABNT NBR 15606-3) constituyen la base para el tratamiento de documentos en la máquina de presentación del Ginga, de acuerdo con el SBTVD-T.

Los Comandos de Edición son codificados como eventos de flujo DSM-CC. La máquina de presentación del Ginga debe obligatoriamente ser capaz de gestionar, al menos, el evento de flujo del Comando de Edición, cuya sintaxis se muestra en la Tabla 42.

Tabela 42 — Descriptor del evento de flujo del comando de edición

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

Los objetos de evento de flujo de un carrusel de objetos son usados para mapear los nombres de los eventos de flujo en identificaciones de los eventos de flujo. Los objetos de evento de flujo son usados para informar el Ginga sobre eventos de flujo DSM-CC que se pueden recibir. Los nombres de eventos permiten especificar los tipos de eventos, ofreciendo un nivel mayor de abstracción para aplicaciones del *middleware* al registrarse y tratar eventos de flujo DSM-CC. El gestor de base privada, así como los objetos de ejecución NCL (por ejemplo, objetos NCLua), deben obligatoriamente registrarse como oyentes de eventos de flujos tratados por ellos, usando nombres de evento.

Los archivos de documentos NCL y los contenidos de los objetos de media NCL son organizados en estructuras de sistemas de archivos. Los parámetros del comando (*command parameters*) de edición, basados en XML, se pueden transportar directamente en la carga (*payload*) de un descriptor de eventos de flujo o, alternativamente, organizados en estructuras de sistemas de archivos que se transportarán, cada una, en un carrusel de objetos. Se usa un generador de carrusel DSM-CC para unir los sistemas de archivos y los objetos de eventos de flujo en un flujo de datos elemental.

La ABNT NBR 15606-2: 2007, Tabla 56, presenta los comandos y los parámetros cargados en el campo de carga de un descriptor de evento *nclEditingCommand*. Los comandos son divididos en tres grupos: el primero, para operación de base privada (para abrir, cerrar, activar, desactivar y salvar bases privadas), el segundo, para manipulación de documentos en una base privada (para adicionar, remover y salvar un documento en una base privada abierta, e iniciar, pausar, retomar e interrumpir las presentaciones del documento en una base privada activa), y el último, para tratar con entidades NCL en una base privada abierta. Para cada entidad NCL, se definen los comandos *add* y *remove*. Si una entidad ya existe, el comando *add* tiene la semántica de actualización (alteración).

NOTA Los parámetros de los Comandos de Edición, basados en XML, siguen el esquema NCL30EdCommand.xsd XML, según lo definido en la ABNT NBR 15606-2. Informalmente, eso significa que la especificación de un parámetro de edición basado en XML es idéntica a la especificación del elemento, a la que se refiere, en los perfiles de EDTV (o BDTV), excepto en el caso del comando *addInterface*, donde el atributo *begin* de un elemento <area> puede recibir el valor "now", especificando o NPT actual del nodeId, que identifica el vídeo principal MPEG en presentación por el decodificador del *hardware*.

Cuando un comando de edición del documento NCL tiene que ser enviado por difusión, un objeto de evento de flujo DSM-CC debe ser creado, mapeando la secuencia "*nclEditingCommand*" en un *id* de flujo de evento seleccionado (ver ABNT NBR 15606-2), y puesto como objeto en un carrusel de objetos DSM-CC. Uno o más descriptores de eventos de flujo DSM-CC, con la *id* seleccionada anteriormente, son entonces creados y enviados en otro flujo elemental MPEG-2 TS. Esos descriptores de eventos de flujo generalmente tienen su referencia de tiempo configurada a cero, pero pueden tener la ejecución prorrogada para un momento específico. El gestor de base privada debe obligatoriamente registrarse como un oyente *"nclEditingCommand",* siendo notificado cuando ese tipo de evento de flujo llegue. El *commandTag* recibido es utilizado entonces por el gestor de base privada para interpretar la semántica completa del *comando*. Si el parámetro del comando, basado en XML, fuera suficientemente corto, puede ser transportado directamente en la carga de los descriptores del evento de flujo, junto a los otros parámetros del comando. En caso contrario, el privateDataPayload carga un conjunto de pares de referencia. En el supuesto de archivos (documentos NCL o contenidos de objetos NCL) recibidos sin solicitud (*pushed data*), cada par relaciona un conjunto de caminos (path) de archivos con su respectiva localización en el sistema de transporte. En el supuesto de archivos recibidos por demanda (*pulled data*) de un canal de interactividad o localizados en el propio receptor, no es necesario enviar ningún par de referencias, excepto el par (uri, "null"} asociado al documento NCL o a la especificación de nudo XML que el comando adicionará.

