

Primeira edição
05.03.2008

Válida a partir de
05.04.2008

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital
Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações

Digital terrestrial television – Data coding and transmission specification for digital broadcasting

Part 5: Ginga – NCL for portable receivers – XML application language for application coding

Palavras-chave: Televisão digital terrestre. Middleware. Ginga. NCL. Receptores portáteis. Perfil one-seg

Descriptors: Digital terrestrial television. Middleware. Ginga. NCL. Portable receivers. one-seg profile.

ICS 33.160.01

ISBN 978-85-07-00536-0

© ABNT 2008

Todos os direitos reservados. A menos que especificado de outro modo, nenhuma parte desta publicação pode ser reproduzida ou utilizada por qualquer meio, eletrônico ou mecânico, incluindo fotocópia e microfilme, sem permissão por escrito pela ABNT.

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

Impresso no Brasil

Sumário

Página

Prefácio.....	v
Introdução	vi
1 Escopo.....	1
2 Referências normativas	1
3 Termos e definições.....	1
4 Abreviaturas.....	7
5 Arquitetura Ginga	8
5.1 Ginga <i>main modules</i>	8
5.2 Interação com o ambiente nativo.....	9
6 Objetos XHTML embutidos em apresentações NCL.....	9
6.1 NCL como linguagem cola	9
6.2 Formato de conteúdo XHTML	10
6.3 XHTML para o perfil portátil	10
6.3.1 Marcações XML.....	10
6.3.2 Folhas de estilo.....	15
7 NCL – Linguagem declarativa XML para especificação de apresentações multimídia interativas	19
7.1 Linguagens modulares e perfis de linguagens	19
7.1.1 Módulos NCL.....	19
7.1.2 Identificadores para módulos e perfis de linguagem da NCL 3.0	22
7.1.3 Informações sobre versões da NCL	24
7.2 Módulos NCL.....	24
7.3 Perfis de linguagem NCL para o SBTVD	24
7.3.1 Módulos de perfis	24
7.3.2 Esquema do perfil NCL 3.0 DTV avançado	24
7.3.3 Esquema do perfil NCL 3.0 CausalConnector	34
7.3.4 Atributos e elementos do perfil NCL 3.0 DTV Básico	34
7.3.5 Esquema do perfil NCL 3.0 DTV Básico	37
8 Objetos de mídia em apresentações NCL.....	45
8.1 Implementação modular de Ginga-NCL	45
8.2 Comportamento esperado dos exibidores de mídia.....	46
8.3 Comportamento esperado dos exibidores de mídia após instruções aplicadas aos objetos de composição.....	46
8.4 Relação entre a máquina de estado de eventos de apresentação de um nó e a máquina de estado do evento de apresentação de seu nó de composição pai.....	46
8.5 Comportamento esperado dos exibidores procedurais Lua	46
9 Transmissão de conteúdo e eventos de fluxo NCL	48
10 Objetos procedurais Lua em apresentações NCL	48
10.1 Linguagem Lua - Funções opcionais da biblioteca de Lua	48
10.2 Modelo de execução.....	49
10.3 Módulos adicionais	49
10.3.1 Módulos obrigatórios.....	49
10.3.2 Módulo <i>ncledit</i>	49
10.3.3 Módulo <i>canvas</i>	68
10.3.4 Módulo <i>event</i>	77
10.3.5 Módulo <i>settings</i>	81
10.3.6 Módulo <i>persistent</i>	82
11 Objetos procedurais Java em documentos NCL	82

12	Requisitos de codificação de mídias e métodos de transmissão referenciados em documentos NCL	84
13	Segurança	84
Anexo A	(normativo) Esquemas dos módulos NCL 3.0 usados nos perfis TVD Básico e TVD Avançado	85
A.1	Módulo <i>Structure</i> : NCL30Structure.xsd	85
A.2	Módulo <i>Layout</i> : NCL30Layout.xsd	86
A.3	Módulo <i>Media</i> : NCL30Media.xsd.....	87
A.4	Módulo <i>Context</i> : NCL30Context.xsd	88
A.5	Módulo <i>MediaContentAnchor</i> : NCL30MediaContentAnchor.xsd	89
A.6	Módulo <i>CompositeNodeInterface</i> : NC30CompositeNodeInterface.xsd.....	91
A.7	Módulo <i>PropertyAnchor</i> : NCL30PropertyAnchor.xsd	92
A.8	Módulo <i>SwitchInterface</i> : NCL30SwitchInterface.xsd.....	93
A.9	Módulo <i>Descriptor</i> : NCL30Descriptor.xsd	94
A.10	Módulo <i>Linking</i> : NCL30Linking.xsd	96
A.11	Módulo <i>ConnectorCommonPart</i> : NCL30ConnectorCommonPart.xsd	97
A.12	Módulo <i>ConnectorAssessmentExpression</i> : NCL30ConnectorAssessmentExpression.xsd	98
A.13	Módulo <i>ConnectorCausalExpression</i> : NCL30ConnectorCausalExpression.xsd	100
A.14	Módulo <i>CausalConnector</i> : NCL30CausalConnector.xsd	103
A.15	Módulo <i>ConnectorBase</i> : NCL30ConnectorBase.xsd.....	104
A.16	NCL30CausalConnectorFunctionality.xsd.....	105
A.17	Módulo <i>TestRule</i> : NCL30TestRule.xsd.....	108
A.18	Módulo <i>TestRuleUse</i> : NCL30TestRuleUse.xsd	110
A.19	Módulo <i>ContentControl</i> : NCL30ContentControl.xsd	111
A.20	Módulo <i>DescriptorControl</i> : NCL30DescriptorControl.xsd	112
A.21	Módulo <i>Timing</i> : NCL30Timing.xsd	113
A.22	Módulo <i>Import</i> : NCL30Import.xsd.....	114
A.23	Módulo <i>EntityReuse</i> : NCL30EntityReuse.xsd	115
A.24	Módulo <i>ExtendedEntityReuse</i> : NCL30ExtendedEntityReuse.xsd.....	116
A.25	Módulo <i>KeyNavigation</i> : NCL30KeyNavigation.xsd	117
A.26	Módulo <i>TransitionBase</i> : NCL30TransitionBase.xsd.....	119
A.27	Módulo <i>Animation</i> : NCL30Animation.xsd.....	120
Bibliografia	121

Prefácio

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

Os Documentos Técnicos ABNT são elaborados conforme as regras das Diretivas ABNT, Parte 2.

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

A ABNT NBR 15606-5 foi elaborada pela Comissão de Estudo Especial de Televisão Digital (ABNT/CEE-00:001.85). O Projeto circulou em Consulta Nacional conforme Edital nº 12, de 13.12.2007 a 11.02.2008, com o número de Projeto 00:001.85-006/5.

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

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

- Parte 1: Codificação de dados;
- Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações;
- Parte 3: Especificação de transmissão de dados;
- Parte 4: Ginga-J – Ambiente para a execução de aplicações procedurais;
- Parte 5: Ginga-NCL para receptores portáteis – Linguagem de aplicação XML para codificação de aplicações.

Introdução

A Associação Brasileira de Normas Técnicas (ABNT) chama atenção para o fato de que a exigência de conformidade com este documento pode envolver o uso de uma patente relativa a NCL, conforme mencionado em 5.1.

A ABNT não se posiciona a respeito de evidências, validade e escopo deste direito de patente.

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

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

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

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

Esta Norma padroniza uma linguagem de aplicação XML que permite que os autores escrevam apresentações multimídia interativas. Este componente da ABNT NBR 15606 é parte das especificações de codificação de dados para o sistema brasileiro de televisão digital terrestre (SBTVD) e compreende a especificação da linguagem utilizada pela máquina de apresentação Ginga-NCL do *middleware* SBTVD, chamado Ginga.

Por meio dessa linguagem, denominada NCL (*Nested Context Language* – Linguagem de Contextos Aninhados), um autor pode descrever o comportamento temporal de uma apresentação multimídia, associar *hyperlinks* (interação do usuário) a objetos de mídia, definir alternativas para apresentação (adaptação) e descrever o leiaute da apresentação em múltiplos dispositivos.

Esta Norma é primordialmente destinada às entidades que estão especificando terminais e/ou padrões baseados no Ginga. Também é destinada aos desenvolvedores de aplicações que utilizam as funcionalidades do Ginga e de suas API. O *middleware* Ginga tem como objetivo garantir a interoperabilidade das aplicações em diferentes implementações de plataformas que o suportam.

As aplicações Ginga são classificadas sob duas categorias, dependendo se a aplicação inicialmente processada possui conteúdo de natureza declarativa ou imperativa. Essas categorias de aplicações são chamadas de aplicações declarativas e aplicações procedurais, respectivamente. Os ambientes de aplicação são igualmente classificados em duas categorias, dependendo se eles processam aplicações declarativas ou procedurais, sendo então chamados de Ginga-NCL e Ginga-J, respectivamente.

Uma implementação unicamente Ginga-NCL é permitida para receptores portáteis, sendo, nesse caso, opcional a implementação do ambiente Ginga-J. Quando implementado apenas com o Ginga-NCL, o *middleware* Ginga dá suporte a código procedurais através da linguagem Lua.

Esta Norma não especifica a forma como os ambientes de aplicação devem ser implementados em um receptor.

Televisão digital terrestre — Codificação de dados e especificações de transmissão para radiodifusão digital

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

1 Escopo

Esta parte da ABNT NBR 15606 especifica a linguagem de aplicação XML, denominada NCL (*Nested Context Language*), a linguagem declarativa do *middleware* Ginga, a codificação e a transmissão de dados para radiodifusão digital.

2 Referências normativas

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

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

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

ECMA 262, *ECMAScript language specification*

3 Termos e definições

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

3.1

ambiente de aplicação

contexto ou ambiente de *software* no qual uma aplicação é processada

3.2

ambiente de aplicação declarativa

ambiente que suporta o processamento de aplicações declarativas

NOTA Um formatador (*user agent*) NCL é um exemplo de ambiente de aplicação declarativa.

3.3

ambiente de aplicação procedural

ambiente que suporta o processamento de aplicações procedurais

3.4

API DOM

API que define a estrutura lógica de um documento XML e a forma de acessar, ou manipular, um documento XML

NOTA Esta API é uma interface independente de plataformas e linguagens e segue o Modelo DOM (*Document Object Model*).

3.5

aplicação

informação que expressa um conjunto específico de comportamentos observáveis

3.6

aplicação declarativa

aplicação que utiliza principalmente, e como ponto de partida, informação declarativa para expressar seu comportamento

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

3.7

aplicação híbrida

aplicação híbrida declarativa ou aplicação híbrida procedural

3.8

aplicação híbrida declarativa

aplicação declarativa que contém conteúdo de objeto ativo

NOTA Um documento NCL com um Java Xlet embutido é um exemplo de aplicação híbrida declarativa.

3.9

aplicação híbrida procedural

aplicação procedural com conteúdo declarativo

NOTA Um Java Xlet que cria e causa a exibição de uma instância de documento NCL é um exemplo de aplicação híbrida procedural.

3.10

aplicação nativa

função intrínseca implementada por uma plataforma receptora

NOTA Uma exibição em *closed caption* é um exemplo de aplicação nativa.

3.11

aplicação procedural

aplicação que utiliza principalmente, e como ponto de partida, informações procedurais para expressar o seu comportamento

NOTA Um programa em Java é um exemplo de uma aplicação procedural.

3.12

armazenamento persistente

memória disponível que pode ser lida ou escrita por uma aplicação e pode ser mantida por mais tempo do que o tempo de vida da própria aplicação

NOTA O armazenamento persistente pode ser volátil ou não-volátil.

3.13

atributo

parâmetro que representa a natureza de uma propriedade

3.14

atributo de um elemento

propriedade de um elemento XML

3.15

autor

pessoa que escreve documentos NCL

3.16

canal de interatividade

canal de retorno

mecanismo de comunicação que fornece conexão entre o receptor e um servidor remoto

3.17

caractere

"letra" específica ou outro símbolo identificável

EXEMPLO "A"

3.18

carrossel de dados

método que envia qualquer conjunto de dados ciclicamente, para que esses dados possam ser obtidos, via radiodifusão, em um intervalo de tempo tão longo quanto necessário

[ISO/IEC 13818-6:2001]

3.19

codificação de caracteres

mapeamento entre um valor de entrada inteiro e o caractere textual, representado por esse mapeamento

3.20

conteúdo de objeto ativo

tipo de conteúdo que toma a forma de um programa executável

NOTA Um Xlet Java compilado é um exemplo de conteúdo de objeto ativo.

3.21

conteúdo NCL

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

3.22

digital storage media command and control

DSM-CC

método de controle que fornece acesso a um arquivo ou fluxo em serviços digitais interativos

[ISO/IEC 13818-6:2001]

3.23

document type definition

DTD

declaração que descreve um tipo de documento XML

3.24

ECMAScript

linguagem de programação definida na ECMA 262

3.25

elemento

unidade de estruturação do documento delimitada por tags

NOTA Um elemento é usualmente delimitado por uma *tag* inicial e uma *tag* final, exceto um elemento vazio, que é delimitado por uma *tag* de elemento vazio.

3.26

elemento *property*

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

3.27

entidade da aplicação

unidade de informação que expressa alguma parte de uma aplicação

3.28

evento

ocorrência no tempo que pode ser instantânea ou ter duração mensurável

3.29

exibidor de mídia

media player

componente identificável de um ambiente de aplicação que decodifica ou executa um tipo específico de conteúdo

3.30

eXtensible HTML

XHTML

versão estendida do HTML como aplicação XML

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

3.31

ferramenta de autoria

ferramenta para auxiliar os autores a criar documentos NCL

3.32

fonte

mecanismo que permite a renderização específica de um caractere

EXEMPLO Tiresias, 12 pontos.

NOTA Na prática, um formato de fonte incorpora aspectos da codificação de um caractere.

3.33

formatador NCL

componente de software responsável por receber a especificação de um documento NCL e controlar sua apresentação, tentando garantir que os relacionamentos entre os objetos de mídia especificados pelo autor sejam respeitados

NOTA Renderizador (*renderer*) de documentos, agente do usuário (*user agent*) e exibidor são outros nomes usados com o mesmo significado do formatador de documentos.

3.34

fluxo de transporte

refere-se à sintaxe do fluxo de transporte MPEG-2 para empacotamento e multiplexação de vídeo, áudio e sinais de dados em sistemas de radiodifusão digital

3.35

fluxo elementar

elementary stream

ES

fluxo básico que contém dados de vídeo, áudio, ou dados privados

NOTA Um único fluxo elementar é transportado em uma seqüência de pacotes PES com um e apenas um identificador (*stream_id*).

3.36

gerenciador de aplicações

entidade responsável por administrar o ciclo de vida das aplicações e que gerencia as aplicações rodando tanto na máquina de apresentação quanto na máquina de execução

3.37

identificador de pacote

PID

valor inteiro único, utilizado para associar os fluxos elementares de um programa, tanto em um fluxo de transporte único como em multiprograma

3.38

informação de serviço

SI

dados que descrevem programas e serviços

3.39

informações específicas do programa

program specific information

PSI

dados normativos necessários para demultiplexar os fluxos de transporte e regenerar os programas

3.40

interface de programação da aplicação

API

bibliotecas de software que oferecem acesso uniforme aos serviços do sistema

3.41

linguagem de marcação

formalismo que descreve uma classe de documentos que empregam marcação para delinear a estrutura, aparência ou outros aspectos do documento

3.42

linguagem de *script*

linguagem utilizada para descrever um conteúdo de objeto ativo embutido em documentos NCL e em documentos HTML

3.43

localizador

identificador que fornece uma referência a uma aplicação ou recurso

3.44

máquina de apresentação

subsistema em um receptor que analisa e apresenta aplicações declarativas, com conteúdos como áudio, vídeo, gráficos e texto, baseadas em regras definidas na máquina de apresentação

NOTA Uma máquina de apresentação é responsável pelo controle do comportamento da apresentação e por iniciar outros processos em resposta a entradas do usuário e outros eventos.

EXEMPLO Navegador HTML e formatador NCL.

3.45

máquina de execução

subsistema em um receptor que avalia e executa aplicações procedurais, consistindo em instruções em linguagem de computador, conteúdo de mídia associados e outros dados

NOTA Uma máquina de execução pode ser implementada com um sistema operacional, compiladores de linguagem de computador, interpretadores e interfaces de programação de aplicações (API), que uma aplicação procedural pode utilizar para apresentar conteúdo audiovisual, interagir com o usuário ou executar outras tarefas que não sejam evidentes ao usuário.

EXEMPLO Ambiente de *software* JavaTV, utilizando linguagem de programação Java e interpretador *bytecode*, API JavaTV e máquina virtual Java para execução do programa.

**3.46
método**

função associada a um objeto que tem permissão de manipular os dados do objeto

**3.47
nó NCL**

elemento <media>, <context>, <body> ou <switch> de NCL

**3.48
normal play time
NPT**

coordenada temporal absoluta que representa a posição em um fluxo

**3.49
objeto de mídia**

coleção de pedaços de dados identificados por nome, que pode representar um conteúdo de mídia ou um programa escrito em linguagem específica

**3.50
perfil**

especificação de uma classe de capacidades, oferecendo diferentes níveis de funcionalidades em um receptor

**3.51
perfil *one-seg***

caracteriza o serviço que pode ser recebido por um sintonizador de banda estreita (430 KHz), e portanto com economia no consumo de bateria

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

**3.52
perfil *full-seg***

caracteriza o serviço que precisa necessariamente de um demodulador de faixa larga (5,7 MHz) para ser recebido

NOTA Dependendo das configurações de transmissão e de funcionalidade específicas do receptor, pode ser recebido em movimento ou apenas por receptores fixos, porém sem o benefício da economia de energia. A resolução do vídeo transmitido pode ser ou não de alta definição.

**3.53
*plug-in***

conjunto de funcionalidades que pode ser adicionado a uma plataforma genérica para fornecer funcionalidade adicional

**3.54
plataforma receptora
plataforma**

hardware, sistema operacional e bibliotecas de software nativas do receptor, escolhidos pelo fabricante

**3.55
recurso**

objeto de dados ou um serviço da rede que é identificado univocamente

3.56**sistema de arquivos local**

sistema de arquivos fornecido pela plataforma receptora local

3.57**tempo de vida de uma aplicação**

período de tempo entre o momento em que uma aplicação é carregada e o momento em que ela é destruída

3.58***uniform resource identifier*****URI**

método de endereçamento que permite o acesso a objetos em uma rede

3.59***user agent***

qualquer programa que interpreta um documento NCL

NOTA Um *user agent* pode exibir um documento tentando garantir que as relações especificadas pelo autor entre objetos de mídia sejam respeitadas, pronunciá-lo em áudio sintetizado, convertê-lo para um outro formato etc.

3.60**usuário**

pessoa que interage com um formatador para visualizar, ouvir ou utilizar de outra forma um documento NCL

3.61**usuário final**

indivíduo que opera ou interage com um receptor

4 Abreviaturas

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

API	<i>Application Programming Interface</i>
BML	<i>Broadcast Markup Language</i>
CLUT	<i>Color Look-up Table</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
DSM-CC	<i>Digital Storage Media Command and Control</i>
DTD	<i>Document Type Definition</i>
DTV	<i>Digital Television</i>
DVB	<i>Digital Video Broadcasting</i>
GIF	<i>Graphics Interchange Format</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
JPEG	<i>Joint Photographic Expert Group</i>
MIME	<i>Multipurpose Internet Mail Extensions</i>
MNG	<i>Multiple Network Graphics</i>
MPEG	<i>Moving Picture Expert Group</i>
NCL	<i>Nested Context Language</i>
NCM	<i>Nested Context Model</i>
NPT	<i>Normal Play Time</i>
OS	<i>Operating System</i>
PAT	<i>Program Association Table</i>

PES	<i>Packetized Elementary Stream</i>
PID	<i>Packet Identifier</i>
PMT	<i>Program Map Table</i>
PNG	<i>Portable Network Graphics</i>
PSI	<i>Program Specific Information</i>
SBTVD	<i>Sistema Brasileiro de Televisão Digital Terrestre</i>
SMIL	<i>Synchronized Multimedia Integration Language</i>
TS	<i>Transport Stream</i>
UCS	<i>Universal (Coded) Character Set</i>
URI	<i>Universal Resource Identifier</i>
URL	<i>Universal Resource Locator</i>
XHTML	<i>eXtensible HTML</i>
XML	<i>Extensible Markup Language</i>
W3C	<i>World-Wide Web Consortium</i>

5 Arquitetura Ginga

5.1 Ginga *main modules*

O universo das aplicações Ginga pode ser particionado em um conjunto de aplicações declarativas e um conjunto de aplicações procedurais. Uma aplicação declarativa é aquela onde o tipo do conteúdo da entidade inicial é declarativo. Por outro lado, uma aplicação procedural é aquela cujo tipo do conteúdo da entidade inicial é procedural. Uma aplicação declarativa pura é aquela na qual o conteúdo de todas as entidades é do tipo declarativo, e uma aplicação procedural pura é aquela na qual o conteúdo de todas as entidades é do tipo procedural. Uma aplicação híbrida é aquela cujo conjunto de entidades possui tanto conteúdo do tipo declarativo quanto procedural. Uma aplicação Ginga não necessita ser puramente declarativa ou procedural.

