PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 36/06

Nested Context Language 3.0

Part 9 – NCL Live Editing Commands

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romualdo Rezende Costa
Marcio Ferreira Moreno

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 36/06 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Dezembro/2006

Nested Context Language 3.0

Part 9 – NCL Live Editing Commands

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romualdo Rezende Costa
Marcio Ferreira Moreno

Laboratório TeleMídia DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ - 22451-900.

lfgs@inf.puc-rio.br, {rogério, Romualdo, Marcio}@telemidia.puc-rio.br

Abstract. *This technical report describes commands for live editing NCL 3.0 documents. NCL (Nested Context Language) is an XML application language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification, with temporal and spatial synchronization among its media objects.*

Keywords: *live editing, digital TV; middleware; declarative environment; NCL.*

Resumo. *Este relatório técnico descreve os comandos para edição ao vivo de documentos NCL é uma aplicação XML baseada no modelo conceitual NCM (Nested Context Model) para a especificação de documentos hipermídia com sincronismo espacial e temporal entre seus objetos.*

Palavras chave: *edição ao vivo; TV digital; middleware; linguagem declarativa; NCL.*

Nested Context Language 3.0
Part 9 – NCL Live Editing Commands

Laboratório TeleMídia

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rua Marquês de São Vicente, 225, Prédio ITS - Gávea

22451-900 – Rio de Janeiro – RJ – Brasil

http://www.telemidia.puc-rio.br

Table of Contents

Nested Context Language 3.0
Part 9 – NCL Live Editing Commands

Luiz Fernando Gomes Soares
Rogério Ferreira Rodrigues
Romualdo Rezende Costa
Marcio Ferreira Moreno

Laboratório TeleMídia DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ - 22451-900.

lfgs@inf.puc-rio.br, {rogério, Romualdo, Marcio}@telemidia.puc-rio.br

Abstract. *This technical report describes commands for live editing NCL 3.0 documents. NCL (Nested Context Language) is an XML application language based on the NCM (Nested Context Model) conceptual model for hypermedia document specification, with temporal and spatial synchronization among its media objects.*

1. Introduction

The core of an NCL presentation engine is composed by the NCL Formatter and the Private Base Manager [SoRo05].

The NCL Formatter is in charge of receiving an NCL document and controlling its presentation, trying to guarantee that the specified relationships among media objects are respected. The formatter deals with NCL documents that are collected inside a data structure known as *private base* [SoRo05]. NCL documents in a private base can be started, paused, resumed, stopped and can reference each other.

The Private Base Manager is in charge of receiving NCL document editing commands and maintaining the active NCL documents (documents being presented).

Editing commands can come from various means. For example, in a Digital TV environment its is usual to adopt the DSM-CC (Digital Storage Media Command and Control) for carrying editing commands in MPEG-2 TS elementary streams, as discussed in Volume 2 of ISDTV-T Standard 06 [ISDTV-T N06 V02].

This report describes the editing commands for live authoring documents based on NCL. It is organized as follows. Section 2 gives an historical evolution of the NCL versions. Section 3 presents a brief overview of the NCL 3.0 elements. Section 4 introduces the NCL editing commands whose XML Schema is presented in Section 5. Section 6 presents the final remarks.

2. NCL Historical Evolution

The first version of NCL [Anto00, AMRS00] was specified through an XML DTD – Document Type Definition [XML98].

The second version of NCL, named NCL 2.0, was specified using XML Schema [SCHE01]. Following recent trends, from version 2.0 on, NCL has been specified in a modular way, allowing the combination of its modules in language profiles.

Besides the modular structure, NCL 2.0 introduced new facilities to the previous version 1.0, among others:

- definition of hypermedia connectors and connector bases;
- use of hypermedia connectors for link authoring;
- definition of ports and maps for composite nodes, satisfying the document compositionality property;
- definition of hypermedia composite-node templates, allowing the specification of constraints on documents;
- definition of composite-node template bases;
- use of composite-node templates for authoring composite nodes;
- refinement of document specifications with content alternatives, through the <switch> element, grouping a set of alternative nodes;
- refinement of document specifications with presentation alternatives, through the <descriptorSwitch> element, grouping a set of alternative descriptors;
- use of a new spatial layout model.