Receptores que apenas implementan el perfil NCL BDTV no están obligados a tratar con los siguientes comandos: *pauseDocument*, *resumeDocument*, *addTransition*, *removeTransition*, *addTransitionBase* y *removeTransitionBase*.

El Ginga asocia al menos una base privada a cada canal de TV (conjunto de servicios): llamada base privada *default* del canal. Cuando un canal es sintonizado, su base privada default es abierta y activada por el gestor de base privada. Las otras bases privadas deben obligatoriamente ser desactivadas. A través del canal sintonizado otras bases privadas pueden ser abiertas (o creadas), pero, como máximo una puede ser asociada con cada servicio del canal sintonizado. Cuando el canal tiene apenas un servicio, como es el caso de la recepción one-seg, una, y solamente una, base privada es asociada a cada canal de TV (la base privada *default* del canal). En ese caso, cualquier comando de edición que venga por el canal de difusión para abrir una base privada debe ignorarse.

La base privada asociada al canal de TV debe obligatoriamente tener el identificador (parámetro baseId) igual al "valor del campo network_id de la NIT (table_id 0x40)". Las otras posibles bases privadas asociadas a un servicio del canal de TV deben tener obligatoriamente su identificador (*baseId parameter*) igual al "valor del campo network_id de la NIT.valor del campo program_number de la PMT". El valor del *program_number* es el asociado al servicio correspondiente.

Las aplicaciones NCL residentes son gestionadas en una base privada específica.

Por razones de seguridad, solo se puede activar una base privada de cada vez, entre las controladas por medio del canal de transmisión sintonizado. La manera más sencilla y restricta para gestionar bases privadas es tener sólo una base privada abierta de cada vez, entre las controladas por medio del canal de transmisión sintonizado. Sin embargo, el número de bases privadas que se pueden mantener abiertas es una decisión de la implementación específica del *middleware*.

NOTA 1 Solamente bases privadas creadas por comandos de edición NCL enviados por el canal de difusión sintonizado y la base privada default del canal de TV pueden ser controladas por comandos de edición NCL procedentes del mismo canal sintonizado. Así, para receptores one-seg, sólo la base privada *default* asociada al canal de televisión sintonizado puede ser controlada por los comandos de edición NCL enviados por medio del mismo canal sintonizado.

NOTA 2 Los eventos NCLua y NCLet (eventos de comando de edición NCL) generados por objetos imperativos de aplicaciones operando en una base privada asociada a un canal de TV solamente pueden controlar las bases privadas asociadas a ese canal.

Los comandos *add* siempre tienen entidades NCL como sus argumentos (parámetros de comando basados en XML). Si la entidad especificada ya existe o no, la consistencia del documento debe obligatoriamente mantenerse por el formateador NCL, en el sentido de que todos los atributos de la entidad, clasificados como necesarios, se definan. Existe solamente una excepción para esa regla – el atributo *interface* de un elemento <bind> hijo de un elemento <link> puede dejarse inconsistente, refiriéndose a un elemento <area> a ser cumplimentado por un comando *addInterface* cuyo atributo *begin* tiene el valor "now", según lo especificado en la ABNT NBR 15606-2. En ese caso, el elemento <link> debe obligatoriamente evaluarse inmediatamente que se reciba el comando *addInterface.*

Se recomienda pautar el desarrollo de aplicaciones por las buenas prácticas de uso. En la interacción a través de mandos a distancia convencionales, se recomienda utilizar una tecla estándar para habilitar y deshabilitar aplicaciones en el plano gráfico.