Em particular, as aplicações declarativas freqüentemente fazem uso de *scripts*, cujo conteúdo é de natureza procedural. Além disso, uma aplicação declarativa pode fazer referência a um código Java TV Xlet embutido. Da mesma forma, uma aplicação procedural pode fazer referência a uma aplicação declarativa, contendo, por exemplo, conteúdo gráfico, ou pode construir e iniciar a apresentação de aplicações com conteúdo declarativo. Portanto, ambos os tipos de aplicação Ginga podem utilizar as facilidades dos ambientes de aplicação declarativo e procedural.

Ginga-NCL, ambiente obrigatório para receptores portáteis, é um subsistema lógico do sistema Ginga, responsável pelo processamento de documentos NCL. Um componente-chave do Ginga-NCL é a máquina de interpretação do conteúdo declarativo (formatador NCL). Outros módulos importantes são o exibidor (*user agent*) XHTML e a máquina de apresentação Lua, que é responsável pela interpretação dos *scripts* Lua (ver ABNT NBR 15606-2:2007, Anexo B).

NOTA NCL é marca registrada e sua especificação é propriedade intelectual da PUC-Rio (INPI Departamento de Transferência Tecnológica – Nº 0007162-5; 20/12/2005).

Ginga-J, ambiente opcional para receptores portáteis, é um subsistema lógico do sistema Ginga, responsável pelo processamento de conteúdos ativos. Um componente-chave do ambiente de aplicação procedural é a máquina de execução do conteúdo procedural, composta por uma máquina virtual Java.

Decodificadores de conteúdo comuns servem tanto às aplicações procedurais quanto às declarativas que necessitam decodificar e apresentar tipos comuns de conteúdo como PNG, JPEG, MPEG e outros formatos. O núcleo comum Ginga (Ginga *Common Core*) é composto pelos decodificadores de conteúdo comuns e por procedimentos para obter conteúdos transportados em fluxos de transporte (*transport streams*) MPEG-2 e através do canal de interatividade. O núcleo comum Ginga também deve obrigatoriamente suportar o modelo conceitual de exibição, conforme descrito na ABNT NBR 15606-1.

A arquitetura (ver Figura 1) e as facilidades Ginga foram projetadas para serem aplicadas a sistemas de radiodifusão e receptores terrestres de radiodifusão. Adicionalmente, a mesma arquitetura e facilidades podem ser aplicadas a sistemas que utilizam outros mecanismos de transporte de dados (como sistemas de televisão via satélite ou a cabo).

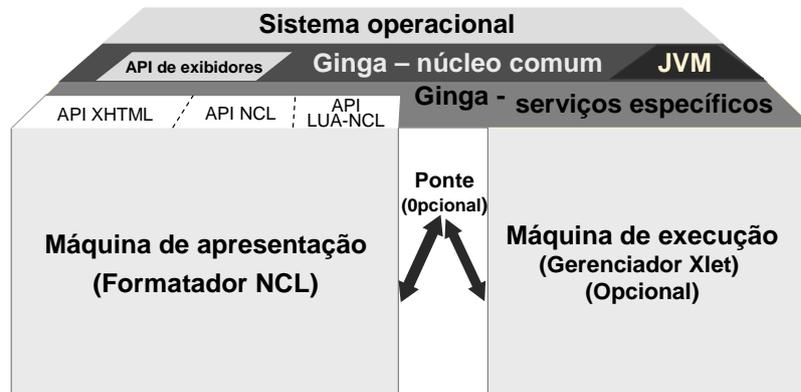


Figura 1 — Arquitetura Ginga

5.2 Interação com o ambiente nativo

Em geral, o Ginga é alheio a quaisquer aplicações nativas que podem também optar por utilizar o plano gráfico. Isso inclui, mas não se limita a aplicações como: *closed caption*, mensagens do sistema de acesso condicional (CA), menus do receptor e guias de programação nativos.

As aplicações nativas podem ter prioridade sobre as aplicações Ginga. O *closed caption* e as mensagens de emergência devem obrigatoriamente ter prioridade sobre o sistema Ginga.

Algumas aplicações nativas, como o *closed caption*, representam um caso especial no qual a aplicação nativa pode estar ativa por longos períodos juntamente com as aplicações Ginga.

6 Objetos XHTML embutidos em apresentações NCL

6.1 NCL como linguagem cola

Diferentemente de XHTML ou HTML, NCL estabelece uma separação bem definida entre o conteúdo e a estrutura de um documento (ou aplicativo), provendo um controle não invasivo da ligação entre o conteúdo e sua apresentação e leiaute.

O foco da linguagem declarativa NCL é mais amplo do que o oferecido pela XHTML. A sincronização espaço-temporal, definida genericamente pelos *links* NCL; adaptabilidade, definida pelos elementos *switch* e *descriptor switch* da NCL; e suporte a múltiplos dispositivos de exibição, definidos por regiões NCL, é o foco dessa linguagem declarativa. A interação do usuário é tratada apenas como caso particular de sincronização temporal.

Como a NCL tem uma separação mais acurada entre o conteúdo e a estrutura, ela não define nenhuma mídia em si. Ao contrário, ela define a cola que prende as mídias em apresentações multimídia.

Um documento NCL apenas define como os objetos de mídia são estruturados e relacionados, no tempo e espaço. Como uma linguagem de cola, ela não restringe ou prescreve os tipos de conteúdo dos objetos de mídia. Nesse sentido, pode-se ter objetos de imagem (GIF, JPEG etc.), de vídeo (MPEG, MOV etc.), de áudio (MP3, WMA etc.), de texto (TXT, PDF etc.), de execução (Xlet, Lua etc.), entre outros, como objetos de mídia NCL. Quais objetos de mídia são suportados depende dos exibidores de mídia que estão acoplados ao formatador NCL (exibidor NCL). Um desses exibidores é o decodificador/exibidor MPEG-4, normalmente implementado em *hardware* no receptor de televisão digital. Dessa forma, o vídeo e o áudio MPEG-4 principal são tratados como todos os demais objetos de mídia que podem estar relacionados utilizando NCL.

Outro objeto de mídia NCL que deve obrigatoriamente ser suportado é o objeto de mídia baseado em XHTML. A NCL não substitui, mas embute documentos (ou objetos) baseados em XHTML. Como acontece com outros objetos de mídia, qual linguagem baseada em XHTML tem suporte em um formatador NCL é uma escolha de implementação e, portanto, depende de qual navegador XHTML, incorporado no formatador NCL, atua como exibidor dessa mídia.

Embora um navegador XHTML deva obrigatoriamente ser suportado, recomenda-se que a utilização de elementos XHTML para definir relacionamentos (inclusive *links* XHTML) seja evitada na autoria de documentos NCL. Recomenda-se que a autoria baseada na estrutura seja priorizada por razões conhecidas e amplamente divulgadas na literatura.

Durante a exibição do conteúdo de objetos de mídia são gerados vários eventos. Alguns exemplos são a apresentação de parte do conteúdo de um objeto de mídia, a seleção de parte do conteúdo de um objeto etc. Os eventos podem gerar ações sobre outros objetos de mídia, como iniciar ou terminar suas apresentações. Portanto, os eventos devem obrigatoriamente ser relatados pelos exibidores de mídia ao formatador NCL que, por sua vez, pode gerar ações a serem aplicadas a esses ou outros exibidores. Ginga-NCL define a API (ver Seção 8) de um adaptador com o objetivo de padronizar a interface entre o formatador Ginga-NCL e cada exibidor específico.

Para que qualquer exibidor de mídia, em particular um navegador XHTML, seja acoplado ao formatador Ginga-NCL, ele deve obrigatoriamente suportar a API dos adaptadores. Assim, para alguns exibidores de mídia, inclusive navegadores XHTML, um módulo adaptador pode ser necessário para que a integração seja alcançada.

Para edição ao vivo, o Ginga-NCL também define eventos de fluxo NCL para oferecer suporte aos eventos gerados ao vivo sobre fluxos de mídia, em particular sobre o fluxo de vídeo do programa principal. Esses eventos são uma generalização do mesmo conceito encontrado em outras linguagens, como, por exemplo, os *b-events* de BML. Embora um navegador XHTML deva obrigatoriamente ser suportado, recomenda-se que a utilização de elementos XHTML para definir relacionamentos (inclusive eventos de fluxo) seja evitada quando da criação de documentos NCL, pela mesma razão, isto é, recomenda-se que a autoria baseada na estrutura seja priorizada por razões conhecidas e amplamente divulgadas na literatura.

6.2 Formato de conteúdo XHTML

Formatos comuns de conteúdo devem obrigatoriamente ser adotados para a produção e intercâmbio de conteúdo multimídia, como definido na ABNT NBR 15606-1. Além disso, no ambiente de aplicação declarativa também é exigida a especificação de formatos comuns de conteúdos XHTML para as aplicações de televisão interativa.

Assim, esta Norma também especifica os elementos obrigatórios dos objetos de mídia XHTML embutidos em aplicações NCL, bem como as propriedades de folhas de estilo obrigatórias.

6.3 XHTML para o perfil portátil

6.3.1 Marcações XML

NOTA Objetos de mídia NCL baseados em XHTML seguem a recomendação W3C “*Modularization of XHTML*”.

As coleções de atributo XHTML são definidas de acordo com a Tabela 1. As marcações XML que devem obrigatoriamente ser suportadas por qualquer implementação, são listadas na Tabela 2.

Tabela 1 — Coleções de atributos

Nome da coleção	Atributos na coleção	Condição do atributo
<i>Core</i>	class (NMTOKENS)	Requerido
	Id (ID),	Requerido
	title (CDATA)	-
I18N	xml:lang (CDATA)	Requerido (default)
<i>Events</i>	onclick (Script)	-
	ondblclick (Script)	-
	onmousedown (Script)	-
	onmouseup (Script)	-
	onmouseover (Script)	-
	onmousemove (Script)	-
	onmouseout (Script)	-
	onkeypress (Script)	-
	onkeydown (Script)	-
	onkeyup (Script)	-
<i>Style</i>	style (CDATA)	Requerido
<i>Common</i>	Core + Events + I18N + Style	

Tabela 2 — Elementos de marcação XML obrigatórios

Módulo		Elemento	Condição do elemento	Atributo	Condição do atributo
Core	Structure	body	Requerido	%Common.attrib	
				%Core.attrib	Requerido
				%l18n.attrib	Requerido
				%Events.attrib	–
		head	Requerido	%l18n.attrib	Requerido
				profile	–
	html	Requerido			
	title	Requerido	%l18n.attrib	Requerido	
	Text	abbr	–		
		acronym	–		
		address	–		
		blockquote	–		
		br	Requerido	%Core.attrib	Requerido
		cite	–		
		code	–		
		dfn	–		
		div	Requerido	%Common.attrib	Requerido
		em	–		
		h1	Requerido	%Common.attrib	Requerido
		h2	Requerido	%Common.attrib	Requerido
		h3	Requerido	%Common.attrib	Requerido
		h4	Requerido	%Common.attrib	Requerido
		h5	Requerido	%Common.attrib	Requerido
		h6	Requerido	%Common.attrib	Requerido
		kbd	–		
		p	Requerido	%Common.attrib	Requerido
		pre	–		
		q	–		
	samp	–			
	span	Requerido	%Common.attrib	Requerido	
	strong	–			
	var	–			
	Hypertext	a	Requerido	%Common.attrib	Requerido
				accesskey	Requerido
				charset	Requerido
				href	Requerido
hreflang				–	
rel				–	
rev				–	
tabindex				–	
type	–				
List	dl	–			
	dt	–			
	dd	–			
	ol	–			
	ul	–			
	li	–			

Tabela 2 (continuação)

Módulo		Elemento	Condição do elemento	Atributo	Condição do atributo	
Applet		applet	–			
		param	–			
Text extension	Presentation	b	–			
		big	–			
		hr	–			
		i	–			
		small	–			
		sub	–			
		sup	–			
		tt	–			
	Edit	del	–			
		ins	–			
Bi-directional text	bdo	–				
Forms	Basic forms	form	–			
		input	–			
		label	–			
		select	–			
		option	–			
		textarea	–			
	Forms	Forms	form	Requerido	%Common.attrib	Requerido
					action	Requerido
					method	Requerido
					enctype	Requerido
					accept-charset	Requerido
					accept	Requerido
			input	Requerido	name	Requerido
					%Common.attrib	Requerido
					accesskey	Requerido
					checked	–
					disabled	Requerido
					readonly	Requerido
					maxlength	Requerido
					alt	–
					size	Requerido
					src	–
					tabindex	–
					accept	–
	type	Requerido				
	value	Requerido				
	select	Requerido				
option	Requerido					
textarea	Requerido					
button	–					
fieldset	–					
label	–					
legend	–					
optgroup	–					

Tabela 2 (continuação)

Módulo		Elemento	Condição do elemento	Atributo	Condição do atributo
Table	Basic tables	caption	–		
		table	Requerido		
		td	Requerido		
		th	–		
		tr	Requerido		
	Tables	caption	–		
		table	–		
		td	–		
		th	–		
		tr	–		
		col	–		
		colgroup	–		
		tbody	–		
		thead	–		
tfoot	–				
Image		img	–		
Client side map		a&	–		
		area	–		
		img&	–		
		input&	–		
		map	–		
		object&	–		
Server side image map		img&	–		
		Input&	–		
Object		object	Requerido	%Common.attrib	Requerido
				archive	–
				classid	–
				codebase	–
				codetype	–
				data	Requerido
				declare	–
				height	Requerido
				name	–
				standby	–
		tabindex	–		
type	Requerido				
width	Requerido				
		param	–		
Frames		frameset	–		
		frame	–		
		noframe	–		
Target		a&	–		
		area&	–		
		base&	–		
		link&	–		
		form&	–		
IFrame		iframe	–		

Tabela 2 (continuação)

Módulo	Elemento	Condição do elemento	Atributo	Condição do atributo
Intrinsic events	a&	–		
	area&	–		
	frameset&	–		
	form&	–		
	body&	–		
	label&	–		
	input&	–		
	select&	–		
	textarea&	–		
	button&	–		
Metainformation	meta	Requerido	%l18n.attrib	–
			http-equiv	–
			name	Requerido
			content	Requerido
			scheme	–
Scripting	noscript			
	script	–	charset	
			type	
			src	
			defer	
Stylesheet	style	Requerido	%l18n.attrib	Requerido
			id	–
			type	Requerido
			media	Requerido
			title	–
Style attribute		Requerido		
Link	link	Requerido		
Base	base	–		

6.3.2 Folhas de estilo

As propriedades de folhas de estilo que devem obrigatoriamente ser suportadas por qualquer implementação estão listadas na Tabela 3.

Tabela 3 — Propriedades de folhas de estilo CSS 2 obrigatórias

Propriedade	Condição da propriedade
Value assignment/Inheritance	
@import	–
!important	–
Media type	
@media	Requerido
box model	
margin-top	–
margin-right	–
margin-bottom	–
margin-left	–
margin	Requerido
padding-top	Requerido
padding-right	Requerido
padding-bottom	Requerido
padding-left	Requerido
padding	Requerido
border-top-width	–
border-right-width	–
border-bottom-width	–
border-left-width	–
border-width	Requerido
border-top-color	–
border-right-color	–
border-bottom-color	–
border-left-color	–
border-color	Requerido
border-top-style	–
border-right-style	–
border-bottom-style	–
border-left-style	–
border-style	Requerido
border-top	–
border-right	–
border-bottom	–
border-left	–
border	Requerido
Visual formatting model	
position	Requerido
left	Requerido
top	Requerido
width	Requerido
height	Requerido
z-index	Requerido
line-height	Requerido
vertical-align	–
display	Requerido
bottom	–
right	–
float	–
clear	–
direction	–

Tabela 3 (continuação)

Propriedade	Condição da propriedade
unicode-bidi	–
min-width	–
max-width	–
min-height	–
max-height	–
Other visual effects	
visibility	Requerido
overflow	Requerido
clip	–
Generated content/Auto numbering/List	
content	–
quotes	–
counter-reset	–
counter-increment	–
marker-offset	–
list-style-type	–
list-style-image	–
list-style-position	–
list-style	–
Page media	
"@page"	–
size	–
marks	–
page-break-before	–
page-break-after	–
page-break-inside	–
page	–
orphans	–
widows	–
Background	
background	–
background-color	–
background-image	Requerido
background-repeat	Requerido
background-position	–
background-attachment	–
Font	
color	Requerido
font-family	Requerido
font-style	Requerido
font-size	Requerido
font-variant	Requerido
font-weight	Requerido
font	Requerido
font-stretch	–
font-adjust	–
Text	
text-indent	–
text-align	Requerido
text-decoration	–

Tabela 3 (continuação)

Propriedade	Condição da propriedade
text-shadow	–
letter-spacing	Requerido
word-spacing	–
text-transform	–
white-space	Requerido
Pseudo class/ Pseudo element	
:link	–
:visited	–
:active	Requerido
:hover	–
:focus	Requerido
:lang	–
:first-child	–
:first-line	–
:first-letter	–
:before	–
:after	–
Table	
caption-side	–
border-collapse	–
border-spacing	–
table-layout	–
empty-cells	–
speech-header	–
User interface	
outline-color	–
outline-width	–
outline-style	–
outline	–
cursor	–
Voice style sheet	
volume	–
speak	–
pause-before	–
pause-after	–
pause	–
cue-before	–
cue-after	–
cue	–
play-during	–
azimuth	–
elevation	–
speech-rate	–
voice-family	–
pitch	–
pitch-range	–
stress	–
richness	–
speak-punctuation	–
peak-numeral	–

Tabela 3 (continuação)

Propriedade	Condição da propriedade
Extended property	
clut	–
color-index	–
background-color-index	–
border-color-index	–
border-top-color-index	–
border-right-color-index	–
border-bottom-color-index	–
border-left-color-index	–
outline-color-index	–
resolution	–
display-aspect-ratio	–
grayscale-color-index	–
nav-index	–
nav-up	–
nav-down	–
nav-left	–
nav-right	–
used-key-list	–

As seguintes restrições devem obrigatoriamente ser aplicadas às propriedades de exibição:

- somente elementos de bloco podem ser aplicados para <p>, <div>, <body>, <input> e <object>;
- somente valores definidos no próprio elemento HTML podem ser aplicados para
, <a> e .

Além disso, as seguintes restrições devem obrigatoriamente ser aplicadas às propriedades de posição:

- somente valores absolutos podem ser aplicados para <p>, <div>, <input> e <object>;
- somente valores estáticos podem ser aplicados para
, e <a>.

Os seletores CSS que devem obrigatoriamente ser suportados por qualquer implementação são os seguintes:

- *universal*;
- *type*;
- *class*;
- *id*.

7 NCL – Linguagem declarativa XML para especificação de apresentações multimídia interativas

7.1 Linguagens modulares e perfis de linguagens

7.1.1 Módulos NCL

A abordagem modular tem sido utilizada em várias linguagens recomendadas pelo W3C.

Módulos são coleções de elementos, atributos e valores de atributos XML semanticamente relacionados que representam uma unidade de funcionalidade. Módulos são definidos em conjuntos coerentes. Essa coerência é expressa por meio da associação de um mesmo *namespace* aos elementos desses módulos.

NOTA *Namespaces* são discutidos em *Namespaces in XML:1999*.

Um perfil de linguagem é uma combinação de módulos. Os módulos são atômicos, isto é, não podem ser subdivididos quando incluídos em um perfil de linguagem. Além disso, a especificação de um módulo pode incluir um conjunto de requisitos para integração, com o qual os perfis de linguagem, que incluem o módulo, devem obrigatoriamente ser compatíveis.

NCL foi especificada de forma modular, permitindo a combinação de seus módulos em perfis de linguagem. Cada perfil pode agrupar um subconjunto de módulos NCL, permitindo a criação de linguagens voltadas para as necessidades específicas dos usuários. Além disso, os módulos e perfis NCL podem ser combinados com módulos definidos em outras linguagens, permitindo a incorporação de características da NCL naquelas linguagens e vice-versa.

Normalmente, há um perfil de linguagem que incorpora quase todos os módulos associados a um único *namespace*. Esse é o caso do perfil *Linguagem NCL*.

Outros perfis de linguagem podem ser especificados como subconjuntos de um perfil maior ou incorporar uma combinação de módulos associados a diferentes *namespaces*. Um exemplo desse último caso é a utilização de módulos SMIL (por exemplo, o módulo *SMIL Transition*) em perfis NCL. Exemplos do primeiro caso são os perfis TVD Básico (perfil BDTV) e TVD Avançado (perfil EDTV) da NCL.

Subconjuntos dos módulos do perfil Linguagem NCL utilizados na definição dos perfis TVD Básico e TVD Avançado são definidos para ajustar a linguagem às características do ambiente de radiodifusão de televisão, com seus vários dispositivos de apresentação: aparelho de televisão, dispositivos móveis etc.

NOTA Uma abordagem similar também é encontrada em outras linguagens (SMIL 2.1 Specification:2005 e XHTML 1.0:2002).

O principal objetivo da conformidade com perfis de linguagem é aumentar a interoperabilidade. Os módulos obrigatórios são definidos de forma que qualquer documento, especificado em conformidade com um perfil de linguagem, resulte em uma apresentação razoável quando apresentado em um perfil distinto daquele para o qual foi especificado. O formatador de documentos, suportando o conjunto de módulos obrigatórios, ignoraria todos os outros elementos e atributos desconhecidos.

NOTA Renderizador de documentos, agente do usuário e exibidor são outros nomes atribuídos ao formatador de documentos.

