

I n t e r n a t i o n a l T e l e c o m m u n i c a t i o n U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.761

(11/2014)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

IPTV multimedia services and applications for IPTV –
IPTV multimedia application frameworks

Nested context language (NCL) and Ginga-NCL

Recommendation ITU-T H.761

ITU-T

ITU-T H-SERIES RECOMMENDATIONS
AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Telepresence	H.420–H.429
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Ubiquitous sensor network applications and Internet of Things	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SERVICES AND APPLICATIONS	
Interoperability compliance testing of personal health systems (HRN, PAN, LAN and WAN)	H.820–H.859
Multimedia e-health data exchange services	H.860–H.869

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T H.761

Nested context language (NCL) and Ginga-NCL

Summary

Recommendation ITU-T H.761 gives the specification of the nested context language (NCL) and of an NCL presentation environment called Ginga-NCL to provide interoperability and harmonization among Internet protocol television (IPTV) multimedia application frameworks.

NCL is a declarative glue language that holds media object presentations synchronized in time and space, no matter the types of the media objects. Ginga-NCL is an NCL presentation engine built as a component of a digital television (DTV) middleware.

This Recommendation includes an electronic attachment containing NCL 3.1 module schemas used in the enhanced DTV profile.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T H.761	2009-04-29	16	11.1002/1000/9715
2.0	ITU-T H.761 v2	2011-06-13	16	11.1002/1000/11307
3.0	ITU-T H.761	2014-11-29	16	11.1002/1000/12237

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2015

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Table of Contents

		Page
1	Scope.....	1
2	References.....	1
3	Definitions	2
	3.1 Terms defined elsewhere	2
	3.2 Terms defined in this Recommendation.....	3
4	Abbreviations and acronyms	5
5	NCL and Ginga-NCL	6
6	Ginga-NCL harmonization with other IPTV declarative environments.....	7
7	NCL: XML application declarative language for multimedia presentations.....	8
	7.1 Identifiers for NCL 3.1 module and language profiles.....	9
	7.2 NCL modules.....	11
	7.3 NCL language profiles for IPTV	62
8	Media objects in NCL presentations.....	69
	8.1 The media player API.....	70
	8.2 Expected behaviour of basic media players	72
	8.3 Expected behaviour of declarative hypermedia players in NCL applications.....	76
	8.4 Expected behaviour of imperative-object media players in NCL applications.....	80
	8.5 Expected behaviour of media players after actions applied to composite objects.....	84
	8.6 Relation between the presentation-event state machine of a node and the presentation-event state machine of its parent-composite node	85
9	NCL editing commands.....	86
	9.1 Private bases	86
	9.2 Command parameters XML schemas	97
	9.3 NCL editing commands in Ginga-NCL	98
10	Lua imperative objects in NCL presentations	102
	10.1 Lua language – functions removed from the standard Lua library	102
	10.2 Execution model.....	102
	10.3 Additional modules	102
11	Security API.....	124
	11.1 Security control in Ginga-NCL	124
	11.2 The NCLua security API.....	125
	Annex A – NCL 3.1 module schemas used in the enhanced DTV profile	129
	Annex B – XML schemas for the media player API	130
	B.1 Data types for the parameters used in the media player API	130
	B.2 Input control API.....	134

	Page
Appendix I – Ginga architecture	135
Appendix II – An NCL example.....	137
Bibliography.....	138

Electronic attachment: NCL 3.1 module schemas used in the enhanced DTV profile.

iv

Introduction

Nested context language (NCL) is a declarative XML-based language initially designed for hypermedia document specification for the web. The intrinsic ability of NCL for easily defining spatio-temporal synchronization among media assets, including viewer interactions, in addition to the language's flexibility, reusability, multi device support and application content adaptability, make it an outstanding solution for IPTV systems. NCL is also the declarative language used in the Japanese Brazilian terrestrial DTV standard (ISDB-T).

NCL is a glue language that holds media objects together in a multimedia presentation, no matter which object types they are. In this sense, media objects may be image objects (JPEG, PNG, etc.), video objects (MPEG, MOV, etc.), audio objects (MP3, WMA, etc.), text objects (TXT, PDF, etc.), imperative objects (with Lua code, etc.), other declarative objects (HTML, LIME, SVG, MHEG, nested NCL applications, etc.), etc. Which media objects are supported depends on which media players are embedded in the NCL player (part of the Ginga-NCL environment). As an example, NCL treats an HTML document as one of its possible media objects. In this way, NCL does not substitute but instead embeds HTML-based documents. The same reasoning applies to other media content and multimedia content objects and also to objects with content coded in any computer language.

Ginga-NCL is an NCL presentation engine built as a component of a DTV middleware. An open source reference implementation of Ginga-NCL is available under the GPLv2 licence (http://www.gingancl.org.br/index_en.html). This reference implementation was developed in such a way so that it can easily integrate a variety of media-object players for audio, video, image, text, etc., including imperative execution engines and other declarative language players.

A special NCL object type defined in Ginga-NCL is NCLua, an imperative media-object with Lua code as its content. Because of its simplicity, efficiency and powerful data description syntax, Lua is considered the default scripting language for Ginga-NCL. The Lua engine is small and written in ANSI/C, making it easily portable to several hardware platforms. The Lua engine is also distributed as free software under the Massachusetts Institute of Technology (MIT) licence (<http://www.lua.org/license.html>).

Copyright International Telecommunication Union/ITU Telecommunication Sector
Provided by IHS Markit under license with ITU-T
No reproduction or networking permitted without license from IHS

Copyright International Telecommunication Union/ITU Telecommunication Sector

Recommendation ITU-T H.761

Nested context language (NCL) and Ginga-NCL

1 Scope

This Recommendation¹ specifies the nested context language (NCL) and an NCL presentation environment, called Ginga-NCL, to provide interoperability and harmonization among IPTV multimedia application frameworks. To provide global standard IPTV services, it is foreseeable that a combination of different standard multimedia application frameworks will be used. Therefore, this Recommendation specifies the nested context language, as one of those standards that compose the multimedia application frameworks, to provide interoperable use of IPTV services. Ginga-NCL is an NCL presentation environment that integrates NCL and Lua players. NCL and Lua frameworks can be used in other declarative environments, but if they are used together they shall follow the Ginga-NCL specification.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

- [ITU-T H.222.0] Recommendation ITU-T H.222.0 (2014) | ISO/IEC 13818-1:2007, *Information technology – Generic coding of moving pictures and associated audio information: Systems*.
- [ITU-T H.750] Recommendation ITU-T H.750 (2008), *High-level specification of metadata for IPTV services*.
- [ITU-T J.200] Recommendation ITU-T J.200 (2010), *Worldwide common core – Application environment for digital interactive television services*.
- [ITU-T X.509] Recommendation ITU-T X.509 (2012), *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- [ITU-T Y.1901] Recommendation ITU-T Y.1901 (2009), *Requirements for the support of IPTV services*.
- [IETF RFC 1421] IETF RFC 1421 (1993), *Privacy Enhancement for Internet Electronic Mail*.
- [IETF RFC 6726] IETF RFC 6726 (2012), *FLUTE – File Delivery over Unidirectional Transport*.
- [ISO/IEC 8859-1] ISO/IEC 8859-1:1998, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*.
- [ISO/IEC 13818-6] ISO/IEC 13818-6:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC*.

¹ This Recommendation includes an electronic attachment containing NCL 3.1 module schemas used in the enhanced DTV profile.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 application [ITU-T J.200]: Information that expresses a specific set of observable behaviour.

3.1.2 application environment (environment) [ITU-T J.200]: The context or software environment in which an application is processed.

3.1.3 application programming interface (API) [ITU-T J.200]: Software libraries that provide uniform access to system services.

3.1.4 character [ITU-T J.200]: Specific "letter" or other identifiable symbol, e.g., "A".

3.1.5 data carousel [ITU-T J.200]: A transmission scheme defined in [ISO/IEC 13818-6], with which data is transmitted repetitively. It can be used for downloading various data in broadcasting. It is the scheme of the DSM-CC User-to-Network Download protocol that embodies the cyclic transmission of data.

3.1.6 declarative application [ITU-T J.200]: An application which primarily makes use of declarative information to express its behaviour; an XML document instance is an example of a declarative application.

3.1.7 declarative application environment [ITU-T J.200]: An environment that supports the processing of declarative applications; an XML user agent (browser) is an example of a declarative application environment.

3.1.8 digital storage media command and control (DSM-CC) [ITU-T J.200]: A control method defined in [ISO/IEC 13818-6], which provides access to files and streams for digital interactive services.

3.1.9 electronic program guide (EPG) [b-ITU-T Y.1901]: A structured set of data, intended to provide information on available content that may be accessed by end-users..

3.1.10 element [b-ITU-T M.3030]: A logical data structure within an XML document; start tags and end tags define the beginning and end of an element. Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by a case-sensitive name, sometimes called its "generic identifier" (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value.

3.1.11 elementary stream (ES) [ITU-T H.222.0]: A generic term for one of the coded video, coded audio or other coded bit streams in PES packets. One elementary stream is carried in a sequence of PES packets with one, and only one, stream id.