7.2 Identificación de recursos

Los comandos de edición "addDocument" y "addNode" deben obligatoriamente usarse para mapear las estructuras de datos del servidor para las estructuras de datos utilizadas en el receptor, cuando documentos XML referenciados por esos comandos (archivos de documento NCL u otros archivos de documento XML, respectivamente) se transmitan por medio de un carrusel de objeto. En ese caso, en el mismo carrusel de objetos que carga la especificación del documento XML, un objeto de evento de flujo debe transmitirse obligatoriamente con el fin de mapear el nombre "*nclEditingCommand*" al eventId del descriptor de evento de flujo DSM-CC, que carga el comando de edición addDocument o addNode. El privateDataPayload del descriptor de evento de flujo debe cargar obligatoriamente un conjunto de pares de referencia (uri, id). El parámetro uri del primer par puede tener el esquema "x-sbtvd" (opcional) y el camino absoluto del documento NCL o de la especificación de nudo NCL (el camino en el servidor de datos). El parámetro correspondiente en el par id debe obligatoriamente referirse a la IOR del documento NCL o a la IOR de la especificación del nudo NCL (carouselId, moduleId, objectKey; ver ABNT NBR 15606-3 e ISO/IEC 13818-6) en el carrusel de objetos. Si otros sistemas de archivos con contenido de medias se tienen que transmitir por medio de otros carruseles de objetos para completar el comando de edición (como es usual en el caso de los comandos addDocument o addNode), otros pares (uri, id) deben obligatoriamente estar presentes en el comando. En ese caso, el parámetro uri puede tener el esquema "x-sbtvd" (opcional) y el camino absoluto de la raíz del sistema de archivos (el camino en el servidor de datacast); ya el parámetro id correspondiente en el par debe obligatoriamente referirse a la IOR (carouselId, moduleId, objectKey; ver ABNT NBR 15606-3 e ISO/IEC 13818-6) de cualquier directorio o archivo secundario de la raíz en el carrusel de objetos (la IOR del *service gateway* del carrusel).

Generalmente, la transmisión de sistemas de archivos por medio del uso de otros carruseles de objeto, diferentes de los que cargan el objeto de evento, es necesaria cuando los sistemas de archivos que representan la aplicación no pueden ser modelados por una única estructura de datos. Se presenta un ejemplo en Nested Context Language 3.0, Part 9.

7.3 Comando startDocument

Con la intención de estar conforme a la ITU-T H.761 para el Ginga-NCL, el comando startDocument se define según la Tabla 43.

Tabela 43 — Definición del comando startDocument

String de comando	Tag de comando	Descripción
startDocument (baseId, documentId, interfaceId, offset,nptBaseId, nptTrigger) [a, b]	0x07	Inicia la reproducción de un documento NCL en una base privada activa, iniciando la presentación a partir de una interfaz específica del documento. La referencia del tiempo transportada en el campo nptTrigger establece el punto de inicio del documento, respecto a la base de tiempo NPT identificada por el campo *nptBaseId.* Pueden acontecer tres casos: a) Si nptTrigger es diferente de cero y mayor o igual al valor NPT corriente de la base temporal NPT identificada por nptBaseId, se espera hasta que NPT alcance el valor dado en nptTrigger y comienza la exhibición del documento de su punto inicial en el tiempo + *offset*; b) Si nptTrigger es diferente de cero y menor que el valor de NPT corriente de la base temporal NPT identificada por nptBaseId, el inicio de la exhibición del documento es inmediato y desplazado en el tiempo de su punto inicial del valor "*offset* + $(NPT - nptTrigger)_{segundos}$"; [c] c) Si nptTrigger es igual a 0, la exhibición del documento es inmediata y a partir de su punto inicial en el tiempo + *offset*

[a] El parámetro *offset* especifica un valor de tiempo

[b] El nptTrigger es obligatoriamente un valor de NPT, y el nptBaseId es obligatoriamente un identificador de una base de tiempo NPT.

[c] Sólo en ese caso, el parámetro *offset* puede recibir un valor negativo, pero *offset* + $(NPT - nptTrigger)_{segundos}$ debe ser un valor positivo.

7.4 Correspondencia entre el control del ciclo de vida usando AIT y *nclEditingCommand*

Para ejecutar una aplicación el receptor necesita saber algunas informaciones sobre ella: dónde encontrar sus archivos, cómo la aplicación es denominada y cuándo el receptor debe iniciarla. En aplicaciones vinculadas al service sintonizado (*service-bound applications*), el Ginga puede usar una tabla SI extra, denominada tabla de informaciones de aplicación (o AIT) para inferir las informaciones mencionadas arriba, además de lo ofrecido por el uso de los comandos de edición NCL. Cada servicio que tiene una aplicación asociada a él puede contener una AIT, y cada AIT puede contener una descripción de cada aplicación, que puede ejecutarse mientras el servicio se está exhibiendo.