NCL 2.1 brought some refinements to the previous version: a module for defining cost functions associated with media object duration was introduced; a module aiming at describing the selection rules of <switch> and <descriptorSwitch> elements was defined; and refinements in some NCL modules were made, mainly in the XTemplate module.

NCL 2.2 made minor refinements in some NCL 2.1 modules, concerning their element definitions, and introduced a different approach in defining NCL modules and profiles.

NCL 2.3 introduced two new modules for supporting base and entity reuse, and refined the definition of some elements in order to support the new features.

NCL 2.4 reviewed and refined the reuse support introduced in version 2.3, and the specification of the switch and descriptor switch elements. This version also split the Timing module introduced by NCL 2.1, creating a new module to encapsulate issues related with time-scaling operations (elastic time computation using temporal cost functions) in hypermedia documents.

The NCL 3.0 edition revised some functionalities contained in NCL 2.4. NCL 3.0 is more specific regarding some attribute values and introduces some SMIL modules to NCL profiles, specially aiming at the Digital TV domain. NCL 3.0 also reviewed the hypermedia connector specification in order to have a more concise notation.

NCM is the model underlying NCL. However, in its present version 3.0, NCL does not reflect all NCM 3.0 facilities yet. In order to understand NCL facilities, it is necessary to

understand the NCM concepts. With the aim of offering a scalable hypermedia model, with characteristics that can be progressively incorporated in hypermedia system implementations, the NCM and NCL family was divided in several parts. The Nested Context Model is composed by Parts 1, 2, 3, and 4 of the collection:

- Part 1: NCM Core – concerned with the main model entities, which should be present in all NCM implementations[1].
- Part 2: NCM Virtual Entities – concerned mainly with the definition of virtual anchors, nodes and links.
- Part 3: NCM Version Control – concerned with model entities and attributes to support versioning.
- Part 4: NCM Cooperative Work – concerned with model entities and attributes to support cooperative document handling.

The Nested Context Language specification is composed by Parts 5, 6, 7, 8 and 9 of the collection:

- Part 5: NCL (Nested Context Language) Full Profile – concerned with the definition of an XML application language for authoring and exchanging NCM-based documents, using all NCL modules, including those for the definition and use of templates, and also the definition of constraint connectors, composite-connectors, temporal cost functions, transition effects and metainformation characterization.
- Part 6: NCL (Nested Context Language) Main Profile – concerned with the definition of an XML application language for authoring and exchanging NCM-based documents, without the modules for template use and definition; for constraint connector and composite-connector specification; for temporal cost function definition; for transition effects definition and for metainformation characterization.
- Part 7: NCL (Nested Context Language) XConnector Profile Family– concerned with the definition of an XML application languages for authoring connector bases. One profile is defined for authoring causal connectors, another one for authoring causal and constraint connectors, and a third one for authoring both simple and composite connectors.
- Part 8: NCL (Nested Context Language) Digital TV Profiles – concerned with the definition of an XML application languages for authoring documents aiming at the digital TV domain. Two profiles are defined: the Enhanced Digital TV (EDTV) profile and the Basic Digital TV (BDTV) profile.
- Part 9: NCL Live Editing Commands– concerned with editing commands used for live authoring applications based on NCL.

In order to understand NCL, the reading of Part 1: NCM Core is recommended.

[1] It is also possible to have NCM implementations that ignore some of the basic entities, but this is not so relevant as to deserve a minimum-core definition.

3. Overview of NCL Elements

NCL is an XML application that follows the modularization approach. The modularization approach has been used in several W3C language recommendations. A *module* is a collection of semantically-related XML elements, attributes, and attribute's values that represents a unit of functionality. Modules are defined in coherent sets. A *language profile* is a combination of modules. Several NCL profiles have been defined, among them those defined by Parts 5, 6, 7, and 8 of the NCL collection presented in Section 2. Of special interest are the profiles defined for Digital TV, the EDTVProfile (*Enhanced Digital TV Profile*) and the BDTVProfile (*Basic Digital TV Profile*). This section briefly describes the elements that compose these profiles. The complete definition of the NCL 3.0 modules for these profiles, using XML Schemas, is presented in [SoRo06]. Any ambiguity found in this text may be clarified by consulting the XML Schemas.