O perfil BDTV é o perfil mínimo exigido para dispositivos portáteis. Alternativamente, pode-se usar o perfil EDTV.

A versão NCL 3.0 revisa as funcionalidades contidas na NCL 2.3 (NCL Main Profile: 2005) e é particionada em 13 áreas funcionais, que são novamente particionadas em módulos. A partir das 13 áreas funcionais, 12 são utilizadas para definir os perfis TVD Avançado e TVD Básico. Além das 12 áreas funcionais de NCL 3.0, 2 áreas funcionais foram importadas de SMIL 2.0 para compor o perfil TVD Avançado. As 14 áreas funcionais utilizadas e seus módulos correspondentes são:

1) *Structure*

Módulo *Structure*

2) *Layout*

Módulo *Layout*

3) *Components*

Módulo *Media*

Módulo *Context*

4) Interfaces

Módulo *MediaContentAnchor*

Módulo *CompositeNodeInterface*

Módulo *PropertyAnchor*

Módulo *SwitchInterface*

5) *Presentation Specification*

Módulo *Descriptor*

6) *Linking*

Módulo *Linking*

7) *Connectors*

Módulo *ConnectorCommonPart*

Módulo *ConnectorAssessmentExpression*

Módulo *ConnectorCausalExpression*

Módulo *ConnectorTransitionAssessment*

Módulo *CausalConnector*

Módulo *CausalConnectorFunctionality*

Módulo *ConnectorBase*

8) *Presentation Control*

Módulo *TestRule*

Módulo *TestRuleUse*

Módulo *ContentControl*

Módulo *DescriptorControl*

9) *Timing*

Módulo *Timing*

10) *Reuse*

Módulo *Import*

Módulo *EntityReuse*

Módulo *ExtendedEntityReuse*

11) *Navigational Key*

Módulo *KeyNavigation*

12) *Animation*

Módulo *Animation*

13) *SMIL Transition Effects*

Módulo *TransitionBase*

Módulo *BasicTransition*

Módulo *TransitionModifiers*

NOTA O módulo *TransitionBase* é um módulo definido pela NCL 3.0; não existe na SMIL 2.0.

14) *SMIL Meta-Information*

Módulo *Metainformation*

7.1.2 Identificadores para módulos e perfis de linguagem da NCL 3.0

Recomenda-se que cada perfil NCL declare explicitamente o URI do *namespace* que será usado para identificá-lo.

Documentos criados em perfis de linguagem que incluem o módulo *Structure* de NCL podem ser associados com o tipo *MIME* "application/x-ncl+xml". Os documentos utilizando o tipo *MIME* "application/x-ncl+xml" devem obrigatoriamente estar em conformidade com a linguagem hospedeira.

Os identificadores de *namespace* XML para o conjunto completo de módulos, elementos e atributos NCL 3.0 estão contidos no seguinte *namespace*: <http://www.ncl.org.br/NCL3.0/>

Cada módulo NCL possui um identificador único a ele associado. Os identificadores dos módulos NCL 3.0 devem obrigatoriamente estar de acordo com a Tabela 4.

Cada módulo SMIL possui um identificador único a ele associado. Os identificadores dos módulos SMIL 2.0 devem obrigatoriamente estar de acordo com a Tabela 5.

Módulos também podem ser identificados coletivamente. As seguintes coleções de módulos são definidas:

- módulos utilizados pelo perfil Linguagem NCL 3.0: <http://www.ncl.org.br/NCL3.0/LanguageProfile>
- módulos utilizados pelo perfil Conector Causal NCL 3.0:
<http://www.ncl.org.br/NCL3.0/CausalConnectorProfile>
- módulos utilizados pelo perfil DTV Avançado NCL 3.0: <http://www.ncl.org.br/NCL3.0/EDTVProfile>
- módulos utilizados pelo perfil DTV Básico NCL 3.0: <http://www.ncl.org.br/NCL3.0/BDTVProfile>

Tabela 4 — Identificadores dos módulos de NCL 3.0

Módulos	Identificadores
Animation	http://www.ncl.org.br/NCL3.0/Animation
CompositeNodeInterface	http://www.ncl.org.br/NCL3.0/CompositeNodeInterface
CausalConnector	http://www.ncl.org.br/NCL3.0/CausalConnector
CausalConnectorFunctionality	http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality
ConnectorCausalExpression	http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression
ConnectorAssessmentExpression	http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression
ConnectorBase	http://www.ncl.org.br/NCL3.0/ConnectorBase
ConnectorCommonPart	http://www.ncl.org.br/NCL3.0/ConnectorCommonPart
ConnectorTransitionAssessment	http://www.ncl.org.br/NCL3.0/ConnectorTransitionAssessment
ContentControl	http://www.ncl.org.br/NCL3.0/ContentControl
Context	http://www.ncl.org.br/NCL3.0/Context
Descriptor	http://www.ncl.org.br/NCL3.0/Descriptor
DescriptorControl	http://www.ncl.org.br/NCL3.0/DescriptorControl
EntityReuse	http://www.ncl.org.br/NCL3.0/EntityReuse
ExtendedEntityReuse	http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse
Import	http://www.ncl.org.br/NCL3.0/Import
Layout	http://www.ncl.org.br/NCL3.0/Layout
Linking	http://www.ncl.org.br/NCL3.0/Linking
Media	http://www.ncl.org.br/NCL3.0/Media
MediaContentAnchor	http://www.ncl.org.br/NCL3.0/MediaContentAnchor
KeyNavigation	http://www.ncl.org.br/NCL3.0/KeyNavigation
PropertyAnchor	http://www.ncl.org.br/NCL3.0/PropertyAnchor
Structure	http://www.ncl.org.br/NCL3.0/Structure
SwitchInterface	http://www.ncl.org.br/NCL3.0/SwitchInterface
TestRule	http://www.ncl.org.br/NCL3.0/TestRule
TestRuleUse	http://www.ncl.org.br/NCL3.0/TestRuleUse
Timing	http://www.ncl.org.br/NCL3.0/Timing
TransitionBase	http://www.ncl.org.br/NCL3.0/TransitionBase

Tabela 5 — Identificadores dos módulos de SMIL 2.0 utilizados em perfis NCL

Módulos	Identificadores
BasicTransitions	http://www.w3.org/2001/SMIL20/BasicTransitions
Metainformation	http://www.w3.org/2001/SMIL20/Metainformation

7.1.3 Informações sobre versões da NCL

As seguintes instruções de processamento devem obrigatoriamente ser incluídas em um documento NCL (elas identificam documentos NCL que contenham apenas os elementos definidos nesta Norma, e a versão NCL com a qual o documento está de acordo):

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<ncl id="qualquer string" xmlns="http://www.ncl.org.br/NCL3.0/profileName">
```

O atributo *id* do elemento <ncl> pode receber qualquer cadeia de caracteres como valor.

O número de versão de uma especificação NCL consiste em um número principal e outro secundário, separados por um ponto. Os números são representados como uma cadeia de caracteres formada por números decimais, na qual os zeros à esquerda são suprimidos. O número de versão inicial do padrão é 3.0.

Novas versões da NCL devem obrigatoriamente ser publicadas de acordo com a seguinte política de versionamento:

- se os receptores compatíveis com versões mais antigas ainda puderem receber um documento com base na especificação revisada, com relação a correções de erro ou por motivos operacionais, a nova versão da NCL deve obrigatoriamente ser publicada com o número secundário atualizado;
- se os receptores compatíveis com versões mais antigas não puderem receber um documento baseado nas especificações revisadas, o número principal deve obrigatoriamente ser atualizado.

Uma versão específica está definida sob o URI <http://www.ncl.org.br/NCLx.y/profileName>, onde o número da versão “x.y” é escrito imediatamente após a sigla “NCL”.

O nome do perfil (*profileName*) no URI deve obrigatoriamente ser *EDTVProfile* (Perfil TVD Avançado) ou *BDTVProfile* (Perfil TVD Básico).

7.2 Módulos NCL

Os módulos NCL devem estar de acordo com a ABNT NBR 15606-2:2007, Subseção 7.2.

7.3 Perfis de linguagem NCL para o SBTVD

7.3.1 Módulos de perfis

Os módulos de perfis devem estar de acordo com a ABNT NBR 15606-2:2007, Subseção 7.3.1.

7.3.2 Esquema do perfil NCL 3.0 DTV avançado

NCL30EDTV.xsd

```
<!--  
XML Schema for the NCL Language  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
-->
```

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:animation="http://www.ncl.org.br/NCL3.0/Animation"
  xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
  xmlns:context="http://www.ncl.org.br/NCL3.0/Context"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"
  xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  xmlns:import="http://www.ncl.org.br/NCL3.0/Import"
  xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
  xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
  xmlns:media="http://www.ncl.org.br/NCL3.0/Media"
  xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
  xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
  xmlns:testRuleUse="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"
  xmlns:transitionBase="http://www.ncl.org.br/NCL3.0/TransitionBase"
  xmlns:metainformation="http://www.w3.org/2001/SMIL20/Metainformation"
  xmlns:basicTransition="http://www.w3.org/2001/SMIL20/BasicTransitions"
  xmlns:profile="http://www.ncl.org.br/NCL3.0/EDTVProfile"
  targetNamespace="http://www.ncl.org.br/NCL3.0/EDTVProfile"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<!-- import the definitions in the modules namespaces -->
<import namespace="http://www.ncl.org.br/NCL3.0/Animation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Animation.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnectorFunctionality.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCausalExpression.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ContentControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Context"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/EntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Import"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Layout"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Linking"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Media"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"

```

```

    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Structure"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRule"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Timing"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TransitionBase"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TransitionBase.xsd"/>
<import namespace="http://www.w3.org/2001/SMIL20/Metainformation"
    schemaLocation="http://www.w3.org/2001/SMIL20/smil20-Metainformation.xsd"/>
<import namespace="http://www.w3.org/2001/SMIL20/BasicTransitions"
    schemaLocation="http://www.w3.org/2001/SMIL20/smil20-BasicTransitions.xsd"/>

<!-- =====>
<!-- Structure -->
<!-- =====>
<!-- extends ncl element -->

<element name="ncl" substitutionGroup="structure:ncl"/>

<!-- extends head element -->

<complexType name="headType">
  <complexContent>
    <extension base="structure:headPrototype">
      <sequence>
        <element ref="profile:importedDocumentBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:ruleBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:transitionBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:regionBase" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:descriptorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:connectorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:meta" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:metadata" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="head" type="profile:headType" substitutionGroup="structure:head"/>

<!-- extends body element -->

<complexType name="bodyType">
  <complexContent>
    <extension base="structure:bodyPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:switch"/>
        <element ref="profile:link"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

```

```

<element name="body" type="profile:bodyType" substitutionGroup="structure:body"/>

<!-- ===== -->
<!-- Layout -->
<!-- ===== -->
<!-- extends regionBase element -->

<complexType name="regionBaseType">
  <complexContent>
    <extension base="layout:regionBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:region"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<complexType name="regionType">
  <complexContent>
    <extension base="layout:regionPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="regionBase" type="profile:regionBaseType" substitutionGroup="layout:regionBase"/>
<element name="region" type="profile:regionType" substitutionGroup="layout:region"/>

<!-- ===== -->
<!-- Media -->
<!-- ===== -->
<!-- extends Media elements -->

<!-- media interface element groups -->
<group name="mediaInterfaceElementGroup">
  <choice>
    <element ref="profile:area"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="mediaType">
  <complexContent>
    <extension base="media:mediaPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:mediaInterfaceElementGroup"/>
      </choice>
      <attributeGroup ref="descriptor:descriptorAttrs"/>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
      <attributeGroup ref="extendedEntityReuse:extendedEntityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="media" type="profile:mediaType" substitutionGroup="media:media"/>

<!-- ===== -->
<!-- Context -->
<!-- ===== -->
<!-- extends context element -->

<!-- composite node interface element groups -->
<group name="contextInterfaceElementGroup">
  <choice>
    <element ref="profile:port"/>
  </choice>

```

```

    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="contextType">
  <complexContent>
    <extension base="context:contextPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:link"/>
        <element ref="profile:switch"/>
      </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="context" type="profile:contextType" substitutionGroup="context:context"/>

<!-- =====>
<!-- MediaContentAnchor -->
<!-- =====>
<!-- extends area element -->

<complexType name="componentAnchorType">
  <complexContent>
    <extension base="mediaAnchor:componentAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="area" type="profile:componentAnchorType" substitutionGroup="mediaAnchor:area"/>

<!-- =====>
<!-- CompositeNodeInterface -->
<!-- =====>
<!-- extends port element -->

<complexType name="compositeNodePortType">
  <complexContent>
    <extension base="compositeInterface:compositeNodePortPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="port" type="profile:compositeNodePortType" substitutionGroup="compositeInterface:port"/>

<!-- =====>
<!-- PropertyAnchor -->
<!-- =====>
<!-- extends property element -->

<complexType name="propertyAnchorType">
  <complexContent>
    <extension base="propertyAnchor:propertyAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="property" type="profile:propertyAnchorType" substitutionGroup="propertyAnchor:property"/>

<!-- =====>
<!-- SwitchInterface -->

```

```

<!-- ===== -->
<!-- extends switchPort element -->

<complexType name="switchPortType">
  <complexContent>
    <extension base="switchInterface:switchPortPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="mapping" substitutionGroup="switchInterface:mapping"/>
<element name="switchPort" type="profile:switchPortType" substitutionGroup="switchInterface:switchPort"/>

<!-- ===== -->
<!-- Descriptor -->
<!-- ===== -->
<!-- substitutes descriptorParam element -->

<element name="descriptorParam" substitutionGroup="descriptor:descriptorParam"/>

<!-- extends descriptor element -->

<complexType name="descriptorType">
  <complexContent>
    <extension base="descriptor:descriptorPrototype">
      <attributeGroup ref="layout:regionAttrs"/>
      <attributeGroup ref="timing:explicitDurAttrs"/>
      <attributeGroup ref="timing:freezeAttrs"/>
      <attributeGroup ref="keyNavigation:keyNavigationAttrs"/>
      <attributeGroup ref="profile:transitionAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="descriptor" type="profile:descriptorType" substitutionGroup="descriptor:descriptor"/>

<!-- extends descriptorBase element -->
<complexType name="descriptorBaseType">
  <complexContent>
    <extension base="descriptor:descriptorBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:descriptor"/>
        <element ref="profile:descriptorSwitch"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="descriptorBase" type="profile:descriptorBaseType" substitutionGroup="descriptor:descriptorBase"/>

<!-- ===== -->
<!-- Linking -->
<!-- ===== -->

<!-- substitutes linkParam and bindParam elements -->
<element name="linkParam" substitutionGroup="linking:linkParam"/>
<element name="bindParam" substitutionGroup="linking:bindParam"/>

<!-- extends bind element and link element, as a consequence-->

<complexType name="bindType">
  <complexContent>
    <extension base="linking:bindPrototype">

```

```

    <attributeGroup ref="descriptor:descriptorAttrs"/>
  </extension>
</complexContent>
</complexType>

<element name="bind" type="profile:bindType" substitutionGroup="linking:bind"/>

<!-- extends link element -->
<complexType name="linkType">
  <complexContent>
    <extension base="linking:linkPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="link" type="profile:linkType" substitutionGroup="linking:link"/>

<!-- ===== -->
<!-- Connector -->
<!-- ===== -->
<!-- extends connectorBase element -->

<complexType name="connectorBaseType">
  <complexContent>
    <extension base="connectorBase:connectorBasePrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:causalConnector" />
      </choice>
    </extension>
  </complexContent>
</complexType>

<complexType name="simpleActionType">
  <complexContent>
    <extension base="connectorCausalExpression:simpleActionPrototype">
      <attributeGroup ref="animation:animationAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="connectorBase" type="profile:connectorBaseType" substitutionGroup="connectorBase:connectorBase"/>

<element name="causalConnector" substitutionGroup="causalConnectorFunctionality:causalConnector"/>

<element name="connectorParam" substitutionGroup="causalConnectorFunctionality:connectorParam"/>

<element name="simpleCondition" substitutionGroup="causalConnectorFunctionality:simpleCondition"/>

<element name="compoundCondition" substitutionGroup="causalConnectorFunctionality:compoundCondition"/>

<element name="simpleAction" type="profile:simpleActionType"
substitutionGroup="causalConnectorFunctionality:simpleAction"/>

<element name="compoundAction" substitutionGroup="causalConnectorFunctionality:compoundAction"/>

<element name="assessmentStatement" substitutionGroup="causalConnectorFunctionality:assessmentStatement"/>

<element name="attributeAssessment" substitutionGroup="causalConnectorFunctionality:attributeAssessment"/>

<element name="valueAssessment" substitutionGroup="causalConnectorFunctionality:valueAssessment"/>

<element name="compoundStatement" substitutionGroup="causalConnectorFunctionality:compoundStatement"/>

<!-- ===== -->

```

```

<!-- TestRule -->
<!-- ===== -->
<!-- extends rule element -->
<complexType name="ruleType">
  <complexContent>
    <extension base="testRule:rulePrototype">
      </extension>
    </complexContent>
  </complexType>

  <element name="rule" type="profile:ruleType" substitutionGroup="testRule:rule"/>

  <!-- extends compositeRule element -->
  <complexType name="compositeRuleType">
    <complexContent>
      <extension base="testRule:compositeRulePrototype">
        </extension>
      </complexContent>
    </complexType>

    <element name="compositeRule" type="profile:compositeRuleType" substitutionGroup="testRule:compositeRule"/>

  <!-- extends ruleBase element -->
  <complexType name="ruleBaseType">
    <complexContent>
      <extension base="testRule:ruleBasePrototype">
        <choice minOccurs="1" maxOccurs="unbounded">
          <element ref="profile:importBase"/>
          <element ref="profile:rule"/>
          <element ref="profile:compositeRule"/>
        </choice>
      </extension>
    </complexContent>
  </complexType>

  <element name="ruleBase" type="profile:ruleBaseType" substitutionGroup="testRule:ruleBase"/>

  <!-- ===== -->
  <!-- TestRuleUse -->
  <!-- ===== -->
  <!-- extends bindRule element -->
  <complexType name="bindRuleType">
    <complexContent>
      <extension base="testRuleUse:bindRulePrototype">
        </extension>
      </complexContent>
    </complexType>

    <element name="bindRule" type="profile:bindRuleType" substitutionGroup="testRuleUse:bindRule"/>

  <!-- ===== -->
  <!-- ContentControl -->
  <!-- ===== -->
  <!-- extends switch element -->

  <!-- switch interface element groups -->
  <group name="switchInterfaceElementGroup">
    <choice>
      <element ref="profile:switchPort"/>
    </choice>
  </group>

  <!-- extends defaultComponent element -->
  <complexType name="defaultComponentType">
    <complexContent>

```

```

<extension base="contentControl:defaultComponentPrototype">
</extension>
</complexContent>
</complexType>

<element name="defaultComponent" type="profile:defaultComponentType"
substitutionGroup="contentControl:defaultComponent"/>

<complexType name="switchType">
<complexContent>
<extension base="contentControl:switchPrototype">
<choice minOccurs="0" maxOccurs="unbounded">
<group ref="profile:switchInterfaceElementGroup"/>
<element ref="profile:bindRule"/>
<element ref="profile:switch"/>
<element ref="profile:media"/>
<element ref="profile:context"/>
</choice>
<attributeGroup ref="entityReuse:entityReuseAttrs"/>
</extension>
</complexContent>
</complexType>

<element name="switch" type="profile:switchType" substitutionGroup="contentControl:switch"/>

<!-- ===== -->
<!-- DescriptorControl -->
<!-- ===== -->
<!-- extends defaultDescriptor element -->
<complexType name="defaultDescriptorType">
<complexContent>
<extension base="descriptorControl:defaultDescriptorPrototype">
</extension>
</complexContent>
</complexType>

<element name="defaultDescriptor" type="profile:defaultDescriptorType"
substitutionGroup="descriptorControl:defaultDescriptor"/>

<!-- extends descriptorSwitch element -->

<complexType name="descriptorSwitchType">
<complexContent>
<extension base="descriptorControl:descriptorSwitchPrototype">
<choice minOccurs="0" maxOccurs="unbounded">
<element ref="profile:descriptor"/>
<element ref="profile:bindRule"/>
</choice>
</extension>
</complexContent>
</complexType>

<element name="descriptorSwitch" type="profile:descriptorSwitchType"
substitutionGroup="descriptorControl:descriptorSwitch"/>

<!-- ===== -->
<!-- Timing -->
<!-- ===== -->

<!-- ===== -->
<!-- Import -->
<!-- ===== -->
<complexType name="importBaseType">
<complexContent>

```

```

<extension base="import:importBasePrototype">
</extension>
</complexContent>
</complexType>

<complexType name="importNCLType">
<complexContent>
<extension base="import:importNCLPrototype">
</extension>
</complexContent>
</complexType>

<complexType name="importedDocumentBaseType">
<complexContent>
<extension base="import:importedDocumentBasePrototype">
</extension>
</complexContent>
</complexType>

<element name="importBase" type="profile:importBaseType" substitutionGroup="import:importBase"/>

<element name="importNCL" type="profile:importNCLType" substitutionGroup="import:importNCL"/>
<element name="importedDocumentBase" type="profile:importedDocumentBaseType"
substitutionGroup="import:importedDocumentBase"/>

<!-- ===== -->
<!-- EntityReuse -->
<!-- ===== -->

<!-- ===== -->
<!-- ExtendedEntityReuse -->
<!-- ===== -->

<!-- ===== -->
<!-- KeyNavigation -->
<!-- ===== -->

<!-- ===== -->
<!-- TransitionBase -->
<!-- ===== -->
<!-- extends transitionBase element -->

<complexType name="transitionBaseType">
<complexContent>
<extension base="transitionBase:transitionBasePrototype">
<choice minOccurs="0" maxOccurs="unbounded">
<element ref="profile:transition"/>
<element ref="profile:importBase"/>
</choice>
</extension>
</complexContent>
</complexType>

<element name="transitionBase" type="profile:transitionBaseType" substitutionGroup="transitionBase:transitionBase"/>

<!-- ===== -->
<!-- BasicTransition -->
<!-- ===== -->

<attributeGroup name="transitionAttrs">
<attribute ref="basicTransition:transIn"/>
<attribute ref="basicTransition:transOut"/>
</attributeGroup>

<element name="transition" substitutionGroup="basicTransition:transition"/>

```

```

<!-- ===== -->
<!-- Metainformation -->
<!-- ===== -->

<element name="meta" substitutionGroup="metainformation:meta"/>

<element name="metadata" substitutionGroup="metainformation:metadata"/>

</schema>

```

7.3.3 Esquema do perfil NCL 3.0 CausalConnector

O esquema do perfil NCL 3.0 CausalConnector deve estar de acordo com a ABNT NBR 15606-2:2007, Subseção 7.3.3.