La AIT se transmite con el ID de la tabla igual a 0x74. Cada AIT contiene dos lazos (*loops*) de nivel superior. El primero de ellos es el lazo común, que contiene un conjunto de descriptores que se refieren a todas las aplicaciones señaladas en aquella AIT particular, o que se aplican al servicio como un todo. El otro lazo de nivel superior es el lazo de aplicaciones, que describe las aplicaciones señaladas por aquella instancia de AIT. Cada interacción en el lazo de aplicaciones describe una aplicación (es decir, representa una línea en la tabla de aplicaciones señaladas). Entre esas informaciones, está el código de control de la aplicación que, en el caso de aplicaciones NCL, debe obligatoriamente adoptar uno de los valores mostrados en la Tabla 44.

Tabela 44 — Control del ciclo de vida de aplicaciones NCL usando AIT (campo AIT: application_type =“0x0009”)

campo AIT: application_ control_code	Descripción	*nclEditingCommand* equivalente	Restricción
AUTOSTART (0x01)	La aplicación se inicia automáticamente cuando el servicio es seleccionado	addDocument (baseId, {uri, id}+); seguido de startDocument (baseId, documentId, interfaceId, offset, null, null) para adicionar (si ya no está adicionada) e iniciar una aplicación en la base privada asociada con el servicio sintonizado	a) La base privada es identificada por el “valor del campo network_id de la NIT”, o el “valor del campo program_number de la PMT que se refiere a la AIT”; b) Para el comando addDocument equivalente: Hay solamente un parámetro uri que debe obligatoriamente recibir el valor definido en el campo base_directory del ginga_ application_location_descriptor. Hay solamente un parámetro de id que debe obligatoriamente recibir el valor IOR correspondiente al documento definido en el campo initial_entity del ginga_ application_location_descriptor. c) Para el comando addDocument equivalente: El atributo id del elemento <ncl> debe recibir el mismo valor del campo application_id de la estructura application_identifier de la AIT; d) La aplicación NCL debe obligatoriamente ser iniciada a partir de todas las interfaces del elemento <body>; e) La aplicación NCL debe obligatoriamente ser iniciada inmediatamente, ya que no puede tener su tiempo inicial refiriéndose a un valor NPT, que se supone ser "do it now"; f) La aplicación NCL debe obligatoriamente ser iniciada de su comienzo, ya que el parámetro *offset* se asume como "0".
PRESENT (0x02)	La aplicación no se inicia automáticamente. Ella se pone en una lista de aplicaciones disponibles en el receptor, pudiendo ser seleccionada por un usuario para ser entonces iniciada	Semejante al AUTOSTART, salvo que la aplicación se inicia por el usuario y no automáticamente.	Ver AUTOSTART
DESTROY (0x03)	La aplicación es interrumpida	stopDocument (baseId, documentId) para interrumpir una aplicación en la base privada asociada al servicio de TV sintonizado o en la base privada *default* asociada con el canal de TV sintonizado. El parámetro baseId debe obligatoriamente identificar la base privada asociada al servicio o la base privada *default* asociada al canal de TV sintonizado	a) El parámetro baseId en los comandos de edición NCL debe recibir "el valor del campo network_id de la NIT” o “el valor del campo program_number de la PMT que se refiere a AIT"; b) El atributo *id* del elemento <ncl> debe obligatoriamente recibir el mismo valor del campo application_id de la estructura application_identifier de la AIT; c) La aplicación NCL debe obligatoriamente ser inmediatamente interrumpida, ya que no puede tener su tiempo final refiriéndose a un valor NPT, que se asume como siendo "do it now"