The basic NCL structure module defines the root element, called <ncl>, and its children elements, the <head> element and the <body> element, following the terminology adopted by other W3C standards.

The <head> element can have <importedDocumentBase>, <ruleBase>, <transitionBase> <regionBase>, <descriptorBase>, <connectorBase>, <meta>, and <metadata> elements as its children.

The <body> element can have <port>, <attribute>, <media>, <context>, <switch>, and <link> elements as its children. The <body> element is treated as an NCM context node. In NCM [SoRo05], the conceptual data model of NCL, a node can be a context, a switch or a media object. Context nodes can contain other NCM nodes and links. Switch nodes contain other NCM nodes. NCM nodes are represented by corresponding NCL elements.

The <media> element defines a media object specifying its type and its content location. NCL only defines how media objects are structured and related, in time and space. As a glue language, it does not restrict or prescribe the media-object content types. However, some types are defined by the language. For example: the "application/x-NCL-settings" type, specifying an object whose attributes are global variables defined by the document author or are reserved environment variables that can be manipulated by the NCL document processing; and the "application/x-NCL-time" type, specifying a special <media> element whose content is the Greenwich Mean Time (GMT).

The <context> element is responsible for the definition of context nodes. An NCM context node is a particular type of NCM composite node and is defined as containing a set of nodes and a set of links [SoRo05]. Like the <body> element, a <context> element can have <port>, <attribute>, <media>, <context>, <switch>, and <link> elements as its children.

The <switch> element allows the definition of alternative document nodes (represented by <media>, <context>, and <switch> elements) to be chosen during presentation time. Test rules used in choosing the switch component to be presented are defined by <rule> or <compositeRule> elements that are grouped by the <ruleBase> element, defined as a child element of the <head> element.

The NCL Interfaces functionality allows the definition of node interfaces that are used in relationships with other node interfaces. The <area> element allows the definition of content anchors representing spatial portions, temporal portions, or temporal and spatial portions of a media object (<media> element) content. The <port> element specifies a composite node (<context>, <body> or <switch> element) port with its respective mapping to an interface of one of its child components. The <attribute> element is used for defining a node attribute or a group of node attributes as one of the node's interfaces. The <switchPort> element allows the creation of <switch> element interfaces that are mapped to a set of alternative interfaces of the switch's internal nodes.

The <descriptor> element specifies temporal and spatial information needed to present each document component. The element can refer a <region> element to define the initial position of the <media> element (that is associated with the <descriptor> element) presentation is some output device. The definition of <descriptor> elements must be included in the document head, inside the <descriptorBase> element, which specifies the set of descriptors of a document. Also inside the document <head> element, the <regionBase> element defines a set of <region> elements, each of which may contain another set of nested <region> elements, and so on, recursively; regions define device areas (e.g. screen windows) and are referenced by <descriptor> elements, as previously mentioned.

A <causalConnector> element represents a relation that can be used for creating <link> elements in documents. In a causal relation, a condition must be satisfied in order to trigger an action. A <link> element binds (through its <bind> elements) a node interface with connector roles, defining a spatio-temporal relationship among NCL objects (represented by <media>, <context>, <body> or <switch> elements).

The <descriptorSwitch> element contains a set of alternative descriptors to be associated with an object. Analogous to the <switch> element, the <descriptorSwitch> choice is done during the document presentation, using test rules defined by <rule> or <compositeRule> elements.

In order to allow an entity base to incorporate another already-defined base, the <importBase> element can be used. Additionally, an NCL document can be imported through the <importNCL> element. The <importedDocumentBase> element specifies a set of imported NCL documents, and must also be defined as a child element of the <head> element.

Some important NCL element's attributes are defined in other NCL modules. The EntityReuse module allows an NCL element to be reused. This module defines the *refer* attribute, which refers to an element URI that will be reused. Only <media>, <context>, <body> and <switch> can be reused. The KeyNavigation module provides the extensions necessary to describe focus movement operations using a control device like a remote control. Basically, the module defines attributes that can be incorporated by <descriptor> elements.

Some SMIL functionalities are also incorporated by NCL. The <transition> element and some transition attributes are defined in the SMIL BasicTransitions module and the SMIL TransitionModifiers module. The NCL <transitionBase> element specifies a set of

transition effects, defined by <transition> elements, and must be defined as a child element of the <head> element.