3.1.12 locator [ITU-T J.200]: A link, expressed in the syntax provided in RFC 2396, which provides a reference to an application or resource.

3.1.13 markup language [ITU-T J.200]: A formalism that describes a document's structure, appearance, or other aspects. An example of a markup language is XHTML.

3.1.14 normal play time (NPT) [ITU-T J.200]: The absolute temporal coordinates that represent the position in a stream at which an event occurs.

3.1.15 packet identifier (PID) [ITU-T H.222.0]: A unique integer value used to identify elementary streams of a program in a single or multi-program transport stream as described in clause 2.4.3 of [ITU-T H.222.0].

3.1.16 persistent storage [ITU-T J.200]: Memory available that can be read/written to by an application and may outlive the application's own life. Persistent storage may be volatile or non-volatile.

3.1.17 plug-in [ITU-T J.200]: A set of functionalities which can be added to a generic platform in order to provide additional functionality.

3.1.18 presentation engine [ITU-T J.200]: A subsystem in a receiver that evaluates and presents declarative applications consisting of content, such as audio, video, graphics, and text primarily based on presentation rules defined in the presentation engine. A presentation engine also responds to formatting information, or "markup", associated with the content, to user inputs, and to script statements, which control presentation behaviour and initiate other processes in response to user input and other events.

3.1.19 receiver platform (platform) [ITU-T J.200]: The receiver's hardware, operating system and native software libraries of the manufacturer's choice.

3.1.20 resource [ITU-T J.200]: A network data object or a service that is uniquely identified in a network, e.g., an application resource or environment resource.

3.1.21 service information (SI) [ITU-T J.200]: Data that describes programs and services.

3.1.22 uniform resource identifier (URI) [ITU-T J.200]: An addressing method to access a resource in local storage or on the Internet.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 application life-cycle: The time period from the moment an application is loaded until the moment it is destroyed.

3.2.2 author: The person who writes NCL documents.

3.2.3 authoring tool: A tool to help authors create NCL documents.

3.2.4 attribute: A parameter that represents the character of a property.

3.2.5 companion application: An independent application or a module of a distributed multi-device application that runs in parallel with the presentation on the main device.

3.2.6 companion screen application (or second screen application): A particular case of a companion application in which its running code renders some content in the screen of its secondary device.

3.2.7 declarative object content (or declarative media object content): A type of content that takes the form of a code written in some declarative language.

NOTE – A HTML-based document, an MHEG application and an embedded NCL application are examples of declarative media objects.

3.2.8 distributed multi-device application: An application with modules that run on secondary devices and on the main device, in which there is some sort of communication between these modules to perform the distributed presentation.

3.2.9 DTV (digital television): A generic reference to any digital TV system: terrestrial digital TV, satellite digital TV, IPTV, etc.

3.2.10 element attribute (or attribute of an element): A property of an XML element.

3.2.11 event: An occurrence in time that may be instantaneous or have measurable duration.

3.2.12 execution engine: A subsystem in a receiver that evaluates and executes imperative applications consisting of computer language instructions and associated data and media content. An execution engine may be implemented with an operating system, computer language compilers, interpreters, and application programming interfaces (APIs), which an imperative application may use to present audiovisual content, interact with a user, or execute other tasks, which are not evident to the user. A common example of an execution engine is the JavaTV software environment, using the Java programming language and byte code interpreter, JavaTV APIs, and a Java virtual machine for program execution.

3.2.13 hybrid application: A hybrid declarative application or a hybrid imperative application.

3.2.14 hybrid declarative application: A declarative application that makes use of imperative object content.

NOTE – An NCL document with an embedded NCLua object is an example of a hybrid declarative application.

3.2.15 hybrid imperative application: An imperative application that makes use of declarative content.

NOTE – A Java Xlet that creates and causes the display of an NCL document instance is an example of a hybrid imperative application.

3.2.16 imperative application: An application that is started by and primarily makes use of, imperative information to express its behaviour.

NOTE – A Java program and a Lua program are examples of imperative applications.

3.2.17 imperative application environment: An environment that supports the processing of imperative applications.

3.2.18 imperative object content (or imperative media object content): A type of content that takes the form of an executable code written in some non-declarative language.

NOTE – A Lua script is an example of imperative object content.

3.2.19 independent application: The main application or a companion application that does not communicate with other devices.

3.2.20 main application: An independent application or a module of a distributed multi-device application that controls the presentation on the main device.

3.2.21 main audiovisual content: The content presented on the main device (usually the main audio and the main video of a transport stream's program). The main audiovisual content at least starts its presentation on the main device and can be either the single content in presentation or be presented with other content on the main device, under control of the main application.

3.2.22 main device: The device that runs the code (a module of the distributed multi-device application or an independent application) that controls the presentation on the main rendering apparatus (e.g., the main screen).

3.2.23 main screen: The screen in which the main audiovisual content starts its presentation. It is the screen controlled by the main device.

3.2.24 media object (or media node): A collection of named pieces of data that may represent media content, multimedia content, or a program written in a specific language. In addition to media content, a media object contains a set of properties concerning its media content, like those specifying the position and size of the media content presentation, etc.

3.2.25 media player: The component of an application environment which decodes or executes a specific content type.

3.2.26 native application: An intrinsic function implemented by a receiver platform.

NOTE – A closed captioning display is an example of a native application.

3.2.27 NCL application: A set of information that consists of an NCL document (the application specification) and a group of data, including objects (media objects) referred in the NCL document.

3.2.28 NCL document (or NCL content): An NCL application specification; an NCL code chunk.

3.2.29 NCL formatter (NCL player): The software component that is in charge of receiving the specification of an NCL document and controlling its presentation, trying to guarantee that author-specified relationships among media objects are respected.

NOTE – NCL document renderer, NCL user agent and NCL player are other names used with the same meaning as that of an NCL formatter.

3.2.30 NCL node (or NCL object): Refers to a <media>, <context>, <body> or <switch> element of NCL.

3.2.31 NCL user agent: Any program that interprets an NCL document written in the document language according to the terms of this specification.

NOTE – A user agent may display a document, trying to guarantee that author-specified relationships among media objects are respected. The relation can be: read it aloud; cause it to be printed; convert it to another format, etc.

3.2.32 profile: The specification for a class of capabilities providing different levels of functionality in a receiver.

3.2.33 property element: The NCL element that defines a property name and its associated value.

3.2.34 secondary device: Any device used to run a code (a module of the distributed multi-device application or an independent application) while the main device is controlling the main presentation.

3.2.35 scripting language: The language used to describe an imperative object content that is embedded in another host language. For example, Lua is a scripting language for NCL documents as ECMAScript is a scripting language for HTML documents.

3.2.36 transport stream (TS): The MPEG-2 transport stream syntax for the packetization and multiplexing of video, audio, and data signals for digital broadcast systems.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ABNT	Brazilian Technical Standards Association
API	Application Programming Interface
DSM-CC	Digital Storage Media Command and Control
DTV	Digital Television
FLUTE	File Delivery over Unidirectional Transport
GIF	Graphics Interchange Format
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ISDB-T _B	International Standard for Digital Broadcasting-Terrestrial TV with Brazilian Innovations
JPEG	Joint Photographic Experts Group
LIME	Lightweight Interactive Multimedia Environment
MIME	Multipurpose Internet Mail Extension

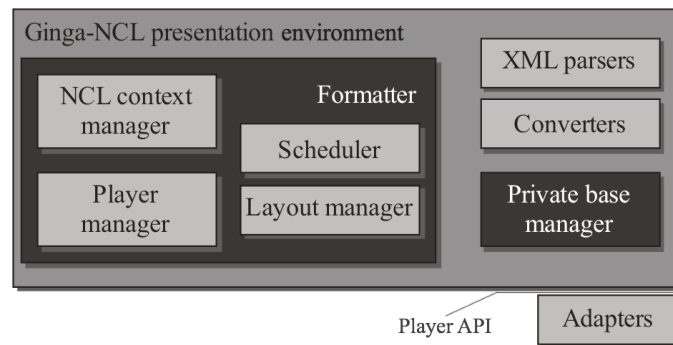


Figure 5-1 – Ginga-NCL presentation environment

6 Ginga-NCL harmonization with other IPTV declarative environments

An NCL application has a strict separation between its content and its structure. NCL itself does not define any media content. Instead, it defines the glue that holds media objects together in multimedia presentations.

An NCL document only defines how media objects are structured and related in time and space. As a glue language, it does not restrict or prescribe the content types of its media objects. Which media objects are supported depends on the media players that are coupled in the NCL formatter. Among these players are the video and audio decoder/players that actuate in the video plane of a structured screen (see clause 7.2.6), usually implemented in hardware in an IPTV receiver. In this way video and audio streams of a service presented in the video plane are treated like all other media objects displayed in any presentation plane (see clause 7.2.6) that may be related using NCL.

Another NCL media object that is required in a Ginga-NCL implementation is the hypertext markup language (HTML)-based media object [b-W3C HTML]. NCL does not substitute, but embeds HTML-based documents (or objects). As with other media objects, which HTML-based language will be supported in an NCL formatter is an implementation choice and therefore it will depend on which HTML browser acts as a media player integrated to the NCL formatter.