Tabla 44 (continuación)

campo AIT: application_ control_code	Descripción	*nclEditingCommand equivalente*	Restricción
KILL (0x04)	La aplicación es interrumpida y retirada del receptor	stopDocument (baseId, documentId) seguido de removeDocument(baseId, documentId) para interrumpir una aplicación, en la base privada asociada al servicio sintonizado o en la base privada *default* asociada con el canal de TV sintonizado, y entonces removerla de la base privada. El parámetro baseId debe obligatoriamente identificar la base privada asociada al servicio o la base privada *default* asociada al canal de TV sintonizado	a) El parámetro baseId en los comandos de edición NCL debe recibir "el valor del campo network_id de la NIT" o "el valor del campo program_number de la PMT que se refiere a AIT"; b) El atributo *id* del elemento <ncl> debe obligatoriamente recibir el mismo valor del campo application_id de la estructura application_identifier de la AIT; c) La aplicación NCL debe obligatoriamente ser inmediatamente interrumpida, ya que no puede tener su tiempo final refiriéndose a un valor NPT, que se asume como siendo "do it now" d) Una aplicación NCL no puede ser reiniciada después de haber sido interrumpida por el control KILL, sin tener todo el proceso AUTOSTART rehecho (salvo si ella fuese iniciada por PREFETCH o UNBOUND)
PREFETCH (0x05)	El exhibidor NCL es preparado. El gestor de base privada debe obligatoriamente esperar el comando addDocument (baseId, {uri, id}+) para adicionar la aplicación en la base privada baseId; y el comando startDocument (baseId, DocumentId, interfaceId, offset, nptBaseId, nptTrigger) para disparar la aplicación.		a) El parámetro baseId en los comandos de edición NCL debe recibir "el valor del campo network_id del NIT" o "el valor del campo program_number de la PMT que se refiere a AIT".
REMOTE (0x06)	La aplicación remota solamente estará disponible para ser ejecutada tras la selección del servicio. El exhibidor NCL es preparado y el gestor de base privada debe obligatoriamente esperar el comando addDocument (baseId, {uri, id}+) para adicionar la aplicación en la base privada y el comando startDocument (baseId, documentId, interfaceId, offset, nptBaseId, nptTrigger) para disparar la aplicación.		a) El parámetro baseId en los comandos de edición NCL debe recibir "el valor del campo network_id del NIT" o "el valor del campo program_number de la PMT que se refiere a AIT".
UNBOUND (0x07)	La aplicación no es iniciada automáticamente. Ella es puesta en una lista de aplicaciones disponibles en el receptor, pudiendo ser seleccionada por un usuario para ser iniciada. La aplicación puede ser almacenada para ser iniciada en un momento posterior.	Semejante al PRESENT, salvo que la aplicación puede ser almacenada para exhibición futura.	Ver PRESENT
STORE (0x08)	El mismo comportamiento de PREFETCH	Ver PREFETCH	Ver PREFETCH

7.5 Valores *default*

Cuando el parámetro *baseId* de un *nclEditingCommand* transportado en un flujo de transporte (TS) es especificado como "nulo", él debe obligatoriamente asumir el valor del campo network_id de la NIT correspondiente (table_id 0x40).

Cuando el parámetro *baseId* de un *nclEditingCommand* procedente de un objeto NCLua ejecutado en una determinada base privada no se especifica, debe obligatoriamente asumirse el mismo valor *baseId* de esa base privada.

En un comando AddDocument, si todos los recursos de la aplicación están bajo la misma raíz, el parámetro *id* del par {uri, id} se puede omitir.

En un *nclEditingCommand*, si el campo eventNPT de un descriptor de eventos de flujo es diferente de “0”, el valor del campo eventId del descriptor de eventos de flujo debe estar listado en un objeto de eventos de flujo DSM-CC cuyo campo *id* de su STR_NPT_USE tap (campo use del tap) identifica la base de tiempo NPT (el campo *id* del tap debe obligatoriamente contener el contentId de la base de tiempo).

En un comando startDocument, si el parámetro nptBaseId es “null”, el valor del campo eventId del descriptor de eventos de flujo debe estar listado en un objeto de eventos de flujo DSM-CC cuyo campo *id* de su STR_NPT_USE tap (campo use del tap) identifica la base de tiempo NPT (el campo *id* del tap debe obligatoriamente contener el contentId de la base de tiempo).

En un comando startDocument, si el valor de nptTrigger es especificado como "nulo", el campo eventNPT del descriptor de eventos de flujo identifica el tiempo NPT que define el tiempo de inicio de la exhibición de una aplicación.