7.3.4 Atributos e elementos do perfil NCL 3.0 DTV Básico

Os elementos e seus atributos, utilizados no perfil NCL 3.0 DTV Básico são apresentados nas Tabelas 6 a 22. Salienta-se que os atributos e conteúdos (elementos-filhos) de elementos podem ser definidos no módulo em si ou no perfil NCL DTV Básico que agrupa os módulos. Os atributos obrigatórios estão sublinhados. Nas Tabelas 6 a 22, os seguintes símbolos são empregados: (?) opcional (zero ou uma ocorrência), () ou (*) zero ou mais ocorrências, (+) uma ou mais ocorrências.

Tabela 6 — Elementos e atributos do módulo *Structure* estendido utilizados no perfil DTV Básico

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

Tabela 7 — Elementos e atributos do módulo *Layout* estendido utilizados no perfil DTV Básico

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

Tabela 8 — Elementos e atributos do módulo *Media* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
Media	<i>id, src, refer, instance, type, descriptor</i>	(area property)*

Tabela 9 — Elementos e atributos do módulo *Context* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
context	<i>id, refer</i>	(port property media context link switch)*

Tabela 10 — Elementos e atributos do módulo *MediaContentAnchor* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
area	<i>id, coords, begin, end, text, position, first, last, label</i>	vazio

Tabela 11 — Elementos e atributos do módulo *CompositeNodeInterface* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
port	<i>id, component, interface</i>	vazio

Tabela 12 — Elementos e atributos do módulo *PropertyAnchor* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
property	<i>name, value</i>	vazio

Tabela 13 — Elementos e atributos do módulo *SwitchInterface* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
switchPort	<i>id</i>	mapping+
mapping	<i>component, interface</i>	vazio

Tabela 14 — Elementos e atributos do módulo *Descriptor* estendido utilizados no perfil DTV Básico

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

Tabela 15 — Elementos e atributos do módulo *Linking* estendido utilizados no perfil DTV Básico

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

Tabela 16 — Elementos e atributos do módulo *CausalConnectorFunctionality* estendido utilizados no perfil DTV Básico

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

Tabela 17 — Elementos e atributos do módulo *ConnectorBase* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
connectorBase	<i>id</i>	(importBase causalConnector)*

Tabela 18 — Elementos e atributos do módulo *TestRule* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
ruleBase	<i>id</i>	(importBase rule compositeRule)+
rule	<i>id, var, comparator, value</i>	vazio
compositeRule	<i>id, operator</i>	(rule compositeRule)+

Tabela 19 — Elementos e atributos do módulo *TestRuleUse* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
bindRule	<i>constituent, rule</i>	vazio

Tabela 20 — Elementos e atributos do módulo *ContentControl* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
switch	<i>id, refer</i>	(defaultComponent?,(switchPort bindRule media context switch)*)
defaultComponent	<i>Component</i>	vazio

Tabela 21 — Elementos e atributos do módulo *DescriptorControl* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
descriptorSwitch	<i>Id</i>	(defaultDescriptor?, (bindRule descriptor)*)
defaultDescriptor	<i>Descriptor</i>	vazio

Tabela 22 — Elementos e atributos do módulo *Import* estendido utilizados no perfil DTV Básico

Elementos	Atributos	Conteúdo
importBase	<i>alias, documentURI, region</i>	vazio
importedDocumentBase	<i>Id</i>	(importNCL)+
importNCL	<i>alias, documentURI,</i>	vazio

7.3.5 Esquema do perfil NCL 3.0 DTV Básico

NCL30BDTV.xsd

```

<!--
XML Schema for the NCL Language

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/profiles/NCL30BDTV.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
  xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
  xmlns:context="http://www.ncl.org.br/NCL3.0/Context"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"
  xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  xmlns:import="http://www.ncl.org.br/NCL3.0/Import"
  xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
  xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
  xmlns:media="http://www.ncl.org.br/NCL3.0/Media"
  xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
  xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
  xmlns:testRuleUse="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"
  xmlns:profile="http://www.ncl.org.br/NCL3.0/BDTVProfile"
  targetNamespace="http://www.ncl.org.br/NCL3.0/BDTVProfile"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- import the definitions in the modules namespaces -->
  <import namespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd"/>

```

```

<import namespace="http://www.ncl.org.br/NCL3.0/CausalConnectorFunctionality"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnectorFunctionality.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ContentControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Context"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/EntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Import"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Layout"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Linking"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Media"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Structure"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRule"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd"/>
<import namespace="http://www.ncl.org.br/NCL3.0/Timing"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd"/>

<!-- ===== -->
<!-- Structure -->
<!-- ===== -->
<!-- extends ncl element -->

<element name="ncl" substitutionGroup="structure:ncl"/>

<!-- extends head element -->

<complexType name="headType">
  <complexContent>
    <extension base="structure:headPrototype">
      <sequence>
        <element ref="profile:importedDocumentBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:ruleBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:regionBase" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="profile:descriptorBase" minOccurs="0" maxOccurs="1"/>
        <element ref="profile:connectorBase" minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="head" type="profile:headType" substitutionGroup="structure:head"/>

```

```

<!-- extends body element -->

<complexType name="bodyType">
  <complexContent>
    <extension base="structure:bodyPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:switch"/>
        <element ref="profile:link"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="body" type="profile:bodyType" substitutionGroup="structure:body"/>

<!-- ===== -->
<!-- Layout -->
<!-- ===== -->
<!-- extends regionBase element -->

<complexType name="regionBaseType">
  <complexContent>
    <extension base="layout:regionBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:region"/>
        <element ref="profile:importBase"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<complexType name="regionType">
  <complexContent>
    <extension base="layout:regionPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="regionBase" type="profile:regionBaseType" substitutionGroup="layout:regionBase"/>
<element name="region" type="profile:regionType" substitutionGroup="layout:region"/>

<!-- ===== -->
<!-- Media -->
<!-- ===== -->
<!-- extends Media elements -->

<!-- media interface element groups -->
<group name="mediaInterfaceElementGroup">
  <choice>
    <element ref="profile:area"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="mediaType">
  <complexContent>
    <extension base="media:mediaPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:mediaInterfaceElementGroup"/>
      </choice>
      <attributeGroup ref="descriptor:descriptorAttrs"/>
    </extension>
  </complexContent>
</complexType>

```

```

    <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    <attributeGroup ref="extendedEntityReuse:extendedEntityReuseAttrs"/>
  </extension>
</complexContent>
</complexType>

<element name="media" type="profile:mediaType" substitutionGroup="media:media"/>

<!-- ===== -->
<!-- Context -->
<!-- ===== -->
<!-- extends context element -->

<!-- composite node interface element groups -->
<group name="contextInterfaceElementGroup">
  <choice>
    <element ref="profile:port"/>
    <element ref="profile:property"/>
  </choice>
</group>

<complexType name="contextType">
  <complexContent>
    <extension base="context:contextPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:contextInterfaceElementGroup"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
        <element ref="profile:link"/>
        <element ref="profile:switch"/>
      </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="context" type="profile:contextType" substitutionGroup="context:context"/>

<!-- ===== -->
<!-- MediaContentAnchor -->
<!-- ===== -->
<!-- extends area element -->

<complexType name="componentAnchorType">
  <complexContent>
    <extension base="mediaAnchor:componentAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="area" type="profile:componentAnchorType" substitutionGroup="mediaAnchor:area"/>

<!-- ===== -->
<!-- CompositeNodeInterface -->
<!-- ===== -->
<!-- extends port element -->

<complexType name="compositeNodePortType">
  <complexContent>
    <extension base="compositeInterface:compositeNodePortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="port" type="profile:compositeNodePortType" substitutionGroup="compositeInterface:port"/>

```

```

<!-- ===== -->
<!-- PropertyAnchor -->
<!-- ===== -->
<!-- extends property element -->

<complexType name="propertyAnchorType">
  <complexContent>
    <extension base="propertyAnchor:propertyAnchorPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="property" type="profile:propertyAnchorType" substitutionGroup="propertyAnchor:property"/>

<!-- ===== -->
<!-- SwitchInterface -->
<!-- ===== -->
<!-- extends switchPort element -->

<complexType name="switchPortType">
  <complexContent>
    <extension base="switchInterface:switchPortPrototype">
    </extension>
  </complexContent>
</complexType>

<element name="mapping" substitutionGroup="switchInterface:mapping"/>
<element name="switchPort" type="profile:switchPortType" substitutionGroup="switchInterface:switchPort"/>

<!-- ===== -->
<!-- Descriptor -->
<!-- ===== -->

<!-- substitutes descriptorParam element -->

<element name="descriptorParam" substitutionGroup="descriptor:descriptorParam"/>

<!-- extends descriptor element -->

<complexType name="descriptorType">
  <complexContent>
    <extension base="descriptor:descriptorPrototype">
      <attributeGroup ref="layout:regionAttrs"/>
      <attributeGroup ref="timing:explicitDurAttrs"/>
      <attributeGroup ref="timing:freezeAttrs"/>
      <attributeGroup ref="keyNavigation:keyNavigationAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="descriptor" type="profile:descriptorType" substitutionGroup="descriptor:descriptor"/>

<!-- extends descriptorBase element -->
<complexType name="descriptorBaseType">
  <complexContent>
    <extension base="descriptor:descriptorBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:descriptor"/>
        <element ref="profile:descriptorSwitch"/>
      </choice>
    </extension>
  </complexContent>

```

```

</complexType>

<element name="descriptorBase" type="profile:descriptorBaseType" substitutionGroup="descriptor:descriptorBase"/>

<!-- ===== -->
<!-- Linking -->
<!-- ===== -->
<!-- substitutes linkParam and bindParam elements -->
<element name="linkParam" substitutionGroup="linking:linkParam"/>
<element name="bindParam" substitutionGroup="linking:bindParam"/>

<!-- extends bind element and link element, as a consequence-->

<complexType name="bindType">
  <complexContent>
    <extension base="linking:bindPrototype">
      <attributeGroup ref="descriptor:descriptorAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="bind" type="profile:bindType" substitutionGroup="linking:bind"/>

<!-- extends link element -->
<complexType name="linkType">
  <complexContent>
    <extension base="linking:linkPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="link" type="profile:linkType" substitutionGroup="linking:link"/>

<!-- ===== -->
<!-- Connector -->
<!-- ===== -->
<!-- extends connectorBase element -->

<complexType name="connectorBaseType">
  <complexContent>
    <extension base="connectorBase:connectorBasePrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:causalConnector" />
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="connectorBase" type="profile:connectorBaseType" substitutionGroup="connectorBase:connectorBase"/>

<element name="causalConnector" substitutionGroup="causalConnectorFunctionality:causalConnector"/>

<element name="connectorParam" substitutionGroup="causalConnectorFunctionality:connectorParam"/>

<element name="simpleCondition" substitutionGroup="causalConnectorFunctionality:simpleCondition"/>

<element name="compoundCondition" substitutionGroup="causalConnectorFunctionality:compoundCondition"/>

<element name="simpleAction" substitutionGroup="causalConnectorFunctionality:simpleAction"/>

<element name="compoundAction" substitutionGroup="causalConnectorFunctionality:compoundAction"/>

<element name="assessmentStatement" substitutionGroup="causalConnectorFunctionality:assessmentStatement"/>

```

```

<element name="attributeAssessment" substitutionGroup="causalConnectorFunctionality:attributeAssessment"/>
<element name="valueAssessment" substitutionGroup="causalConnectorFunctionality:valueAssessment"/>
<element name="compoundStatement" substitutionGroup="causalConnectorFunctionality:compoundStatement"/>
<!-- ===== -->
<!-- TestRule -->
<!-- ===== -->
<!-- extends rule element -->
<complexType name="ruleType">
  <complexContent>
    <extension base="testRule:rulePrototype">
      </extension>
    </complexContent>
  </complexType>
<element name="rule" type="profile:ruleType" substitutionGroup="testRule:rule"/>
<!-- extends compositeRule element -->
<complexType name="compositeRuleType">
  <complexContent>
    <extension base="testRule:compositeRulePrototype">
      </extension>
    </complexContent>
  </complexType>
<element name="compositeRule" type="profile:compositeRuleType" substitutionGroup="testRule:compositeRule"/>
<!-- extends ruleBase element -->
<complexType name="ruleBaseType">
  <complexContent>
    <extension base="testRule:ruleBasePrototype">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element ref="profile:importBase"/>
        <element ref="profile:rule"/>
        <element ref="profile:compositeRule"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
<element name="ruleBase" type="profile:ruleBaseType" substitutionGroup="testRule:ruleBase"/>
<!-- ===== -->
<!-- TestRuleUse -->
<!-- ===== -->
<!-- extends bindRule element -->
<complexType name="bindRuleType">
  <complexContent>
    <extension base="testRuleUse:bindRulePrototype">
      </extension>
    </complexContent>
  </complexType>
<element name="bindRule" type="profile:bindRuleType" substitutionGroup="testRuleUse:bindRule"/>
<!-- ===== -->
<!-- ContentControl -->
<!-- ===== -->
<!-- extends switch element -->
<!-- switch interface element groups -->
<group name="switchInterfaceElementGroup">
  <choice>

```

```

    <element ref="profile:switchPort"/>
  </choice>
</group>

<!-- extends defaultComponent element -->
<complexType name="defaultComponentType">
  <complexContent>
    <extension base="contentControl:defaultComponentPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="defaultComponent" type="profile:defaultComponentType"
substitutionGroup="contentControl:defaultComponent"/>

<complexType name="switchType">
  <complexContent>
    <extension base="contentControl:switchPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <group ref="profile:switchInterfaceElementGroup"/>
        <element ref="profile:bindRule"/>
        <element ref="profile:switch"/>
        <element ref="profile:media"/>
        <element ref="profile:context"/>
      </choice>
      <attributeGroup ref="entityReuse:entityReuseAttrs"/>
    </extension>
  </complexContent>
</complexType>

<element name="switch" type="profile:switchType" substitutionGroup="contentControl:switch"/>

<!-- ===== -->
<!-- DescriptorControl -->
<!-- ===== -->
<!-- extends defaultDescriptor element -->
<complexType name="defaultDescriptorType">
  <complexContent>
    <extension base="descriptorControl:defaultDescriptorPrototype">
      </extension>
    </complexContent>
  </complexType>

<element name="defaultDescriptor" type="profile:defaultDescriptorType"
substitutionGroup="descriptorControl:defaultDescriptor"/>

<!-- extends descriptorSwitch element -->

<complexType name="descriptorSwitchType">
  <complexContent>
    <extension base="descriptorControl:descriptorSwitchPrototype">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="profile:descriptor"/>
        <element ref="profile:bindRule"/>
      </choice>
    </extension>
  </complexContent>
</complexType>

<element name="descriptorSwitch" type="profile:descriptorSwitchType"
substitutionGroup="descriptorControl:descriptorSwitch"/>

<!-- ===== -->
<!-- Timing -->

```

```

<!-- ===== -->

<!-- Import -->
<!-- ===== -->
<complexType name="importBaseType">
  <complexContent>
    <extension base="import:importBasePrototype">
    </extension>
  </complexContent>
</complexType>

<complexType name="importNCLType">
  <complexContent>
    <extension base="import:importNCLPrototype">
    </extension>
  </complexContent>
</complexType>

<complexType name="importedDocumentBaseType">
  <complexContent>
    <extension base="import:importedDocumentBasePrototype">
    </extension>
  </complexContent>
</complexType>

<element name="importBase" type="profile:importBaseType" substitutionGroup="import:importBase"/>

<element name="importNCL" type="profile:importNCLType" substitutionGroup="import:importNCL"/>
<element name="importedDocumentBase" type="profile:importedDocumentBaseType"
substitutionGroup="import:importedDocumentBase"/>

<!-- ===== -->
<!-- EntityReuse -->
<!-- ===== -->

<!-- ===== -->
<!-- ExtendedEntityReuse -->
<!-- ===== -->

<!-- ===== -->
<!-- KeyNavigation -->
<!-- ===== -->

</schema>

```

8 Objetos de mídia em apresentações NCL

8.1 Implementação modular de Ginga-NCL

A apresentação de um documento NCL requer o controle da sincronização de vários objetos de mídia (especificados através do elemento <media>). Para cada objeto de mídia, um *player* (exibidor de mídia) pode ser carregado para o controle do objeto e de seus eventos NCL. Um exibidor de mídia (ou seu adaptador) deve obrigatoriamente ser capaz de receber os comandos de apresentação, controlar as máquinas de estado dos eventos do objeto de mídia controlado e responder às demandas do formatador.

Para favorecer a incorporação de *players* de terceiros dentro da implementação da arquitetura Ginga, um projeto modular do Ginga-NCL é recomendado, para separar os *players* da máquina de apresentação (formatador NCL).

A Figura 2 sugere uma organização modular para a implementação Ginga-NCL. Os exibidores são módulos *plug-in* da máquina de apresentação. Por ser interessante utilizar os *players* já existentes, que possam ter interfaces proprietárias que não sejam compatíveis com as exigidas pela máquina de apresentação, é necessário desenvolver módulos para fazer as adaptações necessárias. Nesse caso, o exibidor é constituído de um adaptador além do exibidor não-conforme em si.

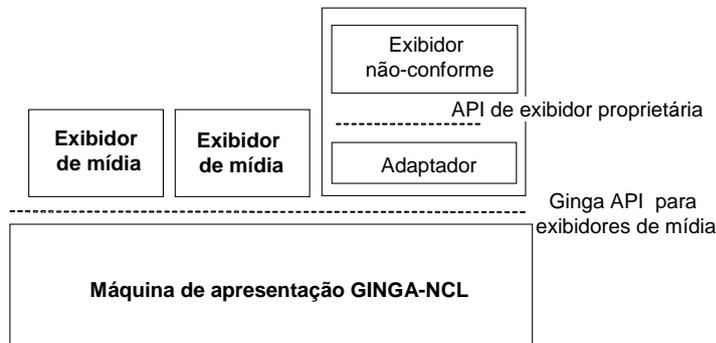


Figura 2 — API para integrar os *players* à implementação da máquina de apresentação NCL

NOTA 1 Como a arquitetura e implementação Ginga-NCL é uma escolha de cada receptor, esta Norma não tem a intenção de padronizar a sintaxe da API da máquina de apresentação e sim apenas definir o comportamento esperado do exibidor de mídia quando se está controlando objetos que fazem parte de um documento NCL.

NOTA 2 O descrito em 8.2 a 8.4 diz respeito aos exibidores de mídia para elemento <media>, cujo conteúdo não é código procedural, isto é, elementos <media> cujos tipos são diferentes de `application/x-ginga-NCLua` e `application/x-ginga-NCLet`. O descrito em 8.5 diz respeito aos exibidores de mídia para os elementos <media>, cujo conteúdo são códigos procedurais Lua (elementos <media>, cujo tipo é `application/x-ginga-NCLua`). Elementos <media> cujo tipo é `application/x-ginga-NCLet` são tratados na Seção 11.

8.2 Comportamento esperado dos exibidores de mídia

O comportamento esperado dos exibidores de mídia deve estar de acordo com a ABNT NBR 15606-2:2007, Subseção 8.2.

8.3 Comportamento esperado dos exibidores de mídia após instruções aplicadas aos objetos de composição

O comportamento esperado dos exibidores de mídia após instruções aplicadas aos objetos de composição deve estar de acordo com a ABNT NBR 15606-2:2007, Subseção 8.3.

8.4 Relação entre a máquina de estado de eventos de apresentação de um nó e a máquina de estado do evento de apresentação de seu nó de composição pai

A relação entre a máquina de estado de eventos de apresentação de um nó e a máquina de estado do evento de apresentação de seu nó de composição pai deve estar de acordo com a ABNT NBR 15606-2:2007, Subseção 8.4.

8.5 Comportamento esperado dos exibidores procedurais Lua

Objetos procedurais em código Lua podem ser inseridos em documentos NCL, trazendo capacidades computacionais adicionais aos documentos declarativos. A forma de adicionar tais objetos em documentos NCL é definir um elemento <media> cujo conteúdo (localizado pelo atributo *src*) é o código procedural a ser executado. Os perfis EDTV e BDTV da NCL 3.0 devem obrigatoriamente dar suporte a elementos <media> do tipo `application/x-ginga-NCLua` (extensão de arquivo `.lua`).