As a consequence, it is possible for example to have a lightweight interactive multimedia environment (LIME) [b-ITU-T H.762] engine embedded in an NCL document player. It is also possible to receive a HTML-based browser code through datacasting and to install it as a plug-in (usually as Lua objects).

It is also possible to have a harmonization browser implemented and to receive the complementary part, if needed, as a plug-in, in order to convert the HTML player into one of the several IPTV browser standards.

Note that in extreme cases, an NCL document may be reduced to having only one HTML media object. In this case, the NCL document player will act very like a HTML browser.

Whatever the case, the HTML-based browser implementation shall be a consequence of the following requirements:

- minimization of the redundancy with existing NCL facilities;
- robustness;
- alignment with World Wide Web Consortium (W3C) specifications;
- rejection of non-conformant content;
- precise content layout control mechanisms;
- support of different pixel aspect ratios;
- support to Ginga-NCL's player application programming interface (API).

Although an HTML-based browser is required to be supported, the use of HTML elements to define relationships (including HTML links) is not recommended when authoring NCL documents. Structure-based authoring should be emphasized for the well-known reasons largely reported in the literature.

When any media player, in particular a HTML-based browser, is integrated to the Ginga-NCL formatter, it shall support the generic API discussed in clause 8. Therefore, for some HTML-based browsers, an adapter module may be necessary to accomplish the integration.

Finally, for live editing, Ginga-NCL also supports event descriptors and editing commands defined by NCL.

Another NCL media object that must be supported by a Ginga-NCL implementation is the declarative NCL media object, i.e., a media object whose content is an NCL application. Therefore, NCL applications can be embedded in NCL parent applications, as well as HTML-based applications.

7 NCL: XML application declarative language for multimedia presentations

The modularization approach has been used in several XML-based language recommendations.

Modules are collections of semantically-related XML elements, attributes and attribute values that represent a unit of functionality. Modules are defined in coherent sets. This coherence is expressed in that the elements of these modules are associated with the same namespace [b-W3C XMLNAMES1].

A language profile is a combination of modules. Modules are atomic, i.e., they shall not be subdivided when included in a language profile. Furthermore, a module specification may include a set of integration requirements to which language profiles that include the module shall comply.

NCL has been specified in a modular way, allowing for the combination of its modules in language profiles [b-NCL DTV]. Each profile may group a subset of NCL modules, allowing for the creation of languages according to the users' needs. Moreover, NCL modules and profiles can be combined with other language modules, allowing for the incorporation of NCL features into those languages and vice-versa.

Commonly, there is a language profile that incorporates nearly all the modules associated with a single namespace. Other language profiles can be specified as subsets of the larger one. This is the case for the enhanced digital television (DTV) profile and the raw DTV profile of NCL, focal points of this Recommendation.

The main purpose of being in conformance with a language profile is to enhance interoperability. The mandatory modules are defined in such a way that any document interchanged in a conforming language profile will yield a reasonable presentation. The document formatter, while supporting the associated mandatory module set, shall ignore all other (unknown) elements and attributes.

NCL edition 3.1 is partitioned into 14 functional areas, which are further partitioned into modules:

- 1) Structure:
 - Structure module.
- 2) Components:
 - Media module.
 - Context module.
- 3) Interfaces:
 - MediaContentAnchor module.
 - CompositeNodeInterface module.
 - PropertyAnchor module.
 - SwitchInterface module.

- 4) Layout:
 - Layout module.
- 5) Presentation specification:
 - Descriptor module.
- 6) Timing:
 - Timing module.
- 7) Transition effects:
 - TransitionBase module.
 - Transition module.
- 8) Navigational key:
 - KeyNavigation module.
- 9) Presentation control:
 - TestRule module.
 - TestRuleUse module.
 - ContentControl module.
 - DescriptorControl module.
- 10) Linking:
 - Linking module.
- 11) Connectors:
 - ConnectorCommonPart module.
 - ConnectorAssessmentExpression module.
 - ConnectorCausalExpression module.
 - CausalConnector module.
 - CausalConnectorFunctionality module.
 - ConnectorBase module.
- 12) Animation:
 - Animation module.
- 13) Reuse:
 - Import module.
 - EntityReuse module.
 - ExtendedEntityReuse module.
- 14) Meta-Information:
 - Metainformation module.

7.1 Identifiers for NCL 3.1 module and language profiles

Each NCL profile should explicitly state the namespace universal resource identifier (URI) that is to be used to identify it.

Documents authored in language profiles that include the NCL structure module can be associated with the "application/x-ncl+xml" mime type. Documents using the "application/x-ncl+xml" mime type are required to be host language conformant.

The XML namespace identifiers for the complete set of NCL 3.1 modules, elements and attributes are contained within the following namespace: <http://www.ncl.org.br/NCL3.1/>.

Each NCL module has a unique identifier. The identifiers for NCL 3.1 modules shall comply with Table 7-1.

Modules may also be identified collectively. NCL 3.1 defines the following module collections:

- modules used by the NCL 3.1 language profile:
<http://www.ncl.org.br/NCL3.1/LanguageProfile>
- modules used by the NCL 3.1 enhanced DTV profile:
<http://www.ncl.org.br/NCL3.1/EDTVProfile>
- modules used by the NCL 3.1 raw DTV profile:
<http://www.ncl.org.br/NCL3.1/RawDTVProfile>

Table 7-1 – The NCL 3.1 module identifiers

Modules	Identifiers
Animation	http://www.ncl.org.br/NCL3.1/Animation
CompositeNodeInterface	http://www.ncl.org.br/NCL3.1/CompositeNodeInterface
CausalConnector	http://www.ncl.org.br/NCL3.1/CausalConnector
CausalConnectorFunctionality	http://www.ncl.org.br/NCL3.1/CausalConnectorFunctionality
ConnectorCausalExpression	http://www.ncl.org.br/NCL3.1/ConnectorCausalExpression
ConnectorAssessmentExpression	http://www.ncl.org.br/NCL3.1/ConnectorAssessmentExpression
ConnectorBase	http://www.ncl.org.br/NCL3.1/ConnectorBase
ConnectorCommonPart	http://www.ncl.org.br/NCL3.1/ConnectorCommonPart
ContentControl	http://www.ncl.org.br/NCL3.1/ContentControl
Context	http://www.ncl.org.br/NCL3.1/Context
Descriptor	http://www.ncl.org.br/NCL3.1/Descriptor
DescriptorControl	http://www.ncl.org.br/NCL3.1/DescriptorControl
EntityReuse	http://www.ncl.org.br/NCL3.1/EntityReuse
ExtendedEntityReuse	http://www.ncl.org.br/NCL3.1/ExtendedEntityReuse
Import	http://www.ncl.org.br/NCL3.1/Import
Layout	http://www.ncl.org.br/NCL3.1/Layout
Linking	http://www.ncl.org.br/NCL3.1/Linking
Media	http://www.ncl.org.br/NCL3.1/Media
MediaContentAnchor	http://www.ncl.org.br/NCL3.1/MediaContentAnchor
KeyNavigation	http://www.ncl.org.br/NCL3.1/KeyNavigation
PropertyAnchor	http://www.ncl.org.br/NCL3.1/PropertyAnchor
Structure	http://www.ncl.org.br/NCL3.1/Structure
SwitchInterface	http://www.ncl.org.br/NCL3.1/SwitchInterface
TestRule	http://www.ncl.org.br/NCL3.1/TestRule
TestRuleUse	http://www.ncl.org.br/NCL3.1/TestRuleUse
Timing	http://www.ncl.org.br/NCL3.1/Timing
TransitionBase	http://www.ncl.org.br/NCL3.1/TransitionBase
Transition	http://www.ncl.org.br/NCL3.1/Transition
Metainformation	http://www.ncl.org.br/NCL3.1/MetaInformation

Three synchronized multimedia integration language (SMIL) modules [b-W3C SMIL 2.1] were used as the basis for the NCL transition module and the NCL Metainformation module definitions.

7.1.1 NCL version information

The following processing instructions shall be written in an NCL document. They identify NCL documents that contain only the elements defined in this Recommendation and the NCL version to which the document conforms.

```
<?xml version="1.0" encoding="UTF-8"?>
<ncl id="any string" xmlns="http://www.ncl.org.br/NCL3.1/profileName">
```

The *id* attribute of an <ncl> element may receive any string that matches the NCName 'type' definition [b-W3C XMLNAMES1] as its value. That is, it may receive any string value that begins with a letter or an underscore and that only contains letters, digits, '-', '.', and '_'.

The version number of an NCL document specification consists of a major number and a minor number, separated by a dot. The numbers are represented in base 10 with leading zeros suppressed. The initial standard version number is 3.0.

New NCL versions shall be released in accordance with the following versioning policy. If receivers that conform to older versions can still receive and play a document based on the revised specification, the new version of NCL shall be released with the minor number updated. If receivers that conform to older versions cannot receive or play a document based on the revised specifications in its full functionalities, the major number shall be updated.

A specific version is specified in the URI path <http://www.ncl.org.br/NCLx.y/profileName>, where the version number "x.y" is written immediately after the "NCL".

The *profileName*, in the URI path, shall be *EDTVProfile* or *RawDTVProfile*.