Finally, the SMIL MetaInformation module is also incorporated. Meta-information does not contain content information that is used or display during a presentation. Instead, it contains information about content that is used or displayed. The Metainformation module contains two elements that allow describing NCL documents. The <meta> element specifies a single property/value pair. The <metadata> element contains information that is also related to meta-information of the document. It acts as the root element of an RDF tree: RDF element and its sub-elements (for more details, refer to W3C metadata recommendations [RDF99]).

4. NCL Editing Commands

The Private Base Manager is in charge of receiving NCL document editing commands and maintaining the active NCL documents (documents being presented). XML-based *NCL editing commands* are divided in three subsets.

The first subset focuses on the private base activation and deactivation (openBase, activateBase, deactivateBase, saveBase, and closeBase commands).

NCL documents can be added to a private base and then can be started, paused, resumed, stopped and removed, through well defined commands that compose the second subset.

The third subset defines commands for live editing, allowing NCL elements to be added and removed, and allowing values to be set to NCL <attribute> elements. *Add* commands always have NCL elements as their arguments. Whether the specified element already exists or not, document consistency must be maintained by the NCL formatter, in the sense that all element attributes classified as required must be defined. The elements are defined using a XML-based syntax notation defined in Section 5, with the exception of the *addInterface* command: the *begin* attribute of an <area> element can receive the "now" value, specifying the current NPT (Normal Play Time) of the nodeId.

Table 4.1 shows the *command strings* and surrounded by round brackets their arguments. He table also gives the semantics of each command.

Table 4.1 – Editing commands for NCL Private Base Manager

Command String	Command Tag	Description
openBase (baseId)	0x00	Opens a new private base. If the private base does not exist, a new base is created.
activateBase (baseId)	0x01	Switches on an opened private base.
deactivateBase (baseId)	0x02	Switches off an opened private base.
saveBase (baseId, device)	0x03	Saves all private base content into a persistent storage device (if available).
closeBase (baseId)	0x04	Closes the private base and dispose all private base content.
addDocument (baseId, {uri, ior}+)	0x05	Adds an NCL document to a private base. The NCL document must be sent in file system structures through a transport system; the pair {uri,ior} relates a file system path in the author side with its respective location in a transport system.

Command String	Command Tag	Description
removeDocument (baseId, documentId)	0x06	Removes an NCL document from a private base.
startDocument (baseId, documentId, interfaceId)	0x07	Starts playing an NCL document in a private base, beginning the presentation from a specific document interface.
stopDocument (baseId, documentId)	0x08	Stops the presentation of an NCL document in a private base. All document events that are occurring must be stopped.
pauseDocument (baseId, documentId)	0x09	Pauses the presentation of an NCL document in a private base. All document events that are occurring must be paused.
resumeDocument (baseId, documentId)	0x0A	Resumes the presentation of an NCL document in a private base. All previously document events that were paused by the pauseDocument editing command must be resumed.
addRegion (baseId, documentId, regionId, xmlRegion)	0x0B	Adds a <region> element as a child of another <region> in the <regionBase> or as a child of the <regionBase> (regionId=”null”) of an NCL document in a private base.
removeRegion (baseId, documentId, regionId)	0x0C	Removes a <region> element from the <regionBase> of an NCL document in a private base.
addRegionBase (baseId, documentId, xmlRegionBase)	0x0D	Adds a <regionBase> element to the <head> element of an NCL document in a private base. The XML specification of the regionBase is sent in the object carousel as a file system; the *xmlRegionBase* parameter is just a reference to this content in the carousel.
removeRegionBase (baseId, documentId, regionBaseId)	0x0E	Removes a <regionBase> element from the <head> element of an NCL document in a private base
addRule (baseId, documentId, xmlRule)	0x0F	Adds a <rule> element to the <ruleBase> of an NCL document in a private base.
removeRule (baseId, documentId, ruleId)	0x10	Removes a <rule> element from the <ruleBase> of an NCL document in a private base.
addRuleBase (baseId, documentId, xmlRuleBase)	0x11	Adds a <ruleBase> element to the <head> element of an NCL document in a private base. The XML specification of the ruleBase is sent in the object carousel as a file system; the *xmlRuleBase*