Autores de documentos podem definir elos NCL para iniciar, parar, pausar, retomar ou abortar a execução de um código Lua. Um exibidor Lua (máquina de execução da linguagem) deve obrigatoriamente prover a interface do ambiente de execução procedural com o formatador NCL.

Analogamente ao realizado pelos exibidores de conteúdos de mídia convencional, os exibidores Lua devem obrigatoriamente controlar as máquinas de estado dos eventos associados com o nó procedural Lua. Como exemplo, se um código terminar sua execução, o exibidor deve obrigatoriamente gerar a transição *stops* na máquina de estado de apresentação do evento correspondente à execução procedural.

NCL permite que a execução do código procedural Lua seja sincronizada com outros objetos NCL (procedurais ou não). Um elemento `<media>` do tipo “application/x-ginga-NCLua” também pode definir âncoras (através de elementos `<area>`) e propriedades (através de elementos `<property>`).

Um código Lua pode ser associado a elementos `<area>` (através do atributo *label*). Se elos externos iniciarem, pararem, pausarem, retomarem ou abortarem a apresentação da âncora, *callbacks* no código procedural devem obrigatoriamente ser disparados. A forma como esses *callbacks* são definidos é responsabilidade de cada código procedural associado com o objeto NCLua.

Por outro lado, um código Lua pode também comandar o início, parada, pausa, retomada ou aborto de suas âncoras, através de uma API oferecida pela linguagem. Essas transições podem ser utilizadas como condições de elos NCL para disparar ações em outros objetos NCL do mesmo documento. Assim, uma sincronização de duas vias pode ser estabelecida entre o código Lua e o restante do documento NCL.

A outra forma que um código Lua pode ser sincronizado com outros objetos NCL é através de elementos `<property>`. Um elemento `<property>` definido como filho de um elemento `<media>` do tipo “application/x-ginga-NCLua” pode ser mapeado para um trecho de código ou para um atributo do código lua. Quando é mapeado para um trecho de código, uma ação de elo “set” aplicada à propriedade deve obrigatoriamente causar a execução do código com os valores atribuídos interpretados como parâmetros de entrada. O atributo *name* do elemento `<property>` deve obrigatoriamente ser utilizado para identificar o trecho do código Lua. Quando o elemento `<property>` é mapeado para um atributo do código procedural, a ação “set” deve obrigatoriamente atribuir o valor ao atributo. Como de costume, a máquina de estado de evento associada à propriedade deve ser controlada pelo exibidor Lua.

Um elemento `<property>` definido como filho de um elemento `<media>` do tipo “application/x-ginga-NCLua” também pode estar associado a um *assessment role* de um elo NCL. Nesse caso, o formatador NCL deve obrigatoriamente questionar o valor da propriedade para avaliar a expressão do elo. Se o elemento `<property>` for mapeado para um atributo de código, seu valor deve obrigatoriamente ser retornado pelo exibidor Lua ao formatador NCL. Se o elemento `<property>` for mapeado para um trecho de código Lua, ele deve obrigatoriamente ser chamado e o valor do resultado de sua execução deve obrigatoriamente ser retornado pelo exibidor Lua ao formatador NCL.

A instrução *start* emitida por um formatador deve obrigatoriamente informar ao exibidor Lua os seguintes parâmetros: o objeto NCLua a ser controlado, seu descritor associado, uma lista de eventos (definidos pelos elementos `<area>` e `<property>`, filhos do elemento `<media>` que define o objeto NCLua) que precisam ser monitorados pelo exibidor Lua, o identificador (*id*) do elemento `<area>` associado ao código Lua a ser executado, e um tempo de retardo, opcional. A partir do atributo *src*, o exibidor Lua deve obrigatoriamente tentar localizar o código Lua e iniciar sua execução. Se o conteúdo não puder ser localizado, o exibidor Lua deve obrigatoriamente encerrar a operação de iniciação sem realizar nenhuma ação.

É aconselhado que a lista de eventos a serem monitorados por um exibidor Lua seja também computada pelo formatador, levando em conta a especificação do documento NCL. O formatador deve obrigatoriamente checar todos os elos dos quais participa o objeto de mídia NCLua e o descritor resultante. Ao computar os eventos a serem monitorados, o formatador deve obrigatoriamente considerar a perspectiva do objeto de mídia NCLua, isto é, o caminho dos vários elementos `<body>` e `<context>` para alcançar em profundidade o elemento `<media>` correspondente. Convém que apenas elos contidos nesses elementos `<body>` e `<context>` sejam considerados na computação dos eventos monitorados.

Como com todos os tipos de elemento <media>, o tempo de retardo é um parâmetro opcional, e seu valor *default* é “zero”. Se maior que zero, esse parâmetro contém um tempo a ser esperado pelo exibidor Lua antes de iniciar a execução.

Diferente dos procedimentos realizados para outros tipos de elementos <media>, se um exibidor Lua receber uma instrução *start* para um evento associado a um elemento <area> e esse evento estiver no estado *sleeping*, ele deve dar início à execução do código Lua associado ao elemento, mesmo se outra parte do código procedural do objeto de mídia estiver em execução (pausado ou não). Contudo, se o evento associado ao elemento <area> alvo estiver no estado *occurring* ou *paused*, a instrução *start* deve ser ignorada pelo exibidor Lua que continua controlando a execução anteriormente iniciada. Como consequência, diferentemente do que ocorre para os outros elementos <media>, uma ação <simpleAction> com o atributo *actionType* igual a “stop”, “pause”, “resume” ou “abort” deve se ligar, por meio de um elo, a uma interface do nó Lua, que não deve ser ignorada quando a ação é aplicada.

A instrução *set* emitida por um formatador pode ser aplicada a um objeto NCLua independentemente do fato dele estar sendo executado ou não (nesse último caso, embora o objeto não esteja sendo executado, seu exibidor Lua deve obrigatoriamente já ter sido instanciado). No primeiro caso, a instrução *set* precisa identificar o objeto NCLua, um evento de atribuição monitorado e um valor a ser passado ao código Lua associado ao evento. No segundo caso, deve obrigatoriamente também identificar o elemento <descriptor> que é usado quando da execução do objeto (análogo ao que é feito para a instrução *start*).

Para cada evento de atribuição monitorado, se o exibidor Lua trocar por si mesmo o valor do atributo, ele deve proceder obrigatoriamente como se tivesse recebido uma instrução externa de *set*.

A linguagem Lua deve obrigatoriamente também oferecer uma API que permita que códigos procedurais questionem quaisquer valores de propriedades predefinidas ou dinâmicas do nó *settings* NCL (elemento <media> do tipo “application/x-ginga-settings”). Contudo, deve-se observar que não é permitido atribuir valores a essas propriedades diretamente. Propriedades dos nós do tipo application/x-ginga-settings podem apenas ser modificadas através do uso de elos NCL.

API que forneçam um conjunto de métodos para dar suporte aos comandos de edição do NCL e aos comandos do Gerenciador da Base Privada também devem obrigatoriamente ser oferecidas, conforme apresentado na Seção 10.

9 Transmissão de conteúdo e eventos de fluxo NCL

A transmissão de conteúdo e eventos de fluxo NCL deve estar de acordo com a ABNT NBR 15606-2:2007, Seção 9.

10 Objetos procedurais Lua em apresentações NCL

10.1 Linguagem Lua - Funções opcionais da biblioteca de Lua

A linguagem de *script* adotada pelo Ginga-NCL é Lua (elementos <media> do tipo application/x-ginga-NCLua). A definição completa de Lua é apresentada ABNT NBR 15606-2:2007, Anexo B.

As funções a seguir são dependentes de plataforma e não são de implementação obrigatória:

- 1) no módulo *package*: *loadlib* é opcional;
- 2) no módulo *io*: todas as funções são opcionais;
- 3) no módulo *os*: *clock*, *execute*, *exit*, *getenv*, *remove*, *rename*, *tmpname* e *setlocale* são opcionais;
- 4) no módulo *debug*: todas as funções são opcionais, a API C de *debug* também é opcional.

10.2 Modelo de execução

O ciclo de vida de um objeto NCLua é controlado pelo formatador NCL. O formatador é responsável por iniciar a execução de um objeto NCLua e por mediar a comunicação entre esse objeto e outros objetos em um documento NCL, como definido em 8.5.

Como com todos os exibidores de objetos de mídia, um exibidor Lua, uma vez instanciado, deve executar os procedimentos de iniciação do objeto NCL que controlará. Porém, diferentemente dos outros exibidores de mídia, o código de iniciação deve também ser especificado pelo autor do objeto NCLua. Os procedimentos de iniciação são executados apenas uma vez, para cada instância, e criam funções e objetos que podem ser usados durante a execução do objeto NCLua e, em particular, registram um ou mais tratadores de eventos para a comunicação com o formatador NCL.

Depois da iniciação, a execução do objeto NCLua torna-se orientada a evento, em ambas as direções. Isto é, qualquer ação comandada pelo formatador NCL é dirigida aos tratadores de evento registrados, e qualquer notificação de mudança de estado de eventos NCL é enviada como um evento ao formatador NCL (como, por exemplo, o fim da execução do código procedural). O exibidor Lua estará então pronto para executar qualquer intrusão de *start* ou *set* (ver 8.5).

10.3 Módulos adicionais

10.3.1 Módulos obrigatórios

Além da biblioteca-padrão de Lua, os seguintes módulos devem ser obrigatoriamente oferecidos e automaticamente carregados:

- 1) módulo *ncledit*: permite que *scripts* NCLua alterem a apresentação de um documento NCL;
- 2) módulo *canvas*: oferece uma API para desenhar primitivas gráficas e manipular imagens;
- 3) módulo *event*: permite que aplicações NCLua comuniquem-se com o *middleware* através de eventos (eventos NCL e de teclas);
- 4) módulo *settings*: exporta uma tabela com variáveis definidas pelo autor do documento NCL e variáveis de ambiente reservadas em um nó "application/x-ginga-settings";
- 5) módulo *persistent*: exporta uma tabela com variáveis persistentes, que estão disponíveis para manipulação apenas por objetos procedurais.

A definição de cada função nos módulos mencionados respeita a seguinte nomenclatura:

funcname (parname1: partypel [; optname1: opttypel]) -> retname: rettype

10.3.2 Módulo *ncledit*

Há uma importante diferença com relação aos comandos de edição provenientes dos eventos de fluxo DSM-CC (ver Seção 9) e os comandos de edição realizados pelos *scripts* Lua (objetos NCLua). O primeiro altera não somente a apresentação de um documento NCL, mas também a especificação de um documento NCL. Ou seja, no final do processo um novo documento NCL é gerado, incorporando todos os resultados da edição. Por outro lado, os comandos de edição provenientes dos objetos de mídia NCLua alteram somente a apresentação do documento NCL. O documento original é preservado durante todo o processo de edição.

Todas as funções descritas podem receber uma referência de tempo como parâmetro opcional (parâmetros opcionais são indicados nas assinaturas da função como parâmetros entre colchetes). Esse parâmetro opcional pode ser usado para especificar o exato momento em que o comando de edição deve ser executado. Se este parâmetro não for fornecido na chamada de função, o comando de edição deve ser executado imediatamente. Quando fornecido, o parâmetro pode ter dois tipos diferentes de valores, com diferentes significados. Se for um

valor numérico, define a quantidade de tempo, em segundos, que a execução do comando deve ser postergada. Contudo, esse parâmetro também pode especificar o exato momento para execução do comando, em termos de valores absolutos. Para isso, o parâmetro deve obrigatoriamente ter um valor de tabela com os seguintes campos: *year* (quatro dígitos), *month* (1 a 12), *day* (1 a 31), *hour* (0 a 23), *minute* (0 a 59), *second* (0 a 61) e *isdst* (sinalizador de horário de verão, um booleano).

ncledit.openBase (baseId, location: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
location	Identifica o local a procurar pela base. Se for nulo, uma nova base deve obrigatoriamente ser criada

Valor de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Abre uma base privada existente localizada pelo atributo location. O atributo location deve obrigatoriamente especificar o dispositivo e o caminho para salvar a base. Se a base privada não existir ou o atributo location não for informado, uma nova base é criada com baseId.

ncledit.activateBase (baseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
--------	-----------------------------

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Ativa uma base privada aberta.

ncledit.deactivateBase (baseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
--------	-----------------------------

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Desativa uma base privada aberta.

ncledit.saveBase (baseId, location: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
location	Identifica o dispositivo e o caminho onde a base privada deve obrigatoriamente ser salva

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Salva todo o conteúdo da base privada em um dispositivo de armazenamento persistente (se disponível). O atributo location deve obrigatoriamente especificar o dispositivo e o caminho para salvar a base.

ncledit.closeBase (baseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
--------	-----------------------------

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Fecha a base privada e descarta todo seu conteúdo.

ncledit.addDocument (baseId, documentId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
document	Definição de documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um documento NCL a uma base privada.

ncledit.removeDocument (baseId, documentId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um documento NCL de uma base privada.

**ncledit.startDocument (baseId, documentId, interfaced: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
interfaced	Identifica uma interface de documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Começa a reprodução de um documento NCL em uma base privada, começando a apresentação a partir da interface de documento especificada.

ncledit.stopDocument (baseId, documentId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Para a apresentação de um documento NCL em uma base privada. Todos os eventos do documento que estão ocorrendo devem obrigatoriamente ser parados.

ncledit.pauseDocument (baseId, documentId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Pausa a apresentação de um documento NCL em uma base privada. Todos os eventos do documento que estão ocorrendo devem obrigatoriamente ser pausados.

ncledit.resumeDocument (baseId, documentId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Retoma a apresentação de um documento NCL em uma base privada. Todos os eventos de documentos que foram previamente pausados pelo comando de edição pauseDocument devem obrigatoriamente ser retomados.

ncledit.addRegion (baseId, documentId, regionBaseId, regionId, region: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
regionBaseId	Identifica um elemento <regionBase>
regionId	Identifica um elemento <region> da NCL
region	Definição do elemento <region> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <region> como filho de outra <region> na <regionBase> de um documento NCL.

ncledit.removeRegion (baseId, documentId, regionId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
regionId	Identifica um elemento <region> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <region> da <regionBase> de um documento NCL em uma base privada.

**ncledit.addRegionBase (baseId, documentId, regionBase: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
regionBase	Definição do elemento <regionBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <regionBase> ao elemento <head> de um documento NCL em uma base privada.

**ncledit.removeRegionBase (baseId, documentId, regionBaseId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
regionBaseId	Identifica um elemento <regionBase> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <regionBase> do elemento <head> de um documento NCL em uma base privada.

ncledit.addRule (baseId, documentId, rule: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
rule	Definição do elemento <rule> da NCL.

Valores de retorno

true	Comando bem-sucedido
------	----------------------

false Comando falhou

Descrição

Adiciona um elemento <rule> à <ruleBase> de um documento NCL em uma base privada.

ncledit.removeRule (baseId, documentId, ruleId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId Identifica uma base privada
documentId Identifica um documento NCL
ruleId Identifica um elemento <rule> da NCL

Valores de retorno

true Comando bem-sucedido
false Comando falhou

Descrição

Remove um elemento <rule> da <ruleBase> de um documento NCL em uma base privada.

ncledit.addRuleBase (baseId, documentId, ruleBase [, timeReference: number or table])

Argumentos

baseId String que identifica uma base privada
documentId String que identifica um documento NCL
ruleBase String com uma definição do elemento <ruleBase> da NCL.

Valores de retorno

true Comando bem-sucedido
false Comando falhou

Descrição

Adiciona um elemento <ruleBase> ao elemento <head> de um documento NCL em uma base privada.

**ncledit.removeRuleBase (baseId, documentId, ruleBaseId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId Identifica uma base privada
documentId Identifica um documento NCL

ruleBaselId Identifica um elemento <ruleBase> da NCL

Valores de retorno

true Comando bem-sucedido

false Comando falhou

Descrição

Remove um elemento <ruleBase> do elemento <head> de um documento NCL em uma base privada.

**ncledit.addConnector (baselId, documentId, connector: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baselId Identifica uma base privada

documentId Identifica um documento NCL

connector Definição do elemento <connector> da NCL.

Valores de retorno

true Comando bem-sucedido

false Comando falhou

Descrição

Adiciona um elemento <connector> à <connectorBase> de um documento NCL em uma base privada.

**ncledit.removeConnector (baselId, documentId, connectorId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baselId Identifica uma base privada

documentId Identifica um documento NCL

connectorId Identifica um elemento <connector> da NCL

Valores de retorno

true Comando bem-sucedido

false Comando falhou

Descrição

Remove um elemento <connector> da <connectorBase> de um documento NCL em uma base privada.

**ncledit.addConnectorBase (baseId, documentId, connectorBase: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
connectorBase	Definição do elemento <connectorBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <connectorBase> ao elemento <head> de um documento NCL em uma base privada.

ncledit.removeConnectorBase (baseId, documentId, connectorBaseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
connectorBaseId	Identifica um elemento <connectorBase> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <connectorBase> do elemento <head> de um documento NCL em uma base privada.

**ncledit.addDescriptor (baseId, documentId, descriptor: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descriptor	Definição do elemento <descriptor> da NCL.

Valores de retorno

true	Comando bem-sucedido
------	----------------------

false Comando falhou

Descrição

Adiciona um elemento <descriptor> à <descriptorBase> de um documento NCL em uma base privada.

**ncledit.removeDescriptor (baseId, documentId, descriptorId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descriptorId	Identifica um elemento <descriptor> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <descriptor> da <descriptorBase> de um documento NCL em uma base privada.

**ncledit.addDescriptorSwitch (baseId, documentId, descrSwitch: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descrSwitch	Definição do elemento <descriptorSwitch> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <descriptorSwitch> à <descriptorBase> de um documento NCL em uma base privada.

ncledit.removeDescriptorSwitch (baseId, documentId, descrSwitchId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descrSwitchId	Identifica um elemento <descriptorSwitch> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <descriptorSwitch> da <descriptorBase> de um documento NCL em uma base privada.

ncledit.addDescriptorBase (baseId, documentId, descriptorBase: string [, timeReference number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descriptorBase	Definição do elemento <descriptorBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <descriptorBase> ao elemento <head> de um documento NCL em uma base privada.

ncledit.removeDescriptorBase (baseId, documentId, descrBaseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
descrBaseId	Identifica um elemento <descriptorBase> da NCL

Valores de retorno

true	Comando bem-sucedido
------	----------------------

false Comando falhou

Descrição

Remove um elemento <descriptorBase> do elemento <head> de um documento NCL em uma base privada.

**ncledit.addTransition (baseId, documentId, transition: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
transition	Definição do elemento <transition> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <transition> à <transitionBase> de um documento NCL em uma base privada.

**ncledit.removeTransition (baseId, documentId, transitionId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
transitionId	Identifica um elemento <transition> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <transition> da <transitionBase> de um documento NCL em uma base privada.

**ncledit.addTransitionBase (baseId, documentId, transitionBase: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
transitionBase	Definição do elemento <transitionBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <transitionBase> ao elemento <head> de um documento NCL em uma base privada.

**ncledit.removeTransitionBase (baseId, documentId, transBaseId: string [, timeReference: number or table])
-> boolean: ret**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
transBaseId	Identifica um elemento <transitionBase> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <transitionBase> do elemento <head> de um documento NCL em uma base privada.

ncledit.addImportBase (baseId, documentId, docBaseId, importBase: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
docBaseId	Identifica um elemento-base do NCL (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> ou <connectorBase>)
importBase	Definição do elemento <importBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <importBase> a um elemento-base (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> ou <connectorBase>) de um documento NCL em uma base privada.

ncledit.removeImportBase (baseId, documentId, docBaseId, documentURI: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
docBaseId	Identifica um elemento-base do NCL (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase> ou <connectorBase>)
documentURI	Identifica um elemento <importBase> da NCL através de seu atributo documentURI

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <importBase> de um elemento-base, identificado pelo parâmetro docBaseId de um documento NCL em uma base privada.

ncledit.addImportedDocumentBase (baseId, documentId, impDocBase: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
impDocBase	Definição do elemento <importedDocumentBase> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <importedDocumentBase> ao elemento <head> de um documento NCL em uma base privada.

ncledit.removeImportedDocumentBase (baseId, documentId, impDocBaseId: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
impDocBaseId	Identifica um elemento <importedDocumentBase> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <importedDocumentBase> do elemento <head> de um documento NCL em uma base privada.

ncledit.addImportNCL (baseId, documentId, importNCL: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
importNCL	Definição do elemento <importNCL> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <importNCL> ao elemento <importedDocumentBase> de um documento NCL em uma base privada.

ncledit.removeImportNCL (baseId, documentId, documentURI: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
documentURI	Identifica um elemento <importNCL> da NCL através de seu atributo documentURI

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <importNCL> do elemento <importedDocumentBase> de um documento NCL em uma base privada.

**ncledit.addNode (baseId, documentId, compositelId, node: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
compositelId	Identifica um nó <i>de composição</i> NCL (<body>, <context> ou <switch>)
node	Identifica uma definição do elemento nó do NCL (<media>, <context> ou <switch>)

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um nó NCL (elemento <media>, <context> ou <switch>) a um nó de composição (elemento <body>, <context> ou <switch>) de um documento NCL em uma base privada.