7.2 NCL modules

7.2.1 General remarks

The main definitions made by the NCL 3.1 modules that are present in the NCL 3.1 enhanced DTV profile are given in clauses 7.2.2 to 7.2.15.

The complete definition of these NCL 3.1 modules, using XML schemas, is presented in Annex A. Any ambiguity found in this text can be clarified by consulting the XML schemas (see clause 7.2.2.1).

As stated in the scope of this Recommendation, NCL can be used in other declarative environments besides Ginga-NCL. Constraints coming only from Ginga-NCL are always pointed out in separate paragraphs of clause 7.2 mentioning the Ginga-NCL specification.

After discussing each module, a table is presented indicating the module elements and their attributes. The value of an attribute may not contain quotation marks ("), as is usual in XML attributes' values. For a given profile, attributes and contents (child elements) of an element may be defined in the module itself or in the language profile that groups the modules. Therefore, tables in this clause show attributes and contents that come from an NCL enhanced DTV profile, besides those defined in the NCL modules themselves. Tables in clause 7.3.2 show the attributes and contents that come from an NCL raw DTV profile, besides those defined in the NCL modules themselves. Element attributes that are required are underlined. In the tables, the following symbols are used: (?) optional (zero or one occurrence), () or, (*) zero or more occurrences, (+) one or more occurrences. The child element order is not specified in the tables.

Additionally, an NCL application is presented in Appendix II, for example purposes only.

7.2.2 Structure functionality

The structure functionality has just one module, called Structure, which defines the basic structure of an NCL document.

7.2.2.1 Structure module

The Structure module defines the <ncl> root element, the <head> element and the <body> element, following the terminology adopted by other W3C standards. The <body> element of an NCL document is treated as a nested context model (NCM) context node [b-NCM Core].

In NCM, the conceptual data model of NCL, a node may be a context, a switch, or a media object. All NCM nodes are represented by corresponding NCL elements. Context nodes (see clause 7.2.3) contain other NCM nodes and links.

Most NCL elements have the *id* attribute. This attribute may receive as its value any string that matches the NCName type definition, [b-W3C XMLNAMES1], i.e., it may receive any string value that begins with a letter or an underscore and that only contains letters, digits, '-', '.' and '_'. The *id* attribute uniquely identifies an element within a document. Its value is an XML identifier.

In particular, the <ncl> element shall define the *id* attribute and the <body> element may define the *id* attribute.

The *xmlns* attribute of <ncl> declares an XML namespace, i.e., it declares the primary collection of XML-defined constructs used in the document. The attribute value is the universal resource locator (URL) identifying where the namespace is officially defined. Two values are allowed for the *xmlns* attribute: "http://www.ncl.org.br/NCL3.1/EDTVProfile" for the enhanced DTV profile and "http://www.ncl.org.br/NCL3.1/RawDTVProfile" for the raw DTV profile. An NCL formatter shall know that the schemaLocation for these namespaces is, by default, respectively:

http://www.ncl.org.br/NCL3.1/profiles/NCL31EDTV.xsd,
http://www.ncl.org.br/NCL3.1/profiles/NCL31RawDTV.xsd

Child elements of <head> and <body> are defined in other NCL modules.

The elements of this module, their child elements and their attributes shall comply with Table 7-2.

Table 7-2 – Extended structure module used in the EDTV profile

Elements	Attributes	Content
ncl	<i>id</i> , <i>xmlns</i>	(head?, body?)
head		(importedDocumentBase?, ruleBase?, transitionBase?, regionBase*, descriptorBase?, connectorBase?, meta*, metadata*)
body	<i>id</i>	(port property media context switch link meta metadata)*

7.2.2.1.1 Exception handling

- Documents with *id* attributes whose values are not strings that match the NCName production [Namespaces in XML] shall be ignored by an implementation in conformance with this Recommendation.
- Documents with *xmlns* attributes different from the three previously mentioned values and that are not in conformance with future versions of this Recommendation shall be ignored by an implementation in conformance with this Recommendation.
- All attributes other than the *id* and *xmlns* attributes of the <ncl> element should be ignored by the NCL player.

7.2.3 Components functionality

The Components functionality is partitioned into two modules: the Media module and the Context module.

7.2.3.1 Media module

The Media module defines basic media object types. For defining media objects, this module defines the <media> element. In addition to the *id* attribute, each media object may define two main attributes: *src*, which defines the URI of the object content and *type*, which defines the object type.

In an implementation in conformance with Ginga-NCL specification, the uniform resource identifiers (URIs) listed in Table 7-3 are defined.

Table 7-3 – Allowed URIs

Scheme	Scheme-specific-part	Use	Agreement
file:	///file_path/#fragment_identifier	Local files	Required
http:	//server_identifier/file_path/#fragment_identifier	Remote files downloaded using the hypertext transfer protocol (HTTP) protocol. It can also refer to streams using HTTP-based streaming protocols like Moving Picture Experts Group (MPEG) DASH	Required
https:	//server_identifier/file_path/#fragment_identifier	Remote files downloaded using the HTTPS protocol	Required
rtsp:	//server_identifier/file_path/#fragment_identifier	Streams using the RTSP protocol	Optional
rtp:	//server_identifier/file_path/#fragment_identifier	Streams using the RTP protocol	Optional
ncl-mirror:	//media_element_identifier	Content flow identical to the one in presentation by another media element	Required
ts:	//sender_identifier/ program_number.component_tag	Elementary streams contained in a transport stream	Optional
dsm-cc:	//sender_identifier/ program_number.component_tag/ carouselId/moduleId/ObjectKey/ file_path/#fragment_identifier	Non-real-time broadcasting of audio, video and data files (filecasting) that are to be stored at the receiver for playback at a later time. Transmission uses the digital storage media command and control (DSM-CC) [ISO/IEC 13818-6] protocol	Optional
flute:	//sender_identifier/channel_identifier/file_path/ #fragment_identifier	Non-real-time broadcasting of audio, video and data files (filecasting) that are to be stored at the receiver for playback at a later time. Transmission uses the file delivery over unidirectional transport (FLUTE) [IETF RFC 6726] protocol	Optional

An absolute URI by itself contains all the information needed to locate its resource. Relative URIs are also allowed. Relative URIs are incomplete addresses that are applied to a base URI to complete the location. The portions omitted can be the URI scheme, the server/sender and potentially, part of the URI path.

The primary benefit of using relative URIs is that documents and directories containing them may be moved or copied to other locations without requiring changing of the URI attribute values within the documents. This is especially interesting when transporting documents from the server part (usually broadcasters) to the receivers. Relative URI paths are typically used as a quick means of locating media files stored in the same directory as the current NCL document, or in a directory close to it. In this case, they often consist of just the filename (optionally with a fragment identifier into the file). They may also have a relative directory path before the filename.

The "ts" scheme identifies the elementary stream by its *program_number* (identification of the service) and *component-tag* (identification of an elementary stream of the identified service). It should be emphasized that references to streaming video or audio resources shall not cause tuning. References that imply tuning to access a resource shall behave as if the resource were unavailable. A relative URI is also allowed when using a "ts" scheme. In this case, *sender_identifier* can be omitted and the source of the tuned "ts" shall be assumed.

The "dsm-cc" scheme identifies the sender, the specific elementary stream (as in the "ts" scheme), the carousel in which the content is encoded, a module in this carousel and the object whose URI is the base for the relative URI specified in the scheme. The relative URI is also allowed when using the "dsm-cc" scheme. In this case, *sender_identifier* can be omitted and the source of the tuned "ts" shall be assumed. The *program_number.component_tag/carouselId/moduleId/ObjectKey* can also be omitted if: (i) it refers to the NCL document pushed in the same carousel and if this part of the URI can be inferred, for example, from NCL editing commands (see clause 9); (ii) if it refers to the object (*file_path* does not exist) and if this part of the URI can be inferred, for example, from NCL editing commands (see clause 9).

The "flute" scheme identifies the sender and the channel (service) to which a session is established. A relative URI is also allowed when using a "flute" scheme. In this case, the *channel_identifier* can be omitted if there is just one channel. The *file_path* can also be omitted if it is the same as the NCL document pushed in the same channel and if this part of the URI can be inferred, for example, from NCL editing commands (see clause 9).

NOTE 1 – Media objects with the same *src* value and with the corresponding URI scheme different from "ncl-mirror" have the same content to be presented. Moreover, the content of each object can have its presentation started at different moments in time, depending on the time the media objects were started. In addition, their presentations are completely independent. On the other hand, if the URI scheme is equal to "ncl-mirror", the media object whose *src* attribute defines this scheme and the media object referred by the scheme shall have the same content presentation and at the same moment in time, if both media objects are being presented, independently from their starting time. Being different media objects, their properties may have different values, as, for example, those that define the presentation location.

Media objects with the same *src* values and whose URIs refer to the same elementary media stream being broadcasted (pushed data) shall have the same content presentation and in the same moment in time, if they are being presented. However, being different media objects, their properties may have different values, as for example those defining the presentation location.