Command String	Command Tag	Description
		parameter is just a reference to this content in the carousel.
removeRuleBase (baseId, documentId, ruleBaseId)	0x12	Removes a <ruleBase> element from the <head> element of an NCL document in a private base
addConnector (baseId, documentId, xmlConnector)	0x13	Adds a <connector> element to the <connectorBase> of an NCL document in a private base.
removeConnector (baseId, documentId, connectorId)	0x14	Removes a <connector> element from the <connectorBase> of an NCL document in a private base.
addConnectorBase (baseId, documentId, xmlConnectorBase)	0x15	Adds a <connectorBase> element to the <head> element of an NCL document in a private base. The XML specification of the connectorBase is sent in the object carousel as a file system; the *xmlConnectorBase* parameter is just a reference to this content in the carousel.
removeConnectorBase (baseId, documentId, connectorBaseId)	0x16	Removes a <connectorBase> element from the <head> element of an NCL document in a private base
addDescriptor (baseId, documentId, xmlDescriptor)	0x17	Adds a <descriptor> element to the <descriptorBase> of an NCL document in a private base.
removeDescriptor (baseId, documentId, descriptorId)	0x18	Removes a <descriptor> element from the <descriptorBase> of an NCL document in a private base.
addDescriptorSwitch (baseId, documentId, xmlDescriptorSwitch)	0x19	Adds a <descriptorSwitch> element to the <descriptorBase> of an NCL document in a private base. The XML specification of the descriptoSwitch is sent in the object carousel as a file system; the *xmlDescriptorSwitch* parameter is just a reference to this content.
removeDescriptorSwitch (baseId, documentId, descriptorSwitchId)	0x1A	Removes a <descriptorSwitch> element from the <descriptorBase> of an NCL document in a private base.
addDescriptorBase (baseId, documentId, xmlDescriptorBase)	0x1B	Adds a <descriptorBase> element to the <head> element of an NCL document in a private base. The XML specification of the descriptorBase is sent in the object carousel as a file system; the *xmlDescriptorBase* parameter is just a reference to this content in the carousel.

Command String	Command Tag	Description
removeDescriptorBase (baseId, documentId, descriptorBaseId)	0x1C	Removes a <descriptorBase> element from the <head> element of an NCL document in a private base.
addTransition (baseId, documentId, xmlTransition)	0x1D	Adds a <transition> element to the <transitionBase> of an NCL document in a private base.
removeTransition (baseId, documentId, transitionId)	0x1E	Removes a <transition> element from the <transitionBase> of an NCL document in a private base.
addTransitionBase (baseId, documentId, xmlTransitionBase)	0x1F	Adds a <transitionBase> element to the <head> element of an NCL document in a private base. The XML specification of the transitionBase is sent in the object carousel as a file system; the *xmlTransitionBase* parameter is just a reference to this content in the carousel.
removeTransitionBase (baseId, documentId, transitionBaseId)	0x20	Removes a <transitionBase> element from the <head> element of an NCL document in a private base
addImportBase (baseId, documentId, docBaseId, xmlImportBase)	0x21	Adds a <importBase> element to the base (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, or <connectorBase> element) of an NCL document in a private base.
removeImportBase (baseId, documentId, docBaseId, importBaseId)	0x22	Removes a <importBase> element from the base (<regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, or <connectorBase> element) of an NCL document in a private base.
addImportedDocumentBase (baseId, documentId, xmlImportedDocumentBase)	0x23	Adds a <importedDocumentBase> element to the <head> element of an NCL document in a private base.
removeImportedDocumentBase (baseId, documentId, importedDocumentBaseId)	0x24	Removes a <importedDocumentBase> element from the <head> element of an NCL document in a private base.
addImportNCL (baseId, documentId, xmlImportNCL)	0x25	Adds a <importNCL> element to the <importedDocumentBase > element of an NCL document in a private base.
removeImportNCL (baseId, documentId, importNCLId)	0x26	Removes a <importNCL> element from the <importedDocumentBase > element of an NCL document in a private base
addNode (baseId, documentId,	0x27	Adds a node (<media>, <context>, or <switch>