En un comando startDocument, si el parámetro interfaceId es especificado como "nulo", todos los elementos <port> del elemento <body> deben obligatoriamente ser disparados (iniciados).

En un comando startDocument, si el parámetro offset es especificado como "nulo", se debe obligatoriamente asumir "0" como valor.

Si el parámetro compositeId es especificado como "nulo", el elemento <body> del documento NCL documentId debe obligatoriamente ser considerado como la composición que se editará.

7.6 Tratamiento de las excepciones

Se recomienda que si el parámetro *baseId* de un *nclEditingCommand* transportado en un flujo TS con identificador network_id tiene un valor diferente del valor network_id, o del valor network_id.program_number (del servicio), este último sólo en el caso de receptores full-seg, se ignore ese valor.

Se recomienda que si el parámetro *baseId* de un *nclEditingCommand* procedente de un objeto NCLua en ejecución en una determinada base privada tiene un valor diferente del valor *baseId* de esa base privada, se ignore ese valor.

Los receptores que solamente implementan el perfil NCL BDTV no son obligados a tratar los siguientes *nclEditingCommands*: *pauseDocument*, *resumeDocument*, *addTransition*, *removeTransition*, *addTransitionBase* y *removeTransitionBase*.

Se recomienda que un *nclEditingCommand* que pueda causar una inconsistencia en el documento, o que se refiera a un elemento NCL inexistente o a cualquier otro identificador inexistente, se ignore. Hay sólo una excepción a esa regla – el atributo *interface* de un elemento <bind> hijo de un elemento <link> puede dejarse inconsistente, refiriéndose a un elemento <area> que se cumplimentará por un comando *addInterface* cuyo atributo *begin* tiene el valor “now”, según lo especificado en la ABNT NBR 15606-2. En ese caso, el <link> debe evaluarse obligatoriamente así que se reciba el comando *addInterface*.

Se recomienda que si en un comando startDocument el parámetro *nptBaseId* es diferente de “null” y se refiere a una base de tiempo inexistente, el comando se ignore.

8 API NCLua

8.1 Consideraciones generales

El lenguaje de script adoptado por el Ginga-NCL para implementar objetos imperativos en documentos NCL es el lenguaje *Lua* (elementos de <media> del tipo application/x-ginga-NCLua o del tipo application/x-ncl-NCLua).

Además de la biblioteca estándar Lua, los siguientes módulos deben implementarse obligatoriamente: canvas; event, settings y persistent.

8.2 Módulo *canvas*

8.2.1 Consideraciones generales

El módulo canvas ofrece una API gráfica para ser utilizada en una aplicación NCLua. Usando la API es posible dibujar líneas, rectángulos, caracteres, imágenes etc.

8.2.2 Valores *default*

En todas las operaciones **canvas: drawXXX,** la anchura de la línea debe obligatoriamente ser considerada como 1 pixel.

En la operación **canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number),** las unidades de ángulo deben obligatoriamente considerarse como grados.

En la operación **canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number),** el ángulo de 0 grado está en la coordenada Y más elevada de la elipsis y la progresión del ángulo sigue el movimiento en el sentido del reloj.

8.3 Módulo *event*

8.3.1 Consideraciones generales

Ese módulo ofrece una API para manipulación de eventos. Usando la API, el formateador NCL puede comunicarse con una aplicación NCLua de forma asíncrona.

En **event.register ([pos: number]; f: function; [class: string]; […: any]),** la posición de registro inicial es 1**.**

En **event.post** de la class='si' y type='epg', el subcampo hasInteractivity de la tabla de datos debe obligatoriamente considerar el carousel_descriptor compatible, los comandos de edición NCL y las tablas de AIT, con el fin de especificar si el evento EPG tiene (o no) una aplicación.

La función `event.post()` y el tratador registrado en `event.register()` reciben eventos como parámetros. Un evento es descrito por una tabla Lua común, donde el campo de la clase es obligatorio e identifica la clase del evento.

Con el fin de dar soporte a eventos punteros, NCLua debe incluir obligatoriamente la siguiente clase:

Clase `pointer:`

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

* el tipo puede ser "press", "release" o "move"

* X e Y se refieren a las coordenadas del acontecimiento del evento puntero

NOTA En la clase pointer, el filtro dependiente de la clase solamente puede ser *type*.