For media objects with the *src* attribute whose value identifies the "ts" scheme, the *program_number.component_tag* values can be substituted by the following reserved words:

- video: The primary video ES of the tuned or identified services.
- audio: The primary audio ES of the tuned or identified services.
- text: The primary text ES of the tuned or identified services.
- video(i): The *i*th smaller video ES *component_tag* listed in the program map table (PMT) of the tuned or identified services.
- audio(i): The *i*th smaller audio ES *component_tag* listed in the PMT of the tuned or identified services.

- text(i): The *i*th smaller text ES *component_tag* listed in the PMT of the tuned or identified services.

The URI schemes shall always be specified, unless a particular DTV system defines a default scheme. For example, ISDB-T adopted in Latin America specifies the dsm-cc scheme as default, when schemes are not defined. The ITU-T Ginga reference implementation assumes as the default scheme the one used to launch the NCL application (via an NCL editing command or any other means).

The allowed values for the *type* attribute shall follow the multipurpose Internet mail extension (MIME) media types format (or, more simply, MIME types). A MIME type is a character string that defines the class of media (audio, video, image, text, application) and a media encoding type (such as jpeg, mpeg, etc.). MIME types may be registered or informal. Registered MIME types are controlled by the Internet Assigned Numbers Authority (IANA). Informal MIME types are not registered with IANA, but are defined by common agreement.

A <media> element whose *type* value is prefixed by "application/x-" may be used to specify a declarative hypermedia-object or an imperative media object in an NCL application. In this case, the object's content (located through the *src* attribute) shall be a declarative or a non-declarative code span to be executed, respectively.

In an implementation in conformance with Ginga-NCL specification, two special types are defined: "application/x-ginga-NCL", and "application/x-ginga-NCLua".

NOTE 2 – In an implementation in conformance with Ginga-NCL specification, "application/x-ginga-NCL" and "application/x-ginga-NCLua" special types may also be defined as "application/x-ncl-NCL" and "application/x-ncl-NCLua", respectively. However this should be avoided since this is deprecated and will likely be unsupported in future versions of NCL.

The "application/x-ginga-NCL" type shall be applied to <media> elements with NCL code content (therefore, an NCL application can embed another NCL application). The "application/x-ginga-NCLua" type shall be applied to <media> elements with Lua imperative code content (see clause 10).

NCL media objects embedded in NCL applications and HTML-based media objects embedded in NCL applications shall follow the guidelines established in clause 8.3 (see also [b-NCL Decl. Obj.]).

NCLua objects embedded in NCL applications shall follow the guidelines established in clause 8.4.

Two other special types shall be supported by any NCL presentation engine: "application/x-ncl-time" and "application/x-ncl-settings".

NOTE 3 – In an implementation in conformance with Ginga-NCL specification, "application/x-ncl-settings" and "application/x-ncl-time" special types may also be defined as "application/x-ginga-settings" and "application/x-ginga-time", respectively. However, this should be avoided since this is deprecated and will likely not be supported in future versions of NCL.

The application/x-ncl-time type shall be applied to a special <media> element (there may be only one in an NCL document), whose content is the absolute value of the universal time coordinated (UTC). Note that any continuous <media> element with no source can be used to define a clock relative to the <media> element start time. In this <media> element, <area> child elements can be defined whose *begin* and *end* attributes delimit a period of time from the start of the <media> element.

The content of a <media> element of application/x-ncl-time type is a string with the following syntax: Year:"Month":Day:"Hours":Minutes:Seconds."Fraction, where Year is an integer; Month is an integer in the [1,12] interval; Day is an integer in the [1,31] interval; Hours is an integer in the [0,23] interval; Minutes is an integer in the [0,59] interval; Seconds is an integer in the [0,59] interval; Fraction is a positive integer.

The "application/x-ncl-settings" type shall be applied to a special <media> element (there may be only one in an NCL document) whose properties are global variables defined by the document author or reserved environment variables that may be manipulated by the NCL player. Table 7-4 states the already defined variables, their semantics and their possible values. In the table, if the value is

specified in *italic*, it means any value of the type in *italic*. If the value is a list, its elements must be separated by ',' and after the ',' there can be zero or more space characters.

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
system – set of variables managed by the receiver system; – they may be read, but they may not have their values changed by an NCL application, a Lua procedure or any other imperative or declarative procedure; – receiver's native applications may change the variables' values; – they shall persist during the receiver life cycle.	system.language	Audio language.	"ISO 639-1 code ISO 639-2 code"
	system.caption	Caption language.	"ISO 639-1 code ISO 639-2 code"
	system.subtitle	Subtitle Language.	"ISO 639-1 code ISO 639-2 code"
	system.returnBitRate(i)	Bit rate of the ith network interface in Kbit/s.	"positive real"
	system.screenSize	Device screen size, in (lines, pixels/line), when a class is not defined.	"positive integer, positive integer"
	system.screenOrientation	Device screen orientation	"portrait" "landscape"
	system.screenVideoSize	Resolution set for the device's screen video plane, in (lines, pixels/line), when a class is not defined	"positive integer, positive integer"
	system.screenBackgroundSize	Resolution set for the device's screen background plane, in (lines, pixels/line), when a class is not defined	"positive integer, positive integer"
	system.screenGraphicSize	Resolution set for the device's screen graphics plane, in (lines, pixels/line), when a class is not defined.	"positive integer, positive integer"
	system.audioType	Type of the device audio, when a class is not defined.	"mono" "stereo" "5.1"
	system.CPU	CPU performance in MIPS, regarding its capacity to run applications.	"positive real"
	system.memory	Minimum memory space in Mbytes provided to applications.	"positive integer"
	system.operatingSystem	Type of the operating system.	"string"

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
	system.luaVersion	Version of the Lua engine supported by the receiver.	"string"
	system.luaSupportedEventClasses	List of event classes supported by NCLua, separated by ','	"string"
	system.nclVersion	NCL language version.	"string"
	system.nclProfiles	Language profiles supported by the receiver, separated by ','	"string"
	system.gingaNclVersion	Ginga-NCL environment version.	"string"
	system.*	Any variable with the "system." prefix not listed in this table shall be reserved for future use.	
user – set of variables managed by the receiver system; – they may be read, but they may not have their values changed by an NCL application, a Lua procedure or any other imperative or declarative procedure; – receiver's native applications may change the variables' values; – they shall persist during the receiver life cycle.	user.age	User age.	"positive integer"
	user.location	User location shall be the country code concatenated with the country post code. The country code specification shall follow the ISO 3166-1 alpha 3 format.	"string"
	user.genre	User genre.	"m" "f"
	user.language	User language.	"ISO 639-1 code ISO 639-2 code"
	user.*	Any variable with the "user." prefix not listed in this table shall be reserved for future use.	

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
si – set of variables managed by the middleware system; – they may be read but they may not have their values changed by an NCL application, a Lua procedure or any other imperative or declarative procedure; – they shall persist at least until the next channel tuning.	si.numberOfServices	Number of services available in the tuned channel for the local country. NOTE – The value of this variable should be obtained from the number of PMT tables specified in the program association table (PAT) table of the transport stream received from the tuned channel (see [ITU-T H.222.0]). The variable value should take into account only the PMT tables whose field country_code are equal to the value of the user.location variable of the Settings node (media object of "application/x-ncl-settings" type).	<i>"positive integer"</i>
	si.channelNumber	Number of the tuned channel.	<i>"positive integer"</i>
	si.*	Any variable with the "si." prefix not listed in this table shall follow the rules specified for the group.	
metadata – set of variables managed by the middleware system; – they may be read but they may not have their values changed by an NCL application, a Lua procedure or any other imperative or declarative procedure; – they shall persist at least until the next channel tuning.	metadata.*	Any variable with the "metadata" prefix shall follow the rules specified for the group. Variables in this group shall follow the high-level specification of metadata for IPTV services [ITU-T H.750].	

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
default – set of variables managed by the receiver system; – they may be read and have their values changed by an NCL application, a Lua procedure or any other imperative or declarative procedure; – receiver's native applications may change the variables' values;	default.focusBorderColor	Default colour applied to the border of an element in focus.	"white" "black" "silver" "gray" "red" "maroon" "fuchsia" "purple" "lime" "green" "yellow" "olive" "blue" "navy" "aqua" "teal"
	default.selBorderColor	Default colour applied to the border of an element in focus when activated.	"white" "black" "silver" "gray" "red" "maroon" "fuchsia" "purple" "lime" "green" "yellow" "olive" "blue" "navy" "aqua" "teal"
	default.focusBorderWidth	Default width (in pixels) applied to the border of an element in focus.	"integer"
– they shall persist during all receiver life cycle, however, they shall be set to their initial values when a new channel is tuned.	default.focusBorderTransparency	Default transparency applied to the border of an element in focus.	"Real value between 0 and 1" "Real value in the range [0,100] ending with the character '%" (e.g., 30%)" NOTE – "1" or "100%" means full transparency and "0" or "0%" means no transparency.
	default.*	Any variable with the "default." prefix not listed in this table shall be reserved for future use.	
service	service.currentFocus	The focusIndex value of the <media> element on focus.	"positive integer"