Command String	Command Tag	Description
compositeId, {uri, ior}+)		element) to a composite node (<body>, <context>, or <switch> element) of an NCL document in a private base. The XML specification of the node and its media content must be sent in file system structures through a transport system; the pair {uri,ior} relates a file system path in the author side with its respective location in a transport system.
removeNode(baseId, documentId, compositeId, nodeId)	0x28	Removes a node (<media>, <context>, or <switch> element) from a composite node (<body>, <context>, or <switch> element) of an NCL document in a private base.
addInterface (baseId, documentId, nodeId, xmlInterface)	0x29	Adds an interface (<port>, <area>, <attribute>, or <switchPort>) to a node (<media>, <body>, <context>, or <switch> element)of an NCL document in a private base.
removeInterface (baseId, documentId, nodeId, interfaceId)	0x2A	Removes an interface (<port>, <area>, <attribute>, or <switchPort>) from a node (<media>, <body>, <context>, or <switch> element)of an NCL document in a private base.
addLink (baseId, documentId, compositeId, xmlLink)	0x2B	Adds a <link> element to a composite node (<body>, <context>, or <switch> element) of an NCL document in a private base.
removeLink (baseId, documentId, compositeId, linkId)	0x2C	Removes a <link> element from a composite node (<body>, <context>, or <switch> element) of an NCL document in a private base.
setAttributeValue(baseId, documentId, nodeId, attributeId, value)	0x2D	Sets the value for an attribute whose *id* is attributeId. The attributeId must identify an <attribute> or a <switchPort> element. The <attribute> or <switchPort> must belongs to a node (<body>, <context>, <switch> or <media> element) of an NCL document in a private base.

The identifiers used in the commands are defined in 4.2

Table 4.2 – Identifiers used in editing commands

Identifiers	Definition
baseId	Broadcast channel identifiers specified by the ISDTV-T.

Identifiers	Definition
documentId	The *id* attribute of an <ncl> element of an NCL document.
regionId	The *id* attribute of a <region> element of an NCL document.
ruleId	The *id* attribute of a <rule> element of an NCL document.
connectorId	The *id* attribute of a <connector> element of an NCL document.
descriptorId	The *id* attribute of a <descriptor> element of an NCL document.
descriptorSwitchId	The *id* attribute of a <descriptorSwitch> element of an NCL document.
transitionId	The *id* attribute of a <transition> element of an NCL document.
regionBaseId	The *id* attribute of a <regionBase> element of an NCL document.
ruleBaseId	The *id* attribute of a <ruleBase> element of an NCL document.
connectorBaseId	The *id* attribute of a <connectorBase> element of an NCL document.
descriptorBaseId	The *id* attribute of a <descriptorBase> element of an NCL document.
transitionBaseId	The *id* attribute of a <transitionBase> element of an NCL document.
docBaseId	The *id* attribute of a <regionBase>, <ruleBase>, <connectorBase>, <descriptorBase>, or <transitionBase> element of an NCL document.
importBaseId	The *id* attribute of an <importBase> element of an NCL document.
importedDocumentBaseId	The *id* attribute of a <importedDocumentBase> element of an NCL document.
ImportNCLId	The *id* attribute of an <importNCL> element of an NCL document.

Identifiers	Definition
compositeID	The *id* attribute of a <body>, <context> or <switch> element of an NCL document.
nodeId	The *id* attribute of a <body>, <context>, <switch> or <media> element of an NCL document.
interfaceId	The *id* attribute of a <port>, <area>, <attribute> or <switchPort> element of an NCL document.
linkId	The *id* attribute of a <link> element of an NCL document.
attributeId	The *id* attribute of an <attribute> or <switchPort> element of an NCL document.

5. Command Parameters XML Schemas

NCL entities used in editing commands must be a document in conformance with the NCL 3.0 Command profile defined by the XML Schema that follows.

Note that different from NCL documents, several ncl elements can be the root element in the XML command parameters.