**ncledit.removeNode (baseId, documentId, compositelId, nodeId: string [, timeReference: number or table])
-> ret: boolean**

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
compositelId	Identifica um nó de composição NCL (<body>, <context> ou <switch>)
nodeId	Identifica um nó do NCL (<media>, <context> ou <switch>)

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um nó NCL (elemento <media>, <context> ou <switch>) de um nó de composição (elemento <body>, <context> ou <switch>) de um documento NCL em uma base privada.

ncledit.addInterface (baseId, documentId, nodeId, interface: string [, timeReference: number or table])
-> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
nodeId	Identifica um nó do NCL (<media>, <body>, <context> ou <switch>)
interface	Identifica uma definição de interface do NCL (<port>, <area>, <property> ou <switchPort>)

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona uma interface do NCL (elemento <port>, <area>, <property> ou <switchPort>) a um nó (elemento <media>, <body>, <context> ou <switch>) de um documento NCL em uma base privada.

ncledit.removeInterface (baseId, documentId, nodeId, interfaceId: string [, timeReference: number or table])
-> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
nodeId	Identifica um nó do NCL (<media>, <body>, <context> ou <switch>)
interfaceId	Identifica uma interface do NCL (<port>, <area>, <property> ou <switchPort>)

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove uma interface do NCL (elemento <port>, <area>, <property> ou <switchPort>) de um nó (elemento <media>, <body>, <context> ou <switch>) de um documento NCL em uma base privada.

ncledit.addLink (baseId, documentId, compositId, link: string [, timeReference: number or table])
 -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
compositId	Identifica um nó de composição NCL (<body>, <context> ou <switch>)
link	Definição do elemento <link> da NCL.

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Adiciona um elemento <link> da NCL a um nó de composição (elemento <body>, <context> ou <switch>) de um documento NCL em uma base privada.

ncledit.removeLink (baseId, documentId, compositId, linked: string [, timeReference: number or table])
 -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
compositId	Identifica um nó de composição NCL (<body>, <context> ou <switch>)
linkId	Identifica um elemento <link> da NCL

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Remove um elemento <link> da NCL de um nó de composição (elemento <body>, <context> ou <switch>) de um documento NCL em uma base privada.

ncledit.setPropertyValue (baseId, documentId, nodeId, propertyId, value: string [, timeReference: number or table]) -> ret: boolean

Argumentos

baseId	Identifica uma base privada
documentId	Identifica um documento NCL
nodeId	Identifica um nó NCL (<body>, <context>, <switch>, <media>) contendo a propriedade a ser modificada.
propertyId	Identifica um elemento <property> da NCL
value	Valor a ser ajustado

Valores de retorno

true	Comando bem-sucedido
false	Comando falhou

Descrição

Define o valor para uma propriedade. O propertyId deve obrigatoriamente identificar um atributo *name* do elemento <property> ou um atributo *id* do elemento <switchPort>. A <property> ou <switchPort> deve obrigatoriamente pertencer a um nó (elemento <body>, <context>, <switch> ou <media>) de um documento NCL em uma base privada identificado pelos parâmetros.

10.3.3 Módulo *canvas*

10.3.3.1 Objeto *canvas*

Quando um objeto de mídia NCLua é iniciado, a região do elemento <media> correspondente (do tipo application/x-ginga-NCLua) fica disponível como a variável global *canvas* para o *script* Lua. Se o elemento de <media> não tiver nenhuma região especificada (propriedades *left*, *right*, *top* and *bottom*), então o valor para *canvas* deve ser estabelecido como "nil".

Exemplificando: para uma região definida em um documento NCL como:

```
<region id="luaRegion" width="300" height="100" top="200" left="20"/>
```

A variável '*canvas*' em um objeto de mídia associado à região "luaRegion" é ligada a um objeto *canvas* de tamanho 300x100, associada com a região (20,200).

Um *canvas* oferece uma API gráfica para ser usada por aplicações NCLua. Através dela é possível desenhar linhas, retângulos, fontes, imagens etc.

Um *canvas* guarda em seu estado atributos sob os quais as primitivas de desenho operam, por exemplo, se seu atributo de cor for azul, uma chamada a `canvas:drawLine()` desenha uma linha azul no *canvas*.

As coordenadas passadas são sempre relativas ao ponto mais à esquerda e ao topo do *canvas* (0,0).

10.3.3.2 Construtores

Através de qualquer canvas é possível criar novos canvas e combiná-los através de operações de composição.

canvas:new (image_path: string) -> canvas: object

Argumentos

image_path	Caminho da imagem
------------	-------------------

Valor de retorno

canvas	Canvas representando a imagem
--------	-------------------------------

Descrição

Retorna um novo canvas cujo conteúdo é a imagem recebida como parâmetro.

O novo canvas deve obrigatoriamente manter os aspectos de transparência da imagem original.

canvas:new (width, height: number) -> canvas: object

Argumentos

width	Largura do canvas
-------	-------------------

height	Altura do canvas
--------	------------------

Valores de retorno

canvas	Novo canvas
--------	-------------

Descrição

Retorna um novo canvas com o tamanho recebido.

Inicialmente todos os pixels devem ser transparentes, obrigatoriamente.

10.3.3.3 Atributos

Todos os métodos de atributos possuem o prefixo attr e servem tanto para ler quanto para alterar um atributo (com algumas exceções).

Quando o método é chamado sem parâmetros de entrada, o valor corrente do atributo é retornado; em contra-partida, quando é chamado com parâmetros, esses devem ser os novos valores do atributo.

canvas:attrSize () -> width, height: number

Argumentos - sem argumentos

Valores de retorno

width	Largura do canvas
height	Altura do canvas

Descrição

Retorna as dimensões do canvas

É importante observar que não é permitido alterar as dimensões de um canvas.

canvas:attrColor (R, G, B, A: number)

Argumentos

R	Componente vermelha da cor
G	Componente verde da cor
B	Componente azul da cor
A	Componente alpha da cor

Descrição

Altera a cor do canvas.

As cores são passadas em RGBA, onde A varia de 0 (totalmente transparente) a 255 (totalmente opaco).

As primitivas (ver 10.3.3.4) são desenhadas com a cor desse atributo do canvas.

O valor inicial é '0,0,0,255' (preto).

canvas:attrColor (clr_name: string)

Argumentos

clr_name	Nome da cor
----------	-------------

Altera a cor do canvas.

As cores são passadas através de uma *string* correspondendo a uma das 16 cores NCL pré-definidas:

'white', 'aqua', 'lime', 'yellow', 'red', 'fuchsia', 'purple', 'maroon',
'blue', 'navy', 'teal', 'green', 'olive', 'silver', 'gray', 'black'

Para valores em *string*, o alpha é opaco (correspondendo a "A = 255").

As primitivas (ver 10.3.3.4) são desenhadas com a cor desse atributo do canvas.

O valor inicial é 'black'.

canvas:attrColor () -> R, G, B, A: number*Valores de retorno*

R	Componente vermelha da cor
G	Componente verde da cor
B	Componente azul da cor
A	Componente alpha da cor

Descrição

Retorna a cor do canvas.

canvas:attrFont (face: string; size: number; style: string)*Argumentos*

face	Nome da fonte
size	Tamanho da fonte
style	Estilo da fonte

Descrição

Altera a fonte do canvas.

As seguintes fontes devem obrigatoriamente estar disponíveis: 'Times', 'Courier' e 'Helvetica'.

O tamanho é em pixels e representa a altura máxima de uma linha escrita com a fonte escolhida.

Os estilos possíveis são: 'bold', 'italic' ou 'bold-italic'. O valor `nil` assume que nenhum dos estilos será usado.

Qualquer valor passado não suportado deve obrigatoriamente gerar um erro.

O valor inicial da fonte é indeterminado.

canvas:attrFont () -> face: string; size: number; style: string*Valores de retorno*

face	Nome da fonte
size	Tamanho da fonte
style	Estilo da fonte

Descrição

Retorna a fonte do canvas.

canvas:attrClip (x, y, width, height: number)

Argumentos

x	Coordenada da área de <i>clipping</i>
y	Coordenada da área de <i>clipping</i>
width	Largura da área de <i>clipping</i>
height	Altura da área de <i>clipping</i>

Descrição

Altera a área de *clipping* do canvas.

As primitivas de desenho (ver 10.3.3.4) e o método `canvas:compose()` só operam dentro desta região de *clipping*.

O valor inicial é o canvas inteiro.

canvas:attrClip () -> x, y, width, height: number

Valores de retorno

x	Coordenada da área de <i>clipping</i>
y	Coordenada da área de <i>clipping</i>
width	Largura da área de <i>clipping</i>
height	Altura da área de <i>clipping</i>

Descrição

Retorna a área de *clipping* do canvas.

10.3.3.4 Primitivas

Todos os métodos a seguir levam em consideração os atributos do canvas.

canvas:drawLine (x1, y1, x2, y2: number)

Argumentos

x1	Extremidade 1 da linha
y1	Extremidade 1 da linha
x2	Extremidade 2 da linha
y2	Extremidade 2 da linha

Descrição

Desenha uma linha com suas extremidades em (x1,y1) e (x2,y2).

canvas:drawRect (mode: string; x, y, width, height: number)*Argumentos*

mode	Modo de desenho
x	Coordenada do retângulo
y	Coordenada do retângulo
width	Largura do retângulo
height	Altura do retângulo

Descrição

Função para desenho e preenchimento de retângulos.

O parâmetro mode pode receber 'frame' para desenhar apenas a moldura do retângulo ou 'fill' para preenchê-lo.

canvas:drawPolygon (mode: string) -> drawer: function*Argumentos*

mode	Modo de desenho
------	-----------------

Valores de retorno

f	Função de desenho
---	-------------------

Descrição

Método para desenho e preenchimento de polígonos.

O parâmetro mode recebe o valor 'open', para desenhar o polígono sem ligar o último ponto ao primeiro; 'close', para desenhar o polígono ligando o último ponto ao primeiro; ou 'fill', para desenhar o polígono ligando o último ponto ao primeiro e colorir a região interior.

A função canvas:drawPolygon retorna uma função anônima "drawer" com a assinatura:

```
function (x, y) end
```

A função retornada recebe as coordenadas do próximo vértice do polígono e retorna a si mesmo com resultado. Esse procedimento recorrente facilita a composição:

```
canvas:drawPolygon('fill')(1,1)(10,1)(10,10)(1,10)()
```

Ao receber nil a função "drawer" efetua a operação encadeada. Qualquer chamada subsequente deve obrigatoriamente gerar um erro.

canvas:drawEllipse (mode: string; xc, yc, width, height, ang_start, ang_end: number)

Argumentos

mode	Modo de desenho
xc	Centro da elipse
yc	Centro da elipse
width	Largura da elipse
height	Altura da elipse
ang_start	Ângulo de início
ang_end	Ângulo de fim

Descrição

Desenha elipses e outras primitivas similares tais como círculos, arcos e setores.

O parâmetro mode pode receber 'arc' para desenhar apenas a circunferência ou 'fill' para preenchimento interno.

canvas:drawText (text: string; x, y: number)

Argumentos

text	Texto a ser desenhado
x	Coordenada do texto
y	Coordenada do texto

Descrição

Desenha o texto passado na posição (x,y) do canvas utilizando a fonte configurada em `canvas:attrFont()`.

10.3.3.5 Miscelânea

canvas:flush ()

Descrição

Atualiza o canvas após operações de desenho e de composição.

É suficiente chamá-lo apenas uma vez após uma seqüência de operações.

canvas:compose (x, y: number; src: canvas; [src_x, src_y, src_width, src_height: number])

Argumentos

x	Posição da composição
y	Posição da composição
src	Canvas a ser composto
src_x	Posição da composição no canvas src
src_y	Posição da composição no canvas src
src_width	Largura da composição no canvas src
src_height	Altura da composição no canvas src

Descrição

Faz sobre o canvas (canvas de destino), em sua posição (x,y), a composição pixel a pixel com src (canvas de origem).

Os outros parâmetros são opcionais e indicam que parte do canvas src compor. Quando ausentes, o canvas inteiro é composto.

Essa operação chama src:flush() automaticamente antes da composição.

A composição satisfaz a equação:

$$Cd = Cs * As + Cd * (255 - As) / 255$$

$$Ad = As * As + Ad * (255 - As) / 255$$

onde:

Cd = cor do canvas de destino (canvas)

Ad = alfa do canvas de destino (canvas)

Cs = cor do canvas de origem (src)

As = alfa do canvas de origem (src)

Após a operação, o canvas de destino possui o resultado da composição e src não sofre qualquer alteração.

canvas:pixel (x, y, R, G, B, A: number)

Argumentos

x	Posição do <i>pixel</i>
y	Posição do <i>pixel</i>
R	Componente vermelha da cor
G	Componente verde da cor
B	Componente azul da cor
A	Componente alpha da cor

Descrição

Altera a cor de um *pixel* do canvas.

canvas:pixel (x, y: number) -> R, G, B, A: number

Argumentos

x	Posição do <i>pixel</i>
y	Posição do <i>pixel</i>

Valores de retorno

R	Componente vermelha da cor
G	Componente verde da cor
B	Componente azul da cor
A	Componente alpha da cor

Descrição

Retorna a cor de um *pixel* do canvas.

canvas:measureText (x, y: number; text: string) -> x1, x2, y1, y2: number

Argumentos

x	Coordenada do texto
y	Coordenada do texto
texto	Texto a ser medido

Valores de retorno

x1	Extremidade mais à esquerda do texto
y1	Extremidade mais à esquerda do texto
x2	Extremidade mais à direita do texto
y2	Extremidade mais à direita do texto

Descrição

Retorna as coordenadas limítrofes para o texto passado, caso ele fosse desenhado na posição (x,y) do canvas com a fonte configurada em `canvas:attrFont()`.

10.3.4 Módulo *event*

10.3.4.1 Visão geral

Este módulo oferece uma API para tratamento de eventos. Através dele o formatador NCL e uma aplicação NCLua podem se comunicar de maneira assíncrona.

Uma aplicação também pode usufruir desse mecanismo internamente, através de eventos da classe "user".

Provavelmente o uso mais comum de aplicações NCLua será tratando eventos: sejam eles eventos NCL (ver ABNT NBR 15606-2:2007, Subseção 7.2.8) ou de interação com o usuário (pelo controle remoto, por exemplo).

Durante sua iniciação, antes de se tornar orientado a eventos, um *script* Lua deve obrigatoriamente registrar uma função de tratamento de eventos. Após a iniciação, qualquer ação tomada pela aplicação é somente em resposta a um evento enviado pelo formatador NCL à função "handler".

```
=== example.lua ===
...           -- código de iniciação
function handler (evt)
  ...         -- código tratador
end
event.register(handler) -- registro do tratador no middleware
=== fim ===
```

Entre os tipos de eventos que podem chegar ao handler estão todos os eventos gerados pelo formatador NCL. Um *script* Lua também é capaz de gerar eventos, ditos "espontâneos", com uma chamada à função `event.post(evt)`.

10.3.4.2 Funções

event.post (dst: string; evt: event) -> sent: boolean; err_msg: string

Argumentos

dst	Destinatário do evento
evt	Evento a ser postado

Valores de retorno

sent	Se o evento foi enviado com sucesso
err_msg	Mensagem de erro em caso de falha

Descrição

Posta o evento passado.

O parâmetro "dst" é o destinatário do evento e pode assumir os valores "in" (envio para a própria aplicação) e "out" (envio para o formatador NCL).

event.timer (time: number, f: function)

Argumentos

time	Tempo em milissegundos
f	Função de <i>callback</i>

Descrição

Cria um timer que expira após time (em milissegundos) e então chama a função f.

A assinatura de f é simples, sem recebimento de parâmetros:

```
function f () end
```

Uma vez criado, não há como cancelar o timer.

O valor de 0 milissegundos é válido. Nesse caso, event.timer() deve, obrigatoriamente, retornar imediatamente e f deve ser chamada assim que possível.

event.register (f: function)

Argumentos

f	Função de <i>callback</i>
---	---------------------------

Descrição

Registra a função passada como um *listener* de eventos, isto é, sempre que ocorrer um evento, f será chamada. A função f é, assim, a função de tratamento de eventos (function "handler").

A assinatura de *f* é:

```
function f (evt) end -> handled: boolean
```

Onde *evt* é o evento que, ao ocorrer, ativa a função.

A função pode retornar “true”, para sinalizar que o evento foi tratado e, portanto, não deve ser enviado a outros tratadores.

É recomendado que a função, definida pela aplicação, retorne rapidamente, já que, enquanto ela estiver executando, nenhum outro evento é processado.

O formatador NCL deve obrigatoriamente garantir que as funções recebam os eventos na ordem em que foram registradas e, se nenhuma delas retornar o valor “true”, o formatador NCL deve notificar os outros tratadores registrados.

event.unregister (f: function)

Argumentos

<i>f</i>	Função de <i>callback</i>
----------	---------------------------

Descrição

Tira do registro a função passada como um *listener*, isto é, novos eventos não serão mais passados a *f*.

event.uptime () -> ms: number

Valores de retorno

ms	Tempo em milissegundos
----	------------------------

Descrição

Retorna o número de milissegundos decorridos desde o início da aplicação.

10.3.4.3 Classes de eventos

A função `event.post()` e o “handler” registrado em `event.register()` recebem eventos como parâmetros.

Um evento é descrito por uma tabela Lua normal, onde o campo `class` é obrigatório e identifica a classe do evento.

As seguintes classes de eventos são definidas:

Classe `key`:

```
evt = { class='key', type: string, key: string}
```

* `type` pode ser 'press' ou 'release'.

* `key` é o valor da tecla em questão.

EXEMPLO `evt = { class='key', type='press', key='0' }`

Classe ncl:

As relações entre os nós de mídia NCL são baseadas em eventos. Lua tem acesso a esses eventos através da Classe ncl.

Os eventos podem agir nas duas direções, isto é, o formatador pode enviar eventos de ação para mudar o estado do exibidor Lua, e este, por sua vez, pode disparar eventos de transição para indicar mudanças de estado.

Nos eventos, o campo type deve obrigatoriamente assumir um dos três valores:
'presentation', 'selection' ou 'attribution'

Eventos podem ser direcionados a âncoras específicas ou ao nó como um todo, isto é identificado no campo area, que assume o nó inteiro, quando ausente.

No caso de um evento gerado pelo formatador, o campo action deve obrigatoriamente ter um dos valores:
'start', 'stop', 'abort', 'pause', 'resume' e 'set'

Tipo 'presentation':

```
evt = { class='ncl', type='presentation', area='?', action='start'/stop'/abort'/pause'/resume' }
```

Tipo 'attribution':

```
evt = { class='ncl', type='attribution', area='?', action='set' }
```

Para eventos gerados pelo exibidor Lua, o campo "transition" deverá obrigatoriamente assumir um dos valores:
'starts', 'stops', 'aborts', 'pauses', 'resumes' e 'stops'

Tipo 'presentation':

```
evt = { class='ncl', type='presentation', area='?', transition='starts'/stops'/aborts'/pauses'/resumes' }
```

Tipo 'selection':

```
evt = { class='ncl', type='selection', area='?', transition='stops' }
```

Tipo 'attribution':

```
evt = { class='ncl', type='attribution', area='?', transition='stops' }
```

tcp class:

O uso do canal de interatividade é realizado por meio desta classe de eventos.

Uma aplicação NCLua envia dados por um canal de interatividade, postando eventos na forma:

```
evt = { class='tcp', to='addr:port', value=string }
```

De forma similar, uma aplicação NCLua recebe dados transportados por um canal de interatividade fazendo uso de eventos na forma:

```
evt = { class='tcp', from='addr:port', value=string }
```

NOTA Recomenda-se que questões como autenticação, temporização/reconexão de uma conexão, se uma conexão deve ser mantida aberta ou não etc. sejam tratadas em uma implementação específica do *middleware*.

sms class:

O compartimento de envio e recebimento por meio de SMS é muito semelhante ao definido na classe *tcp*.

Uma aplicação NCLua envia dados por SMS, postando eventos na forma:

```
evt = { class='sms', to='phone number', value=string }
```

De forma similar, uma aplicação NCLua recebe dados transportados por SMS fazendo uso de eventos na forma:

```
evt = { class='sms', from='phone number', value=string }
```

NOTA Recomenda-se que questões como autenticação etc. sejam tratadas em uma implementação específica do *middleware*.

Classe user:

Utilizando Classe user, aplicações podem estender suas funcionalidades, criando seus próprios eventos.

Nessa classe, nenhum campo está definido (além, obviamente, do campo class).

10.3.5 Módulo settings

Exporta a tabela *settings* com variáveis definidas pelo autor do documento NCL e variáveis de ambiente reservadas, contidas no nó application/x-ginga-settings.

Não é permitido atribuir valores aos campos representando variáveis no nós *settings*. Um erro deve ser gerado caso uma tentativa de atribuição seja feita. Propriedades de um nós *settings* só podem ser modificadas por meio de elos NCL.

A tabela *settings* particiona seus grupos em várias subtabelas, correspondendo a cada grupo do nó application/x-ginga-settings. Por exemplo, em um objeto NCLua, a variável do nó *settings* "system.CPU" é referida como settings.system.CPU.