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

La clase *sms* es obligatoria solamente en la implementación del Ginga para receptores *one-seg* y debe obligatoriamente estar de acuerdo con las siguientes especificaciones:

---

clase sms:

Una aplicación NCLua envía datos, utilizando SMS, por medio del envío de eventos en la siguiente manera:

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

El campo `to` contiene el número de destino (número de teléfono o número de la cuenta). Si no se especifican, la región y los prefijos de código del país recibirán la región y los códigos del país de donde el mensaje se está enviando.

El campo `value` contiene el contenido del mensaje.

El campo `id` se puede usar para identificar el SMS que se enviará. La aplicación es responsable de definir el valor de `id` y asegurar su unicidad.

Un evento de confirmación debe obligatoriamente enviarse de vuelta a la aplicación NCLua, siguiendo el formato:

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

En el mensaje de confirmación, el campo `to` debe obligatoriamente tener el mismo valor que en el evento original enviado por la aplicación NCLua. El campo `sent` notifica si el SMS fue despachado por el dispositivo (verdadero) o no. El campo `error` es opcional. Si el valor del campo `sent` es falso, puede contener un mensaje de error detallado. Si el SMS original es enviado con el campo `id` definido, el evento de confirmación debe llegar obligatoriamente con el mismo valor de id. Así, la aplicación NCLua puede hacer una asociación entre ambos eventos y tratar con varios mensajes SMS enviados simultáneamente.

Similarmente, una aplicación NCLua se registra para recibir mensajes SMS, enviando eventos en la forma:

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

El campo `port` debe recibir obligatoriamente un número válido de puerta TCP. Para el cumplimiento de las normas GSM (3GPP TS 23.040 V6.8.1, de 2006-10), ese valor debe obligatoriamente estar en el intervalo [16000,16999].

Eventos recibidos por el tratador tienen el siguiente formato:

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

El campo `port` se define como en el tipo = 'register'. El campo `from` contiene el número del mensaje de origen (número del teléfono o número de la cuenta). El prefijo de la región y el código del país se pueden omitir si son los mismos que los del receptor. El campo `value` tiene el contenido del mensaje.

En cualquier momento, la aplicación puede solicitar parar de recibir mensajes SMS en una determinada puerta, registrando el evento:

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

El campo puerta se define como en el tipo = 'register'.

En el momento en que la presentación de la media NCLua pare, la implementación del *middleware* debe obligatoriamente asegurar que todas las puertas sean desregistradas.

NOTA 1 Se recomienda que una implementación específica de *middleware* trate cuestiones como autenticación, etc.

NOTA 2 En la clase sms, el filtro dependiente de la clase sólo puede ser *from* y *port*, en ese orden.

NOTA 3 La finalidad del número de puerta es evitar conflictos entre los mensajes comunes SMS recibidos por un usuario y los mensajes SMS que deben obligatoriamente ser tratados solamente por la aplicación.

NOTA 4 Una implementación del Ginga-NCL obligatoriamente retorna *false* inmediatamente a cada llamada para event.post () que usa una clase de evento no soportada. Se recomienda que la aplicación NCLua capture esa condición de error para verificar si el envío del SMS falló.

8.3.2 Valores *default*

En *ncl class,* los eventos pueden ser direccionados para anclas específicas o para el nudo entero. Eso es identificado por el campo *label*, que asume el nudo entero cuando está ausente.

8.3.3 Tratamiento de las excepciones

En **event.register ([pos: number]; f: function; [class: string]; […: any]),** cuando un *tratador* se registre en una posición ocupada por otro, cada posición de tratador, a partir de esa posición, debe obligatoriamente ser incrementada para dar lugar a la nueva inserción**.** Cuando un *tratador* es retirado, todas las otras posiciones de tratador, a partir de la posición de tratador eliminada, deben obligatoriamente ser disminuidas.

Bibliografia

[1] Soares, L.F.G. et al. *Nested Context Language 3.0 – Implementors Guide.* Relatorio Técnico, Departamento de Informática, PUC-Rio, nº 13/09. Rio de Janeiro. Enero de 2009. ISSN 0103-9741.