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
– set of variables managed by the NCL player; – they may be read and have their values changed by an NCL application of the same service; – they may be read but they may not have their values changed by a Lua procedure or any other imperative or declarative procedure of the same service; however, variable changes may be done using NCL commands; – they shall persist at least during the application life cycle.	service.currentKeyMaster	Identifier (id) of the <media> element that controls the navigational keys; if the <media> element is not being presented or is not paused, the navigational key control pertains to the NCL Formatter.	"string"
	service.screenSize(i)	Screen size of the class (i) of devices in (lines, pixels/line).	"positive integer, positive integer"
	service.screenGraphicSize(i)	Resolution set for the screen graphics plane of the class (i) of devices, in (lines, pixels/line).	"positive integer, positive integer"
	service.audioType(i)	Type of the audio of the class (i) of devices.	"mono" "stereo" "5.1"
	service.classDevNumber(i)	Number of devices registered in the class (i).	"positive integer"
	service.classType(i)	Type of the class (i).	"passive" "active" "mediaCapture" "onDemandMedia"
	service.classDevMin(i)	Minimum number of devices in class (i)	"positive integer"
	service.classDevMax(i)	Maximum number of devices in class (i)	"positive integer" "unbounded"
	service.classInfo(i)	List of class (i)'s media players.	"string"
	service.classNumber	Number of classes that have been defined.	"positive integer"
	service.*	Any variable with the "service." prefix not listed in this table shall follow the rules specified for the group.	
channel – set of variables managed by the NCL player;	channel.keyCapture	Request of alphanumeric keys for NCL applications.	"string"
	channel.virtualKeyboard	Request of a virtual keyboard for NCL applications.	"true" "false"

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
– they may be read and have their values changed by an NCL application of the same channel; – they may be read but they may not have their values changed by a Lua procedure or any other imperative or declarative procedure; however, variable changes may be done using NCL commands; – they shall persist at least until the next channel tuning. NOTE – A channel is defined as a set of related services.	channel.keyboardBounds	Virtual keyboard region (left, top, width, height).	<i>"positive integer, positive integer, positive integer, positive integer"</i>
	channel.*	Any variable with the "channel." prefix not listed in this table shall follow the rules specified for the group.	

Table 7-4 – Global variables

Group	Variable	Semantics	Possible values
shared – set of variables managed by the NCL player; – they may be read and have their values changed by an NCL application; – they may be read but they may not have their values changed by a Lua procedure or any other imperative or declarative procedure; however, variable changes may be done using NCL commands; – they shall persist at least during the receiver life cycle.	shared.*	Any variable with the "shared" prefix shall follow the rules specified for the group.	

Copyright International Telecommunication Union/ITU Telecommunication Sector
 Provided by IHS Markit under license with ITU-T
 No reproduction or networking permitted without license from IHS
 Full document details: ITU-T H.761 2014

Table 7-5 shows some possible values of the *type* attribute for the enhanced DTV profile and the associated file extensions for an implementation in conformance with Ginga-NCL specification. The required types shall be defined for each particular TV system. The *type* attribute is optional and should be used to guide the player's (presentation tool) choice by the formatter. When the *type* attribute is not specified, the formatter shall use the content extension specification in the *src* attribute to make the player's choice.

When there is more than one player for the type supported by the formatter, the *player* property of the <media> element may specify which one will be used for presentation. Otherwise the formatter shall use a default player for that type of media.

Table 7-5 – MIME media types for Ginga-NCL formatters

Media type	File extensions
text/html	htm, html
text/plain	txt
text/css	css
text/xml	xml
image/bmp	bmp
image/png	png
image/mng	mng
image/gif	gif
image/jpeg	jpg, jpeg
audio/basic	wav
audio/mp3	mp3
audio/mp2	mp2
audio/mpeg	mpeg, mpg
audio/mpeg4	mp4, mpg4
video/mpeg	mpeg, mpg
application/x-ginga-NCL	ncl
application/x-ginga-NCLua	lua
application/x-ncl-settings	<i>no src (source)</i>
application/x-ncl-time	<i>no src (source)</i>

The *instance*, *refer* and *descriptor* attributes of <media> elements are extensions defined in other modules and are discussed in the definition of these modules.

However, it should be stressed that a <media> element of application/x-ginga-NCL type may not have the *instance* and *refer* attributes.

The elements of the Media module, their child elements and their attributes shall comply with Table 7-6.

Table 7-6 – Extended Media module

Elements	Attributes	Content
media	<i>id</i> , <i>src</i> , <i>refer</i> , <i>instance</i> , <i>type</i> , <i>descriptor</i>	(area property)*

7.2.3.1.1 Default values

- The `system.nclVersion` and `system.gingaNclVersion` properties of the media object of "application/x-ginga-settings" (or "application/x-ncl-settings") type have "3.1" and "1.1" as default values, respectively.

7.2.3.1.2 Exception handling

- If a <media> element does not define the *src* and *type* attributes it shall be considered a media object with a continuous content. In the <media> element, <area> child elements can be defined whose *begin* and *end* attributes delimit a period of time from the start of the <media> element.
- Every action on a <media> element representing an unavailable resource shall be ignored by the NCL formatter. Every condition or assessment based on a <media> element representing an unavailable resource shall be considered as false.
- If the number of media objects of a certain type exceeds the maximum allowed number for that type in a particular device, the first of the exceeding media objects shall be ignored.
- The "ncl-mirror" scheme shall not refer to a <media> element of application/x-ginga-NCL, and application/x-ginga-NCLua types. If a <media> element whose *src* attribute specifies the "ncl-mirror" scheme and this scheme refers to a <media> element of application/x-ginga-NCL or application/x-ginga-NCLua types, the <media> element shall be ignored.
- References to streaming video or audio resources may not cause tuning. References that imply tuning to access a resource shall behave as if the resource were unavailable.
- If the file associated with a media object is updated in an object carousel and if the media object is not being presented when the updated version of the file arrives, the new version must replace the previous one. Therefore, when the media object is started after the update is completed, it will load the contents of the latest version. If the media object is started during the update, it must load the contents from the latest version of the file that has completely arrived. That version must be kept throughout the update process, only being replaced at the end. If the file associated with a media object is updated in an object carousel and if the media object is already being presented when the updated version of the file arrives, the media object presentation shall not be affected. However, the updated version must be used in future presentations.

7.2.3.2 Context module

The Context module is responsible for the definition of context nodes (context objects) through <context> elements. 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. As usual, the *id* attribute uniquely identifies each <context> element within a document.

The *refer* attribute is an extension defined in the Reuse module (see clause 7.2.14).

The elements of the Context module, their child elements and their attributes shall be in agreement with Table 7-7.

Table 7-7 – Extended Context module

Elements	Attributes	Content
context	<i>id</i> , <i>refer</i>	(port property media context link switch meta metadata)*

7.2.4 Interfaces functionality

The Interfaces functionality allows for the definition of node (media object or composite object) interfaces that will be used in relationships with other node interfaces. This functionality is partitioned into four modules:

- MediaContentAnchor, which allows for content anchor (or area) definitions as interfaces of media nodes (<media> elements);
- PropertyAnchor, which allows for the definition of node properties as interfaces of nodes;
- CompositeNodeInterface, which allows for port definitions as interfaces of composite nodes (<context> and <switch> elements); and
- SwitchInterface, which allows for the definition of special interfaces for <switch> elements.

7.2.4.1 MediaContentAnchor module

The MediaContentAnchor module defines the <area> element, which allows for the definition of content anchors representing:

- spatial portions, through the *coords* attribute (as in HTML);
- temporal portions relative to the begin time of the content presentation, through *begin* and *end* attributes;
- temporal and spatial portions through *coords*, *begin* and *end* attributes;
- textual chunks, through the *beginText*, *beginPosition*, *endText* and *endPosition* attributes that define the string and the string's occurrence in the text, respectively;
- temporal portions based on the number of audio samples or video frames, through *first* and *last* attributes, which shall indicate the initial and final sample/frame;
- temporal portions, through *first* and *last* attributes based on normal play time (NPT) values.
- When values of the *first* and *last* attributes of an <area> element are specified in NPT, they refer to the temporal base specified in the *contentId* attribute of the <media> element that contains the <area> element.

Moreover, the <area> element also allows for defining a content anchor based on:

- the *label* attribute, which specifies a string that should be used by a media player to identify a content region;
- the *clip* attribute, which specifies a triple value that shall be used by a media player to identify a clip in the content of a declarative hypermedia object.

Except for the <media> element of the application/x-ncl-time type, the *begin* and *end* attributes shall be specified according to one of the following syntaxes:

- Hours": "Minutes": "Seconds". "Fraction, Hours is an integer in the [0,23] interval; Minutes is an integer in the [0,59] interval; Seconds is an integer in the [0,59] interval; Fraction is a positive integer; or
- Seconds"s", where Seconds is a positive real number.

For the <media> element of the "application/x-ncl-time" type, the *begin* and *end* attributes shall be specified according to the following syntax:

Year": "Month": "Day": "Hours": "Minutes": "Seconds". "Fraction (according to the country time zone).

The NCL user agent is responsible for translating the value for the country time zone to the UTC time.

The first and last attributes shall be specified according to one of the following syntaxes:

- i) Samples"s", where Samples is a positive integer;
- ii) Frames"f", where Frames is a positive integer;
- iii) NPT"npt", where NPT is the normal play time value.