Exemplos de uso:

```
lang = settings.system.language
```

```
age = settings.user.age
```

```
val = settings.default.selBorderColor
```

```
settings.service.myVar = 10
```

```
settings.user.age = 18 --> ERRO!
```

10.3.6 Módulo *persistent*

Aplicações NCLua podem salvar dados em uma área restrita do *middleware* e recuperá-los entre execuções. O exibidor Lua permite a uma aplicação NCLua persistir um valor para ser posteriormente usado por ela ou por um outro objeto procedural. Para tanto, o exibidor define uma área reservada, inacessível a objetos NCL não procedurais. Esta área é dividida entre os grupos “*service*”, “*channel*” e “*shared*”, com a mesma semântica dos grupos homônimos do nó NCL *settings*. Não existe nenhuma variável pré-definida ou reservada nesses grupos, e objetos procedurais podem atribuir valores a essas variáveis diretamente. Recomenda-se que outras linguagens procedurais, Java em particular para os objetos NCLets (<media> elements of type application/x-ginga-NCLet), ofereçam uma API dando acesso à mesma área.

Neste módulo *persistent*, Lua oferece uma API para exportar a tabela *persistent* com as variáveis definidas na área reservada.

O uso da tabela *persistent* é semelhante ao uso da tabela *settings*, exceto pelo fato de que, neste caso, o código procedural pode mudar os valores dos campos.

Exemplos de uso:

```
persistent.service.total = 10
```

```
color = persistent.shared.color
```

11 Objetos procedurais Java em documentos NCL

O suporte a objetos procedurais em código Java é opcional em documentos NCL para dispositivos *one-seg*. A forma de adicionar tais objetos em documentos NCL é definir um elemento <media> cujo conteúdo (localizado pelo atributo *src*) é o código procedural a ser executado. Os perfis EDTV e BDTV da NCL 3.0 podem, opcionalmente, dar suporte a elementos <media> do tipo “application/x-ginga-NCLet” (extensão de arquivo .class ou .jar).

Autores de documentos podem definir elos NCL para iniciar, parar, pausar, retomar ou abortar a execução de um código Java. Um exibidor Java (máquina de execução da linguagem) deve obrigatoriamente prover a interface do ambiente de execução procedural com o formatador NCL.

Analogamente ao realizado pelos exibidores de conteúdos de mídia convencional, os exibidores Java devem obrigatoriamente controlar as máquinas de estado dos eventos associados com o nó procedural. Como exemplo, se um código terminar sua execução, o exibidor deve obrigatoriamente gerar a transição *stops* na máquina de estado de apresentação do evento correspondente à execução procedural.

NCL permite que a execução do código procedural java seja sincronizada com outros objetos NCL (procedurais ou não). Um elemento <media> do tipo “application/x-ginga-NCLet” também pode definir âncoras (através de elementos <area>) e propriedades (através de elementos <property>).

Um código Java pode ser associado a elementos <area> (através do atributo *label*). Se elos externos iniciarem, pararem, pausarem, retomarem ou abortarem a apresentação da âncora, *callbacks* no código procedural devem obrigatoriamente ser disparados. A forma como esses *callbacks* são definidos é responsabilidade de cada código procedural associado com o objeto NCLet.

Por outro lado, um código Java pode também comandar o início, parada, pausa, retomada ou aborto de suas âncoras, através de uma API oferecida pela linguagem. Essas transições podem ser utilizadas como condições de elos NCL para disparar ações em outros objetos NCL do mesmo documento. Assim, uma sincronização (ponte) de duas vias pode ser estabelecida entre o código Java e o restante do documento NCL.

A outra forma que um código Java pode ser sincronizado com outros objetos NCL é através de elementos <property>. Um elemento <property> definido como filho de um elemento <media> do tipo “application/x-ginga-NCLet” pode ser mapeado para um trecho de código ou para um atributo do código Java. Quando é mapeado para um trecho código, uma ação de elo “set” aplicada à propriedade deve obrigatoriamente causar a execução do código com os valores atribuídos interpretados como parâmetros de entrada. O atributo *name* do elemento <property> deve obrigatoriamente ser utilizado para identificar o trecho de código Java. Quando o elemento <property> é mapeado para um atributo do código procedural, a ação “set” deve obrigatoriamente atribuir o valor ao atributo. A máquina de estado de evento associada à propriedade deve ser controlada pelo exibidor Java.

Um elemento <property> definido como filho de um elemento <media> do tipo “application/x-ginga-NCLet” também pode estar associado a um *assessment role* de um elo NCL. Nesse caso, o formatador NCL deve obrigatoriamente questionar o valor da propriedade para avaliar a expressão do elo. Se o elemento <property> for mapeado para um atributo de código, seu valor deve obrigatoriamente ser retornado pelo exibidor Java ao formatador NCL. Se o elemento <property> for mapeado para um trecho de código Java, ele deve obrigatoriamente ser chamado e o valor do resultado de sua execução deve obrigatoriamente ser retornado pelo exibidor Java ao formatador NCL.

A instrução *start* emitida por um formatador deve obrigatoriamente informar ao exibidor Java os seguintes parâmetros: o objeto NCLet a ser controlado, seu descritor associado, uma lista de eventos (definidos pelos elementos <area> e <property>, filhos do elemento <media> que define o objeto NCLet) que precisam ser monitorados pelo exibidor Java, o identificador (*id*) do elemento <area> associado ao código Java a ser executado e um tempo de retardo, opcional. A partir do atributo *src*, o exibidor Java deve tentar localizar o código Java e iniciar sua execução. Se o conteúdo não puder ser localizado, o exibidor Java deve obrigatoriamente encerrar a operação de iniciação sem realizar nenhuma ação.

É aconselhado que a lista de eventos a serem monitorados por um exibidor Java também seja computada pelo formatador, levando em conta a especificação do documento NCL. Ele deve obrigatoriamente checar todos os elos dos quais participa o objeto de mídia NCLet e o descritor resultante. Ao computar os eventos a serem monitorados, o formatador deve obrigatoriamente considerar a perspectiva do objeto de mídia NCLet, isto é, o caminho dos vários elementos <body> e <context> para alcançar em profundidade o elemento <media> correspondente. Convém que apenas elos contidos nesses elementos <body> e <context> sejam considerados na computação dos eventos monitorados.

Como com todos os tipos de elemento <media>, o tempo de retardo é um parâmetro opcional, e seu valor *default* é “zero”. Se maior que zero, esse parâmetro contém um tempo a ser esperado pelo exibidor Java antes de iniciar a execução.

De forma idêntica aos procedimentos realizados para elementos <media> com código procedural Lua, se um exibidor Java receber uma instrução *start* para um evento associado a um elemento <area> e esse evento estiver no estado *sleeping*, ele pode dar início à execução do código Java associado ao elemento, mesmo se outra parte do código procedural do objeto de mídia estiver em execução (pausado ou não). Contudo, se o evento associado ao elemento <area> alvo estiver no estado *occurring* ou *paused*, a instrução *start* deve ser ignorada pelo exibidor Java que continua controlando a execução anteriormente iniciada. Como consequência, de forma similar ao que ocorre para elementos <media> contendo código Lua, uma ação <simpleAction> com o atributo *actionType* igual a “stop”, “pause”, “resume” ou “abort” deve se ligar, por meio de um elo, a uma interface do nó NCLet, que não deve ser ignorada quando a ação é aplicada.

A instrução *set* emitida por um formatador também pode ser aplicada a um objeto NCLet, independentemente do fato dele estar sendo executado ou não (nesse último caso, embora o objeto não esteja sendo executado, seu exibidor Java deve obrigatoriamente já ter sido instanciado). No primeiro caso, a instrução *set* precisa identificar o objeto NCLet, um evento de atribuição monitorado e um valor a ser passado ao código Java associado ao evento. No segundo caso, deve obrigatoriamente também identificar o elemento <descriptor> que será usado quando da execução do objeto (análogo ao que é feito para a instrução *start*).

Para cada evento de atribuição monitorado, se o exibidor Java trocar por si mesmo o valor do atributo, ele deve proceder, obrigatoriamente, como se tivesse recebido uma instrução externa de *set*.

Recomenda-se também que se ofereça uma API Java que permita que os códigos procedurais questionem quaisquer valores de propriedades predefinidas ou dinâmicas do nó *settings* NCL (elemento <media> do tipo “application/x-ginga-settings”). Contudo, deve-se observar que não é permitido atribuir valores a essas propriedades diretamente. Propriedades dos nós do tipo application/x-ginga-settings podem apenas ser modificadas através do uso de elos NCL. Recomenda-se também, que se ofereça uma API Java que permita manipular variáveis persistentes definidas na área reservada de persistência de Lua.

API que forneçam um conjunto de métodos para dar suporte aos comandos de edição do NCL e comandos do Gerenciador da Base Privada também são recomendadas, caso o suporte a Java seja oferecido.

12 Requisitos de codificação de mídias e métodos de transmissão referenciados em documentos NCL

A codificação de mídias e métodos de transmissão referenciados em documentos NCL devem estar de acordo com a ABNT NBR 15606-2:2007, Seção 12.

13 Segurança

A segurança na execução das aplicações deve estar de acordo com ABNT NBR 15606-2:2007, Seção 13.

Anexo A (normativo)

Esquemas dos módulos NCL 3.0 usados nos perfis TVD Básico e TVD Avançado

A.1 Módulo *Structure*: NCL30Structure.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Structure.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the Structure module namespace,
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:structure="http://www.ncl.org.br/NCL3.0/Structure"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Structure"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- ===== -->
  <!-- define the top-down structure of an NCL language document. -->
  <!-- ===== -->

  <complexType name="nclPrototype">
    <sequence>
      <element ref="structure:head" minOccurs="0" maxOccurs="1"/>
      <element ref="structure:body" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="title" type="string" use="optional"/>
  </complexType>

  <complexType name="headPrototype">
  </complexType>

  <complexType name="bodyPrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="ncl" type="structure:nclPrototype"/>
  <element name="head" type="structure:headPrototype"/>
  <element name="body" type="structure:bodyPrototype"/>

</schema>

```

A.2 Módulo *Layout*: NCL30Layout.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Layout.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Layout module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:layout="http://www.ncl.org.br/NCL3.0/Layout"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Layout"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="regionBasePrototype">
    <attribute name="id" type="ID" use="optional"/>
    <attribute name="type" type="string" use="optional"/>
    <attribute name="device" type="string" use="optional"/>
  </complexType>

  <complexType name="regionPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="layout:region" />
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="title" type="string" use="optional"/>
    <attribute name="height" type="string" use="optional"/>
    <attribute name="left" type="string" use="optional"/>
    <attribute name="right" type="string" use="optional"/>
    <attribute name="top" type="string" use="optional"/>
    <attribute name="bottom" type="string" use="optional"/>
    <attribute name="width" type="string" use="optional"/>
    <attribute name="zIndex" type="integer" use="optional"/>
  </complexType>

  <!-- declare global attributes in this module -->

  <!-- define the region attributeGroup -->
  <attributeGroup name="regionAttrs">
    <attribute name="region" type="string" use="optional"/>
  </attributeGroup>

  <!-- declare global elements in this module -->
  <element name="regionBase" type="layout:regionBasePrototype"/>
  <element name="region" type="layout:regionPrototype"/>

</schema>

```

A.3 Módulo *Media*: NCL30Media.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Media.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Media module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:media="http://www.ncl.org.br/NCL3.0/Media"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/Media"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="mediaPrototype">  
    <attribute name="id" type="ID" use="required"/>  
    <attribute name="type" type="string" use="optional"/>  
    <attribute name="src" type="anyURI" use="optional"/>  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="media" type="media:mediaPrototype"/>  
  
</schema>
```

A.4 Módulo *Context*: NCL30Context.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Context.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Context module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:context="http://www.ncl.org.br/NCL3.0/Context"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/Context"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <!-- define the compositeNode element prototype -->  
  <complexType name="contextPrototype">  
    <attribute name="id" type="ID" use="required"/>  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="context" type="context:contextPrototype"/>  
  
</schema>
```

A.5 Módulo *MediaContentAnchor*: NCL30MediaContentAnchor.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30MediaContentAnchor.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Media Content Anchor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mediaAnchor="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/MediaContentAnchor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- define the temporalAnchorAttrs attribute group -->
  <attributeGroup name="temporalAnchorAttrs">
    <attribute name="begin" type="string" use="optional"/>
    <attribute name="end" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the textAnchorAttrs attribute group -->
  <attributeGroup name="textAnchorAttrs">
    <attribute name="text" type="string" use="optional"/>
    <attribute name="position" type="unsignedLong" use="optional"/>
  </attributeGroup>

  <!-- define the sampleAnchorAttrs attribute group -->
  <attributeGroup name="sampleAnchorAttrs">
    <attribute name="first" type="unsignedLong" use="optional"/>
    <attribute name="last" type="unsignedLong" use="optional"/>
  </attributeGroup>

  <!-- define the coordsAnchorAttrs attribute group -->
  <attributeGroup name="coordsAnchorAttrs">
    <attribute name="coords" type="string" use="optional"/>
  </attributeGroup>

  <!-- define the labelAttrs attribute group -->
  <attributeGroup name="labelAttrs">
    <attribute name="label" type="string" use="optional"/>
  </attributeGroup>

  <complexType name="componentAnchorPrototype">
    <attribute name="id" type="ID" use="required"/>
    <attributeGroup ref="mediaAnchor:coordsAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:temporalAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:textAnchorAttrs" />
    <attributeGroup ref="mediaAnchor:sampleAnchorAttrs" />
  </complexType>

```

```
<attributeGroup ref="mediaAnchor:labelAttrs" />
</complexType>

<!-- declare global elements in this module -->
<element name="area" type="mediaAnchor:componentAnchorPrototype"/>

</schema>
```

A.6 Módulo *CompositeNodeInterface*: NC30CompositeNodeInterface.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30CompositeNodeInterface.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Composite Node Interface module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:compositeInterface="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/CompositeNodeInterface"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="compositeNodePortPrototype">  
    <attribute name="id" type="ID" use="required" />  
    <attribute name="component" type="IDREF" use="required"/>  
    <attribute name="interface" type="string" use="optional" />  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="port" type="compositeInterface:compositeNodePortPrototype" />  
  
</schema>
```

A.7 Módulo *PropertyAnchor*: NCL30PropertyAnchor.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30PropertyAnchor.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Property Anchor module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:propertyAnchor="http://www.ncl.org.br/NCL3.0/PropertyAnchor"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/PropertyAnchor"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="propertyAnchorPrototype">  
    <attribute name="name" type="string" use="required" />  
    <attribute name="value" type="string" use="optional" />  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="property" type="propertyAnchor:propertyAnchorPrototype"/>  
  
</schema>
```

A.8 Módulo *SwitchInterface*: NCL30SwitchInterface.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30SwitchInterface.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Switch Interface module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:switchInterface="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  targetNamespace="http://www.ncl.org.br/NCL3.0/SwitchInterface"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="mappingPrototype">
    <attribute name="component" type="IDREF" use="required"/>
    <attribute name="interface" type="string" use="optional"/>
  </complexType>

  <complexType name="switchPortPrototype">
    <sequence>
      <element ref="switchInterface:mapping" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="mapping" type="switchInterface:mappingPrototype"/>
  <element name="switchPort" type="switchInterface:switchPortPrototype" />

</schema>

```

A.9 Módulo *Descriptor*: NCL30Descriptor.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Descriptor.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Descriptor module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptor="http://www.ncl.org.br/NCL3.0/Descriptor"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Descriptor"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="descriptorParamPrototype">
    <attribute name="name" type="string" use="required" />
    <attribute name="value" type="string" use="required"/>
  </complexType>

  <complexType name="descriptorPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="descriptor:descriptorParam"/>
    </sequence>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="player" type="string" use="optional"/>
  </complexType>

<!--
Formatters should support the following descriptorParam names.
* For audio players: soundLevel; balanceLevel; trebleLevel; bassLevel.
* For text players: style, which refers to a style sheet with information for text presentation.
* For visual media players: background, specifying the background color used to fill the area of a region displaying
media; scroll, which allows the specification of how an author would like to configure the scroll in a region; fit,
indicating how an object will be presented (hidden, fill, meet, meetBest, slice); transparency, indicating the degree
of transparency of an object presentation (the value must be between 0 and 1, or a real number in the range
[0,100] ending by the character "%" (ex. 30%)); visible, indicating if the presentation is to be seen or hidden; and the
object positioning parameters: top, left, bottom, right, width, height, size and bounds.
* For players in general: reusePlayer, which determines if a new player must be instantiated or if a player already
instantiated must be used; and playerLife, which specifies what will happen to the player instance at the end of the
presentation.
-->

  <complexType name="descriptorBasePrototype">
    <attribute name="id" type="ID" use="optional"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="descriptorParam" type="descriptor:descriptorParamPrototype"/>
  <element name="descriptor" type="descriptor:descriptorPrototype"/>

```

```
<element name="descriptorBase" type="descriptor:descriptorBasePrototype"/>  
  
<!-- declare global attributes in this module -->  
<attributeGroup name="descriptorAttrs">  
  <attribute name="descriptor" type="string" use="optional"/>  
</attributeGroup>  
  
</schema>
```

A.10 Módulo *Linking*: NCL30Linking.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Linking.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Linking module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:linking="http://www.ncl.org.br/NCL3.0/Linking"
  targetNamespace="http://www.ncl.org.br/NCL3.0/Linking"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="paramPrototype">
    <attribute name="name" type="string" use="required"/>
    <attribute name="value" type="anySimpleType" use="required"/>
  </complexType>

  <complexType name="bindPrototype">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="linking:bindParam"/>
    </sequence>
    <attribute name="role" type="string" use="required"/>
    <attribute name="component" type="IDREF" use="required"/>
    <attribute name="interface" type="string" use="optional"/>
  </complexType>

  <complexType name="linkPrototype">
    <sequence>
      <element ref="linking:linkParam" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="linking:bind" minOccurs="2" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID" use="optional"/>
    <attribute name="xconnector" type="string" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="linkParam" type="linking:paramPrototype"/>
  <element name="bindParam" type="linking:paramPrototype"/>
  <element name="bind" type="linking:bindPrototype" />
  <element name="link" type="linking:linkPrototype" />

</schema>

```

A.11 Módulo *ConnectorCommonPart*: NCL30ConnectorCommonPart.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Common Part module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="parameterPrototype">
    <attribute name="name" type="string" use="required"/>
    <attribute name="type" type="string" use="optional"/>
  </complexType>

  <simpleType name="eventPrototype">
    <restriction base="string">
      <enumeration value="presentation" />
      <enumeration value="selection" />
      <enumeration value="attribution" />
      <enumeration value="composition" />
    </restriction>
  </simpleType>

  <simpleType name="logicalOperatorPrototype">
    <restriction base="string">
      <enumeration value="and" />
      <enumeration value="or" />
    </restriction>
  </simpleType>

  <simpleType name="transitionPrototype">
    <restriction base="string">
      <enumeration value="starts" />
      <enumeration value="stops" />
      <enumeration value="pauses" />
      <enumeration value="resumes" />
      <enumeration value="aborts" />
    </restriction>
  </simpleType>

</schema>

```

**A.12 Módulo *ConnectorAssessmentExpression*:
NCL30ConnectorAssessmentExpression.xsd**

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorAssessmentExpression.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Assessment Expression module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorAssessmentExpression="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<!-- import the definitions in the modules namespaces -->
<import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd"/>

<simpleType name="comparatorPrototype">
  <restriction base="string">
    <enumeration value="eq" />
    <enumeration value="ne" />
    <enumeration value="gt" />
    <enumeration value="lt" />
    <enumeration value="gte" />
    <enumeration value="lte" />
  </restriction>
</simpleType>

<simpleType name="attributePrototype">
  <restriction base="string">
    <enumeration value="repeat" />
    <enumeration value="occurrences" />
    <enumeration value="state" />
    <enumeration value="nodeProperty" />
  </restriction>
</simpleType>

<simpleType name="statePrototype">
  <restriction base="string">
    <enumeration value="sleeping" />
    <enumeration value="occurring" />
    <enumeration value="paused" />
  </restriction>
</simpleType>

```

```

<simpleType name="valueUnion">
  <union memberTypes="string" connectorAssessmentExpression:statePrototype"/>
</simpleType>

<complexType name="assessmentStatementPrototype" >
  <sequence>
    <element ref="connectorAssessmentExpression:attributeAssessment"/>
    <choice>
      <element ref="connectorAssessmentExpression:attributeAssessment"/>
      <element ref="connectorAssessmentExpression:valueAssessment"/>
    </choice>
  </sequence>
  <attribute name="comparator" type="connectorAssessmentExpression:comparatorPrototype" use="required"/>
</complexType>

<complexType name="attributeAssessmentPrototype">
  <attribute name="role" type="string" use="required"/>
  <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="required"/>
  <attribute name="key" type="string" use="optional"/>
  <attribute name="attributeType" type="connectorAssessmentExpression:attributePrototype" use="optional"/>
  <attribute name="offset" type="string" use="optional"/>
</complexType>

<complexType name="valueAssessmentPrototype">
  <attribute name="value" type="connectorAssessmentExpression:valueUnion" use="required"/>
</complexType>

<complexType name="compoundStatementPrototype">
  <choice minOccurs="1" maxOccurs="unbounded">
    <element ref="connectorAssessmentExpression:assessmentStatement" />
    <element ref="connectorAssessmentExpression:compoundStatement" />
  </choice>
  <attribute name="operator" type="connectorCommonPart:logicalOperatorPrototype" use="required"/>
  <attribute name="isNegated" type="boolean" use="optional"/>
</complexType>

  <!-- declare global elements in this module -->
  <element name="assessmentStatement" type="connectorAssessmentExpression:assessmentStatementPrototype" />
  <element name="attributeAssessment" type="connectorAssessmentExpression:attributeAssessmentPrototype" />
  <element name="valueAssessment" type="connectorAssessmentExpression:valueAssessmentPrototype" />
  <element name="compoundStatement" type="connectorAssessmentExpression:compoundStatementPrototype" />

</schema>