NCL30EdCommand.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/NCL30EdCommand.xsd
Author: TeleMidia Laboratory
Revision: 19/09/2006
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:attributeAnchor="http://www.ncl.org.br/NCL3.0/AttributeAnchor"
  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: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/EdCommandProfile"
  targetNamespace="http://www.ncl.org.br/NCL3.0/EdCommandProfile"
  elementFormDefault="qualified" attributeFormDefault="unqualified" >

  <!-- import the definitions in the modules namespaces -->
  <import namespace="http://www.ncl.org.br/NCL3.0/AttributeAnchor"
```

```
    schemaLocation="http://www.ncl.org.br/NCL3.0/modules/
NCL30AttributeAnchor.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/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/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"/>

  <!-- ========================================================== -->
  <!--EditingCommand                                              -->
  <!-- ========================================================== -->
  <!--defines the command element -->

  <!--This is a pseudo-element, only defined to show the elements that
can be used in the root of the command parameters XML document-->

  <!--
  <complexType name="commandType">
        <choice minOccurs="1" maxOccurs="1">
          <element ref="profile:ncl"/>
          <element ref="profile:region"/>
          <element ref="profile:rule"/>
          <element ref="profile:connector"/>
          <element ref="profile:descriptor"/>
          <element ref="profile:descriptorSwitch"/>
          <element ref="profile:transition"/>
          <element ref="profile:regionBase"/>
          <element ref="profile:ruleBase"/>
          <element ref="profile:connectorBase"/>
          <element ref="profile:descriptorBase"/>
          <element ref="profile:transitionBase"/>
          <element ref="profile:importBase"/>
          <element ref="profile:importedDocumentBase"/>
          <element ref="profile:importNCL"/>
          <element ref="profile:media"/>
          <element ref="profile:context"/>
          <element ref="profile:switch"/>
          <element ref="profile:port"/>
          <element ref="profile:area"/>
          <element ref="profile:attribute"/>
          <element ref="profile:switchPort"/>
          <element ref="profile:link"/>
        </choice>
  </complexType>
  <element name="command" type="profile:commandType"/>
-->

  <!-- ========================================================== -->
  <!-- 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: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:attribute"/>
    </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"/>
      <element ref="profile:attribute"/>
    </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="descriptor:descriptorAttrs"/>
        <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">
         <attribute name="now" type="string" use="optional"/>
      </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"/>

  <!-- ============================================================ -->
  <!-- AtributeAnchor                                               -->
  <!-- ============================================================ -->
  <!-- extends attribute element -->

  <complexType name="attributeAnchorType">
    <complexContent>
      <extension base="attributeAnchor:attributeAnchorPrototype">
      </extension>
    </complexContent>
  </complexType>
```

```
  <element name="attribute" type="profile:attributeAnchorType"
substitutionGroup="attributeAnchor:attribute"/>

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

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

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

6. Final Remarks

In order to offer a scalable hypermedia model, with characteristics that can be progressively incorporated in hypermedia system implementations, NCM was divided in several parts. This technical report deals with the editing commands used for live authoring NCL documents, which comprises Part 9: NCL Live Editing Commands.

References

[Anto00] Antonacci M.J. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial. **Master Dissertation, Departamento de Informática, PUC-Rio**, April 2000.

[AMRS00] Antonacci M.J., Muchaluat-Saade D.C., Rodrigues R.F., Soares L.F.G. NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia na Web, **VI Simpósio Brasileiro de Sistemas Multimídia e Hipermídia - SBMídia2000**, Natal, Rio Grande do Norte, June 2000.

[ISDTV-T N06 V02] Volume 2 of ISDTV-T Standard 06. *ISDTV-T Fórum*. December 2006.

[RDF99] Resource Description Framework (RDF) Model and Syntax Specification, Ora Lassila and Ralph R. Swick. W3C Recommendation, 22 February 1999. Available at http://www.w3.org/TR/REC-rdf-syntax/

[SCHE01] XML Schema Part 0: Primer, **W3C Recommendation**, in http://www.w3.org/TR/xmlschema-0/, May 2001.

[SoRo05] Soares L.F.G; Rodrigues R.F. Nested Context Model 3.0: Part 1 – NCM Core, **Technical Report, Departamento de Informática PUC-Rio**, May 2005, ISSN: 0103-9741.

[SoRo06] Soares L.F.G; Rodrigues R.F. Nested Context Language 3.0: Part 8 – NCL Digital TV Profiles, **Technical Report, Departamento de Informática PUC-Rio**, October 2006, ISSN: 0103-9741.

[XML98] Bray T., Paoli J., Sperberg-McQueen C.M., Maler E. Extensible Markup Language (XML) 1.0 (Second Edition), **W3C Recommendation**, in http://www.w3.org/TR/REC-xml, February 1998.