When values of the first and last attributes of an <area> element are specified in NPT, they refer to the temporal base specified in the contentId property of the <media> element that contains the <area> element.

For media objects of text type, the *beginText* and *beginPosition* attributes specifies the beginning of the text anchor. The text anchor ending may be specified using the *endTex* and *endPosition* attributes, which also define a string and the string's occurrence in the text, respectively.

EXAMPLE Assume the text content: "AAA AA AA AAA"; and the attributes *beginText*="AA".

For *beginPosition*=1, the anchor begins in the underlined bold string: **AAA** AA AA AAA

For *beginPosition*=2, the anchor begins in the underlined bold string: AA**AA** AA AA AAA

For *beginPosition*=3, the anchor begins in the underlined bold string: AAA **AA** AA AAA

For media objects of application/x-ginga-NCL type, the *clip* and *label* attribute values may be defined and shall follow the guidelines established for any declarative hypermedia objects in NCL, as follows.

A declarative hypermedia object (a <media> element of a declarative *type* with prefix "application/x-") is handled by the NCL parent application as a set of temporal chains. A temporal chain corresponds to a sequence of presentation events (occurrences in time, see definition of *events* in clause 7.2.12), initiated from the event that corresponds to the beginning of the declarative hypermedia object presentation. Sections in these chains may be associated with declarative hypermedia object's <area> child elements using the *clip* attribute. The *clip* value is a triple of the form "(chainId, beginOffset, endOffset)". The *chainId* parameter identifies one of the chains defined by the declarative hypermedia object. The *beginOffset* and *endOffset* parameters have the same syntax defined for the *begin* and *end* attributes and define the begin time and the end time of the content anchor, with regards the beginning time of the chain. When a declarative hypermedia object defines just one temporal chain, the *chainId* parameter may be omitted (and also the first comma in the list). The *beginOffset* and *endOffset* may also be omitted when they assume their default values: 0s or the chain end time, respectively e.g., "(chainId, 50s)".

For a declarative hypermedia object with NCL code (i.e., <media> element of "application/x-ginga-NCL" type), a temporal chain is identified by one of the NCL document entry points, defined by <port> elements (see clause 7.2.4.3), children of the document's <body> element.

A declarative hypermedia object's content anchor can also refer to any content anchor defined inside the declarative code itself. In this case, the *label* attribute of the <area> element that defines the content anchor has a value such that the declarative hypermedia object player is able to identify one of its internally defined content anchors. For a declarative hypermedia object with NCL code (i.e., <media> element of "application/x-ginga-NCL" type), one of its <area> elements may refer to a <port> element, child of its <body> element, through its *label* attribute (that must have the <port> element's *id* as its value). In its turn, the <port> element may be mapped to an <area> element defined in any object nested in the declarative NCL hypermedia object. Note that a declarative hypermedia object can externalize content anchors defined inside its content to be used in links defined by the parent NCL object in which the declarative hypermedia object is included.

In a media object of "application/x-ginga-NCLua" type, an imperative-code span may be associated with an <area> element using the *label* attribute. In this case, the *label* value shall identify the code

span. An <area> element may also be used just as an interface to be used as conditions of NCL links (set by Lua code) to trigger actions on other objects.

As usual, <area> elements shall have the *id* attribute, which uniquely identifies the element within a document.

In NCM, every node (media or context node) shall have an anchor with a region representing the whole content of the node. This anchor is called the *whole content anchor* and is declared by default in NCL documents. Except for media objects with imperative code content (e.g., <media> element of "application/x-ginga-NCLua" type), every time an NCL component is referred to without specifying one of its anchors, the *whole content anchor* is assumed.

The <area> element and its attributes shall comply with Table 7-8.

Table 7-8 – Extended MediaContentAnchor module

Elements	Attributes	Content
area	<i>id, coords, begin, end, beginText, beginPosition, endText, endPosition, first, last, label, clip</i>	Empty

7.2.4.1.1 Default values

- If the *begin* attribute is defined, but the *end* attribute is not specified, the end of the whole media content presentation shall be assumed as the anchor ending. On the other hand, if the *end* attribute is defined, but without an explicit *begin* definition, the start of the whole media content presentation shall be considered as the anchor beginning. Analogous behaviour is expected from the *first* and *last* attributes.
- In textual content anchors, if the end of the anchor region is not defined, the end of the text content shall be assumed. If the beginning of the content anchor region is not defined, the beginning of the text content shall be assumed.
- When a declarative hypermedia-object defines just one temporal chain, the *chainId* parameter may be omitted. The *beginOffset* and *endOffset* may also be omitted when they assume their default values: 0s and the chain end time respectively.
- Except for media objects with imperative code content (<media element of "application/x-ginga-NCLua" type, for example), every time an NCL component is referred without specifying one of its anchors, the *whole content anchor* is assumed.

7.2.4.2 PropertyAnchor module

The PropertyAnchor module defines an element named <property>, which may be used for defining a node (media object, context or switch) property or a group of node properties as one of its interfaces (anchors). The <property> element defines the *name* attribute, which indicates the name of the property or property group and the *value* attribute, an optional attribute that defines an initial value for the *name* property. The parent element shall not have <property> elements with the same *name* attribute values.

It is possible to have NCL document players (formatters) that define some node properties as node interfaces, implicitly. However, in general, it is a good practice to explicitly declare the interfaces.

A <media> element may have several embedded properties. Examples of these properties can be found among those that define the media object placement during a presentation, the presentation duration and others that define additional presentation characteristics: device (defining the device class in which the presentation will take place), top, left, bottom, right, width, height, zIndex, plane or plan (defining in which plane of a structured screen an object will be placed), explicitDur, background (specifying the background colour used to fill the area of a region displaying a media that

is not filled by the media itself), transparency (indicating the degree of transparency of an object presentation), rgbChromakey (defining the RGB colour to be set as transparent), visible (allowing the object presentation to be seen or hidden), fit (indicating how an object will be presented), scroll (which allows for the specification of how an author would like to configure the scroll in a region), style (which refers to a style sheet [b-W3C CSS2] with information for text presentation, for example), soundLevel, balanceLevel, trebleLevel, bassLevel, fontColor, fontFamily, fontStyle, fontSize, fontVariant, fontWeight, player, reusePlayer (which determines if a new player shall be instantiated or if a player already instantiated shall be used), playerLife (which specifies what will happen to the player instance at the end of the presentation), moveLeft, moveRight, moveUp, moveDown, focusIndex, focusBorderColor, focusBorderWidth, focusBorderTransparency, focusSrc, focusSelSrc, selBorderColor, transIn, transOut, freeze, etc. Application authors can also define their own other properties associated with <media> elements.

All properties may assume as their initial values those defined in homonym attributes of their node-associated descriptor and region (see clauses 7.2.6 and 7.2.5). However, it should be noted that <descriptor>, <descriptorParam> and <region> elements (see clauses 7.2.5 and 7.2.6) are only additional options for defining initial values for properties. Every property (and their initial values) defined by those elements may be defined using only <property> elements. Moreover, when the value of a property is specified in a <property> element, it has precedence over the value defined in homonym attributes of its node's associated descriptor or region.

Every <property> element has a Boolean attribute called *externable* that shall be set to "true" by default. When the property is intended to be used in a relationship, it shall be explicitly declared as a <property> (interface) element and with the *externable* attribute equal to "true".

NOTE 1 – If a property is defined in a <descriptor> or <region> element, its *externable* attribute shall be set to "false" by default.

Neither the *id* property (attribute) of a <media>, <context> or <switch> element nor the *src* property (attribute) of a <media> element shall have the *externable* associated attribute equal to "true". In other words, they cannot be changed by an action of a <link> element.

If the left, right, top, bottom, width or height properties defined in <property> elements have values in percentages (%), the percentages refer to the screen size of the device where the media object will be displayed.

For soundLevel, trebleLevel and bassLevel audio properties, their values must be interpreted relative to the recorded volume of the media. A setting of "0%" mutes the media. A value of '100%' will play the media at its recorded volume (i.e., 0 dB).

The plane property defines in which plane of a structured screen an object will be placed. The plan property has the same meaning but must be avoided, since it is deprecated, and it will likely be unsupported in future versions of NCL. The number of planes depends on the DTV system. Usually we can have at least three planes: background, video and graphics. The graphics plane superposes the video plane that superposes the background plane. The *zIndex* property gives the superposition order in just one plane, i.e., the plane superposition order has precedence over *zIndex* values.

If a video stream of a tuned service, which is not referred by any *src* attribute of <media> elements, is being presented on the video plane, the first started media object referring to this stream gets control of this content presentation, i.e., no new content presentation is started. Any other further <media> element that refers to this content by using the *src* attribute, when started, begins a new presentation.

If there is no media object being presented on the video plane referring (through its *src* attribute) to a video stream of a tuned service (no matter in which application of the private base that represents this TV channel), the video streams that were previously being presented in this plane when there were no application running shall be presented, with the same previous video parameters, although not being referred by any media object in presentation.

If an audio stream of a tuned service, which is not referred by any *src* attribute of <media> elements, is being presented, the first started media object referring to this stream gets control of this content presentation, i.e., no new content presentation is started. Any other further <media> element that refers to this content by using the *src* attribute, when started, begins a new presentation.