```

A.13 Módulo *ConnectorCausalExpression*: NCL30ConnectorCausalExpression.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCausalExpression.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL Connector Causal Expression module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- import the definitions in the modules namespaces -->
  <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorCommonPart.xsd"/>

  <simpleType name="conditionRoleUnion">
    <union memberTypes="string connectorCausalExpression:conditionRolePrototype"/>
  </simpleType>

  <simpleType name="conditionRolePrototype">
    <restriction base="string">
      <enumeration value="onBegin" />
      <enumeration value="onEnd" />
      <enumeration value="onPause" />
      <enumeration value="onResume" />
      <enumeration value="onAbort" />
    </restriction>
  </simpleType>

  <simpleType name="maxUnion">
    <union memberTypes="positiveInteger connectorCausalExpression:unboundedString"/>
  </simpleType>

  <simpleType name="unboundedString">
    <restriction base="string">
      <pattern value="unbounded"/>
    </restriction>
  </simpleType>

  <complexType name="simpleConditionPrototype">
    <attribute name="role" type="connectorCausalExpression:conditionRoleUnion" use="required"/>
    <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="optional"/>
    <attribute name="key" type="string" use="optional"/>
    <attribute name="transition" type="connectorCommonPart:transitionPrototype" use="optional"/>
    <attribute name="delay" type="string" use="optional"/>
  </complexType>

```

```

<attribute name="min" type="positiveInteger" use="optional"/>
<attribute name="max" type="connectorCausalExpression:maxUnion" use="optional"/>
<attribute name="qualifier" type="connectorCommonPart:logicalOperatorPrototype" use="optional"/>
</complexType>

<complexType name="compoundConditionPrototype">
  <attribute name="operator" type="connectorCommonPart:logicalOperatorPrototype" use="required"/>
  <attribute name="delay" type="string" use="optional"/>
</complexType>

<simpleType name="actionRoleUnion">
  <union memberTypes="string connectorCausalExpression:actionNamePrototype"/>
</simpleType>

<simpleType name="actionNamePrototype">
  <restriction base="string">
    <enumeration value="start" />
    <enumeration value="stop" />
    <enumeration value="pause" />
    <enumeration value="resume" />
    <enumeration value="abort" />
    <enumeration value="set" />
  </restriction>
</simpleType>

<simpleType name="actionOperatorPrototype">
  <restriction base="string">
    <enumeration value="par" />
    <enumeration value="seq" />
  </restriction>
</simpleType>

<complexType name="simpleActionPrototype">
  <attribute name="role" type="connectorCausalExpression:actionRoleUnion" use="required"/>
  <attribute name="eventType" type="connectorCommonPart:eventPrototype" use="optional"/>
  <attribute name="actionType" type="connectorCausalExpression:actionNamePrototype" use="optional"/>
  <attribute name="delay" type="string" use="optional"/>
  <attribute name="value" type="string" use="optional"/>
  <attribute name="repeat" type="positiveInteger" use="optional"/>
  <attribute name="repeatDelay" type="string" use="optional"/>
  <attribute name="min" type="positiveInteger" use="optional"/>
  <attribute name="max" type="connectorCausalExpression:maxUnion" use="optional"/>
  <attribute name="qualifier" type="connectorCausalExpression:actionOperatorPrototype" use="optional"/>
</complexType>

<complexType name="compoundActionPrototype">
  <choice minOccurs="2" maxOccurs="unbounded">
    <element ref="connectorCausalExpression:simpleAction" />
    <element ref="connectorCausalExpression:compoundAction" />
  </choice>
  <attribute name="operator" type="connectorCausalExpression:actionOperatorPrototype" use="required"/>
  <attribute name="delay" type="string" use="optional"/>
</complexType>

<!-- declare global elements in this module -->
<element name="simpleCondition" type="connectorCausalExpression:simpleConditionPrototype" />

```

```
<element name="compoundCondition" type="connectorCausalExpression:compoundConditionPrototype" />  
<element name="simpleAction" type="connectorCausalExpression:simpleActionPrototype" />  
<element name="compoundAction" type="connectorCausalExpression:compoundActionPrototype" />  
  
</schema>
```

A.14 Módulo *CausalConnector*: NCL30CausalConnector.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30CausalConnector.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Causal Connector module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:causalConnector="http://www.ncl.org.br/NCL3.0/CausalConnector"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/CausalConnector"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="causalConnectorPrototype">  
    <attribute name="id" type="ID" use="required"/>  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="causalConnector" type="causalConnector:causalConnectorPrototype"/>  
</schema>
```

A.15 Módulo *ConnectorBase*: NCL30ConnectorBase.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ConnectorBase.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Connector Base module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:connectorBase="http://www.ncl.org.br/NCL3.0/ConnectorBase"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/ConnectorBase"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="connectorBasePrototype">  
    <attribute name="id" type="ID" use="optional"/>  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="connectorBase" type="connectorBase:connectorBasePrototype"/>  
</schema>
```

A.16 NCL30CausalConnectorFunctionality.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/
NCL30CausalConnectorFunctionality.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL CausalConnectorFunctionality module namespace.
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:connectorCommonPart="http://www.ncl.org.br/NCL3.0/
ConnectorCommonPart"
  xmlns:connectorAssessmentExpression="http://www.ncl.org.br/NCL3.0/
ConnectorAssessmentExpression"
  xmlns:connectorCausalExpression="http://www.ncl.org.br/NCL3.0/
ConnectorCausalExpression"
  xmlns:causalConnector="http://www.ncl.org.br/NCL3.0/
CausalConnector"
  xmlns:causalConnectorFunctionality="http://www.ncl.org.br/NCL3.0/
CausalConnectorFunctionality"
  targetNamespace="http://www.ncl.org.br/NCL3.0/
CausalConnectorFunctionality"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <!-- import the definitions in the modules namespaces -->

  <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCommonPart"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorCommonPart.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorAssessmentExpression"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorAssessmentExpression.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/ConnectorCausalExpression"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30ConnectorCausalExpression.xsd"/>
  <import namespace="http://www.ncl.org.br/NCL3.0/CausalConnector"
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30CausalConnector.xsd"/>

  <!-- ===== -->
  <!-- CausalConnectorFunctionality -->
  <!-- ===== -->
  <element name="connectorParam" type="connectorCommonPart:parameterPrototype"/>

  <!-- extends causalConnector element -->

  <complexType name="causalConnectorType">

```

```

<complexContent>
  <extension base="causalConnector:causalConnectorPrototype">
    <sequence>
      <element ref="causalConnectorFunctionality:connectorParam" minOccurs="0" maxOccurs="unbounded"/>
      <choice>
        <element ref="causalConnectorFunctionality:simpleCondition" />
        <element ref="causalConnectorFunctionality:compoundCondition" />
      </choice>
      <choice>
        <element ref="causalConnectorFunctionality:simpleAction" />
        <element ref="causalConnectorFunctionality:compoundAction" />
      </choice>
    </sequence>
  </extension>
</complexContent>
</complexType>

<!-- extends compoundCondition element -->

<complexType name="compoundConditionType">
  <complexContent>
    <extension base="connectorCausalExpression:compoundConditionPrototype">
      <sequence>
        <choice>
          <element ref="causalConnectorFunctionality:simpleCondition" />
          <element ref="causalConnectorFunctionality:compoundCondition" />
        </choice>
        <choice minOccurs="1" maxOccurs="unbounded">
          <element ref="causalConnectorFunctionality:simpleCondition" />
          <element ref="causalConnectorFunctionality:compoundCondition" />
          <element ref="causalConnectorFunctionality:assessmentStatement" />
          <element ref="causalConnectorFunctionality:compoundStatement" />
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="causalConnector" type="causalConnectorFunctionality:causalConnectorType"
substitutionGroup="causalConnector:causalConnector"/>

<element name="simpleCondition" substitutionGroup="connectorCausalExpression:simpleCondition"/>

<element name="compoundCondition" type="causalConnectorFunctionality:compoundConditionType"
substitutionGroup="connectorCausalExpression:compoundCondition"/>

<element name="simpleAction" substitutionGroup="connectorCausalExpression:simpleAction"/>

<element name="compoundAction" substitutionGroup="connectorCausalExpression:compoundAction"/>

<element name="assessmentStatement"
substitutionGroup="connectorAssessmentExpression:assessmentStatement"/>

<element name="attributeAssessment"
substitutionGroup="connectorAssessmentExpression:attributeAssessment"/>

```

```
<element name="valueAssessment" substitutionGroup="connectorAssessmentExpression:valueAssessment"/>  
  
<element name="compoundStatement"  
substitutionGroup="connectorAssessmentExpression:compoundStatement"/>  
  
</schema>
```

A.17 Módulo *TestRule*: NCL30TestRule.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TestRule.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL TestRule module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRule"
  targetNamespace="http://www.ncl.org.br/NCL3.0/TestRule"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="rulePrototype">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="var" type="string" use="required"/>
    <attribute name="value" type="string" use="required"/>
    <attribute name="comparator" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="eq"/>
          <enumeration value="ne"/>
          <enumeration value="gt"/>
          <enumeration value="gte"/>
          <enumeration value="lt"/>
          <enumeration value="lte"/>
        </restriction>
      </simpleType>
    </attribute>
  </complexType>

  <complexType name="compositeRulePrototype">
    <choice minOccurs="2" maxOccurs="unbounded">
      <element ref="testRule:rule"/>
      <element ref="testRule:compositeRule"/>
    </choice>
    <attribute name="id" type="ID" use="required"/>
    <attribute name="operator" use="required">
      <simpleType>
        <restriction base="string">
          <enumeration value="and"/>
          <enumeration value="or"/>
        </restriction>
      </simpleType>
    </attribute>
  </complexType>

```

```
<complexType name="ruleBasePrototype">  
  <attribute name="id" type="ID" use="optional"/>  
</complexType>  
  
<!-- declare global elements in this module -->  
<element name="rule" type="testRule:rulePrototype"/>  
<element name="compositeRule" type="testRule:compositeRulePrototype"/>  
<element name="ruleBase" type="testRule:ruleBasePrototype"/>  
  
</schema>
```

A.18 Módulo *TestRuleUse*: NCL30TestRuleUse.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TestRuleUse.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL TestRuleUse module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:testRule="http://www.ncl.org.br/NCL3.0/TestRuleUse"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/TestRuleUse"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="bindRulePrototype">  
    <attribute name="constituent" type="IDREF" use="required" />  
    <attribute name="rule" type="string" use="required" />  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="bindRule" type="testRule:bindRulePrototype"/>  
  
</schema>
```

A.19 Módulo *ContentControl*: NCL30ContentControl.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ContentControl.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL ContentControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:contentControl="http://www.ncl.org.br/NCL3.0/ContentControl"
  targetNamespace="http://www.ncl.org.br/NCL3.0/ContentControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="defaultComponentPrototype">
    <attribute name="component" type="IDREF" use="required" />
  </complexType>

  <!-- define the switch element prototype -->

  <complexType name="switchPrototype">
    <choice>
      <element ref="contentControl:defaultComponent" minOccurs="0" maxOccurs="1"/>
    </choice>
    <attribute name="id" type="ID" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="defaultComponent" type="contentControl:defaultComponentPrototype"/>
  <element name="switch" type="contentControl:switchPrototype"/>

</schema>

```

A.20 Módulo *DescriptorControl*: NCL30DescriptorControl.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30DescriptorControl.xsd
Author: TeleMidia Laboratory
Revision: 19/06/2006

Schema for the NCL DescriptorControl module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:descriptorControl="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  targetNamespace="http://www.ncl.org.br/NCL3.0/DescriptorControl"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <complexType name="defaultDescriptorPrototype">
    <attribute name="descriptor" type="IDREF" use="required" />
  </complexType>

  <!-- define the descriptor switch element prototype -->
  <complexType name="descriptorSwitchPrototype">
    <choice>
      <element ref="descriptorControl:defaultDescriptor" minOccurs="0" maxOccurs="1"/>
    </choice>
    <attribute name="id" type="ID" use="required"/>
  </complexType>

  <!-- declare global elements in this module -->
  <element name="defaultDescriptor" type="descriptorControl:defaultDescriptorPrototype"/>
  <element name="descriptorSwitch" type="descriptorControl:descriptorSwitchPrototype"/>

</schema>

```

A.21 Módulo *Timing*: NCL30Timing.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Timing.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Timing module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:timing="http://www.ncl.org.br/NCL3.0/Timing"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/Timing"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <!-- declare global attributes in this module -->  
  
  <!-- define the explicitDur attribute group -->  
  <attributeGroup name="explicitDurAttrs">  
    <attribute name="explicitDur" type="string" use="optional"/>  
  </attributeGroup>  
  
  <!-- define the freeze attribute group -->  
  <attributeGroup name="freezeAttrs">  
    <attribute name="freeze" type="boolean" use="optional"/>  
  </attributeGroup>  
  
</schema>
```

A.22 Módulo *Import*: NCL30Import.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Import.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Import module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:import="http://www.ncl.org.br/NCL3.0/Import"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/Import"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="importBasePrototype">  
    <attribute name="alias" type="ID" use="required"/>  
    <attribute name="region" type="IDREF" use="optional"/>  
    <attribute name="documentURI" type="anyURI" use="required"/>  
  </complexType>  
  
  <complexType name="importNCLPrototype">  
    <attribute name="alias" type="ID" use="required"/>  
    <attribute name="documentURI" type="anyURI" use="required"/>  
  </complexType>  
  
  <complexType name="importedDocumentBasePrototype">  
    <sequence minOccurs="1" maxOccurs="unbounded">  
      <element ref="import:importNCL" />  
    </sequence>  
    <attribute name="id" type="ID" use="optional" />  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="importBase" type="import:importBasePrototype"/>  
  <element name="importNCL" type="import:importNCLPrototype"/>  
  <element name="importedDocumentBase" type="import:importedDocumentBasePrototype"/>  
  
</schema>
```

A.23 Módulo *EntityReuse*: NCL30EntityReuse.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30EntityReuse.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL EntityReuse module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:entityReuse="http://www.ncl.org.br/NCL3.0/EntityReuse"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/EntityReuse"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <attributeGroup name="entityReuseAttrs">  
    <attribute name="refer" type="string" use="optional"/>  
  </attributeGroup>  
  
</schema>
```

A.24 Módulo *ExtendedEntityReuse*: NCL30ExtendedEntityReuse.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30ExtendedEntityReuse.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL ExtendedEntityReuse module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:extendedEntityReuse="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/ExtendedEntityReuse"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <attributeGroup name="extendedEntityReuseAttrs">  
    <attribute name="instance" type="string" use="optional"/>  
  </attributeGroup>  
  
</schema>
```

A.25 Módulo *KeyNavigation*: NCL30KeyNavigation.xsd

```

<!--
XML Schema for the NCL modules

This is NCL
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
See http://www.telemidia.puc-rio.br

Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30KeyNavigation.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006

Schema for the NCL KeyNavigation module namespace.
-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:keyNavigation="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  targetNamespace="http://www.ncl.org.br/NCL3.0/KeyNavigation"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

<simpleType name="colorPrototype">
  <restriction base="string">
    <enumeration value="white" />
    <enumeration value="black" />
    <enumeration value="silver" />
    <enumeration value="gray" />
    <enumeration value="red" />
    <enumeration value="maroon" />
    <enumeration value="fuchsia" />
    <enumeration value="purple" />
    <enumeration value="lime" />
    <enumeration value="green" />
    <enumeration value="yellow" />
    <enumeration value="olive" />
    <enumeration value="blue" />
    <enumeration value="navy" />
    <enumeration value="aqua" />
    <enumeration value="teal" />
  </restriction>
</simpleType>

<!-- declare global attributes in this module -->

<!-- define the keyNavigation attribute group -->
<attributeGroup name="keyNavigationAttrs">
  <attribute name="moveLeft" type="IDREF" use="optional"/>
  <attribute name="moveRight" type="IDREF" use="optional"/>
  <attribute name="moveUp" type="IDREF" use="optional"/>
  <attribute name="moveDown" type="IDREF" use="optional"/>
  <attribute name="focusIndex" type="IDREF" use="optional"/>
  <attribute name="focusBorderColor" type="keyNavigation:colorPrototype" use="optional"/>
  <attribute name="focusBorderWidth" type="string" use="optional"/>
  <attribute name="focusBorderTransparency" type="string" use="optional"/>
  <attribute name="focusScr" type="string" use="optional"/>
  <attribute name="focusSelScr" type="string" use="optional"/>

```

```
<attribute name="selBorderColor" type="keyNavigation:colorPrototype" use="optional"/>
</attributeGroup>

</schema>
```

A.26 Módulo *TransitionBase*: NCL30TransitionBase.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2006 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30TransitionBase.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Transition Base module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:transitionBase="http://www.ncl.org.br/NCL3.0/TransitionBase"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/TransitionBase"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <complexType name="transitionBasePrototype">  
    <attribute name="id" type="ID" use="optional"/>  
  </complexType>  
  
  <!-- declare global elements in this module -->  
  <element name="transitionBase" type="transitionBase:transitionBasePrototype"/>  
</schema>
```

A.27 Módulo *Animation*: NCL30Animation.xsd

```
<!--  
XML Schema for the NCL modules  
  
This is NCL  
Copyright: 2000-2005 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.  
See http://www.telemidia.puc-rio.br  
  
Public URI: http://www.ncl.org.br/NCL3.0/modules/NCL30Animation.xsd  
Author: TeleMidia Laboratory  
Revision: 19/09/2006  
  
Schema for the NCL Timing module namespace.  
-->  
<schema xmlns="http://www.w3.org/2001/XMLSchema"  
  xmlns:animation="http://www.ncl.org.br/NCL3.0/Animation"  
  targetNamespace="http://www.ncl.org.br/NCL3.0/Animation"  
  elementFormDefault="qualified" attributeFormDefault="unqualified" >  
  
  <!-- declare global attributes in this module -->  
  
  <!-- define the animation attribute group -->  
  <attributeGroup name="animationAttrs">  
    <attribute name="duration" type="string" use="optional"/>  
    <attribute name="by" type="string" use="optional"/>  
  </attributeGroup>  
  
</schema>
```

Bibliografia

- [1] ISO 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet N° 1*
- [2] ISO/IEC 13818-6:2001, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC*
- [3] ITU Recommendation J.201:2004, *Harmonization of declarative content format for interactive television applications*
- [4] ARIB STD-B24:2004, *Data coding and transmission specifications for digital broadcasting*
- [5] ACAP, Advanced Application Platform (ACAP), ATSC Standard: Document A/101. Agosto de 2005
- [6] Cascading Style Sheets, Cascading Style Sheets, level 2, Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs. W3C Recommendation 12. Maio de 1998, disponível em <<http://www.w3.org/TR/REC-CSS2>>
- [7] DVB-HTML, Perrot P. DVB-HTML - An Optional Declarative Language within MHP 1.1, EBU Technical Review. 2001
- [8] Namespaces in XML, Namespaces in XML, W3C Recommendation. Janeiro de 1999
- [9] NCM Core, Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 1 – NCM Core, Technical Report, Departamento de Informática PUC-Rio. Maio de 2005, ISSN: 0103-9741. Também disponível em <<http://www.ncl.org.br>>
- [10] NCL Digital TV Profiles, Soares L.F.G; Rodrigues R.F. Part 8 – NCL (Nested Context Language) Digital TV Profiles, Technical Report, Departamento de Informática PUC-Rio, No. 35/06. Outubro de 2006, ISSN: 0103-9741. Também disponível em <<http://www.ncl.org.br>>
- [11] NCL Live Editing Commands, Soares L.F.G; Rodrigues R.F; Costa, R.R.; Moreno, M.F. Part 9 – NCL Live Editing Commands. Technical Report, Departamento de Informática PUC-Rio, No. 36/06. Dezembro de 2006, ISSN: 0103-9741. Também disponível em <<http://www.ncl.org.br>>
- [12] NCL-Lua, Cerqueira, R.; Sant'Anna, F. Nested Context Model 3.0: Part 11 – Lua Scripting Language for NCL, Technical Report, Departamento de Informática PUC-Rio. Maio de 2007, ISSN: 0103-9741.
- [13] NCL Main Profile, Soares L.F.G; Rodrigues R.F; Costa, R.R. Nested Context Model 3.0: Part 6 – NCL (Nested Context Language) Main Profile, Technical Report, Departamento de Informática PUC-Rio. Maio de 2005, ISSN: 0103-9741. Também disponível em <<http://www.ncl.org.br>>
- [14] RDF, Resource Description Framework (RDF) Model and Syntax Specification, Ora Lassila and Ralph R. Swick. W3C Recommendation. 22 de fevereiro de 1999. Disponível em <<http://www.w3.org/TR/REC-rdf-syntax/>>
- [15] SMIL 2.1 Specification, *SMIL 2.1 – Synchronized Multimedia Integration Language – SMIL 2.1 Specification, W3C Recommendation. Dezembro de 2005*
- [16] XHTML 1.0, XHTML™ 1.0 2º Edition – Extensible HyperText Markup Language, W3C Recommendation, Agosto de 2002
- [17] Programming in Lua, Segunda Edição, de Roberto Ierusalimsky et al. Março de 2006, ISBN 85-903798-2-5.