If there is no media object being presented referring (through its *src* attribute) to an audio stream of a tuned service (no matter in which application of the private base that represents this TV channel), the audio streams that were previously being presented when there were no applications running shall be presented, with the same previous audio parameters, although not being referred to by any media object in presentation.

The *visible* property may also be associated with a <context> or <body> element. In these cases, when the property's value is equal to "true", the *visible* property of each child element of the composition shall be taken into account. When the property's value is equal to "false", all child elements of the composition shall be hidden, i.e., although the values of their *visible* properties do not change, the child elements are not visible. In particular, when a document has its <body> element with its *visible* property set to "false" and its presentation event in the *paused* state (see definition of *events* in clause 7.2.12), the document is said to be in standby. When all applications in presentation are in standby, the video streams that were previously being presented in the video plane, when there was no application running shall be presented, with the same previous video parameters; similarly, the audio streams that were previously being presented when there were no application running shall be presented, with the same previous audio parameters. When any application returns from its standby state, it regains the whole control of these audio and video streams and the presentation resumes with the same media content presentation as before when the last application has entered in the standby state.

An object with a *visible* property equal to "false", i.e., a hidden object, may not transit selection event machines defined by its content anchors to the "occurring" state (see clause 7.2.12) while the *visible* property value persists as "false".

Some properties have their values defined by the middleware system, as for example, the *contentId* property (associated with a continuous-media object whose content is defined referring to an elementary stream), which has the "null" string as its initial value, and is set to the identifier value transported in the NPT reference descriptor (in a field of the same name: *contentId*), as soon as the associated continuous-media object is started. Another example is the standby property that shall be set to "true" while an already started continuous-media object content referring to an elementary stream is temporarily interrupted by another interleaved content, in the same elementary stream.

NOTE 2 – The standby property may be set to "true" when the identifier value transported in the NPT reference descriptor (in a field of the same name: *contentId*) signaled as non-paused is different from the *contentId* property value.

The *standby* property can be used to pause an application when the continuous media object content referring to an elementary stream is temporarily interrupted by another interleaved content, for example an advertisement (TV commercial). The same property can then be used to resume the application.

A group of node properties may also be explicitly declared as a single <property> (interface) element, allowing authors to specify the value of several properties within a single property. The following groups shall be recognized by an NCL formatter: *location*, grouping (left, top), in this order; *size*, grouping (width, height), in this order; and *bounds*, grouping (left, top, width, height), in this order. When a formatter treats a change in a property group, it shall only test the process consistency at its end.

The words device, top, left, bottom, right, width, height, *zIndex*, plane, plan, *explicitDur*, background, transparency, *rgbChromaKey*, *visible*, fit, scroll, style, *soundLevel*, *balanceLevel*, *trebleLevel*, *bassLevel*, *fontColor*, *fontFamily*, *fontStyle*, *fontSize*, *fontVariant*, *fontWeight*, *player*, *reusePlayer*,

playerLife, moveLeft, moveRight, moveUp, moveDown, focusIndex, focusBorderColor, focusBorderWidth, focusBorderTransparency, focusSrc, focusSelSrc, selBorderColor, transIn, transOut, freeze, location, size and bounds are reserved words for values of the *name* attribute of the <property> element. The possible values for these reserved property names shall comply with Table 7-9. Implementation in agreement with this Recommendation shall be able to handle all these properties.

Table 7-9 – Reserved parameter/attribute and possible values

Parameter/attribute name	Value	Default
top, left, bottom, right, width, height	Type: layout A real number in the range [0,100] ending with the character "%" (e.g., 30%), or an integer value specifying the attribute in pixels (a positive integer, in the case of width and height), ending optionally with "px" string (e.g., 30 px).	If any of these properties is not defined and cannot be inferred from the NCL rules, they shall assume value "0"
location	Type: position Two numbers separated by a comma, each one following the value rule specified for left and top parameters, respectively. There may be zero or more space characters after the comma that shall be ignored.	See first row
size	Type: size Two values separated by a comma. Each value shall follow the same rule specified for width and height parameters, respectively. There may be zero or more space characters after the comma that shall be ignored.	See first row
bounds	Type: sizePosition Four values separated by commas. Each value shall follow the same rule specified for left, top, width and height parameters, respectively. There may be zero or more space characters after the comma that shall be ignored.	See first row
plane	Type: plane "background", "video" or "graphic", following the plane definition of the DTV system.	"video", for media with <i>src</i> attribute referring to a transport stream (TS)'s packetized elementary stream (PES), "graphics", for all other cases.
baseDeviceRegion	Type: sizePosition. Four values separated by comma. Each value shall follow the same rule specified for left, top, width and height parameters, respectively. There can be zero or more space characters after the comma that shall be ignored.	See first row
device	Type: device. "class (i)" value, where <i>i</i> is an integer greater than zero.	class (0)
explicitDur	Type: time	For continuous media, the default value shall be

Table 7-9 – Reserved parameter/attribute and possible values

Parameter/attribute name	Value	Default
	i) Hours": "Minutes": "Seconds". "Fraction, where Hours is an integer in the [0,23] interval; Minutes is an integer in the [0,59] interval; Seconds is an integer in the [0,59] interval; and Fraction is a positive integer. ii) Seconds"s", where Seconds is a positive real number. iii) The null string.	set to the natural content presentation duration, otherwise it must be set to the "null" string
background	Type: color Reserved colour names: "white", "black", "silver", "gray", "red", "maroon", "fuchsia", "purple", "lime", "green", "yellow", "olive", "blue", "navy", "aqua", or "teal". The background value may also be the reserved value "transparent". This can be helpful to present transparent images, like transparent graphics interchange formats (GIFs), superposed on other images or videos.	transparent
visible	Type: Boolean "true" or "false".	true
transparency	Type: percent A real number in the range [0,1] or a real number in the range [0,100] and ending with the character "%" (e.g., 30%), specifying the degree of transparency of an object presentation ("1" or "100%" means full transparency and "0" or "0%" means opaque).	0
rgbChromakey	Type: RGB888 An RGB 888 value, or the "null" string.	the "null" string
fit	Type: fit "fill", "hidden", "meet", "meetBest", or "slice" value. "fill": scale the object's media content so that it touches all edges of the box defined by the object's width and height attributes. "hidden": if the intrinsic height (width) of the media content is smaller than the height (width) attribute, the object shall be rendered starting from the top (left) edge and have the remaining height (width) filled up with the background colour; if the intrinsic height (width) of the media content is greater than the height (width) attribute, the object shall be rendered starting from the top (left) edge until the height (width) defined in the attribute is reached, and have the part of the media content below (to the right of) the height (width) clipped. "meet": scale the visual media object while preserving its aspect ratio until its height or width is equal to the value specified by the height or width attributes. The media content left-top corner is positioned at the top-left coordinates of the box; the	fill

Table 7-9 – Reserved parameter/attribute and possible values

Parameter/attribute name	Value	Default
	empty space at the right or the bottom shall be filled up with the background colour. "meetBest": the semantic is identical to "meet" except that the image is not scaled greater than 100% in either dimension. "slice": scale the visual media content while preserving its aspect ratio until its height or width are equal to the value specified in the height and width attributes and the defined presentation box is completely filled. Some parts of the content may get clipped. Overflow width is clipped from the right of the media object. Overflow height is clipped from the bottom of the media object.	
scroll	Type: scroll "none", "horizontal", "vertical", "both", or "automatic" value.	none
style	Type: style The locator of a stylesheet file, or the "null" string.	the "null" string
soundLevel, trebleLevel, bassLevel	Type: percent A real number in the range [0,1] or a real number in the range [0,100] and ending with the character "%" (e.g., 30%).	1
balanceLevel	Type: balance A real number in the range [-1,1].	0
zIndex	Type: unsigned integer An integer number in the range [0,255], where regions with greater <i>zIndex</i> values are stacked on top of regions with smaller <i>zIndex</i> values.	0
fontColor	Type: color "white", "black", "silver", "gray", "red", "maroon", "fuchsia", "purple", "lime", "green", "yellow", "olive", "blue", "navy", "aqua", or "teal".	white
textAlign	Type: alignment. Sets the horizontal alignment of text: "left", "right", "center", or "justified" value.	left
fontFamily	Type: fontFamily A prioritized list of font family names and/or generic family names.	DTV system dependent
fontStyle	Type: fontStyle Sets the style of the font ("normal" or "italic" value).	normal
fontSize	Type: fontSize The size of a font.	DTV system dependent
fontVariant	Type: fontVariant Displays text in a "small-caps" font or a "normal" font.	normal

Table 7-9 – Reserved parameter/attribute and possible values

Parameter/attribute name	Value	Default
fontWeight	Type: fontWeight Sets the weight of a font ("normal" or "bold" value).	normal
player	Type: string. Establishes the player to be used	-
reusePlayer	Type: Boolean "false", "true" value.	false
playerLife	Type: playerLife "keep" or "close" value.	close
moveLeft, moveRight, moveUp, moveDown, focusIndex	Type: optionalInteger Positive integer or the "null" string.	the "null" string
focusBorderColor;	Type: color "white", "black", "silver", "gray", "red", "maroon", "fuchsia", "purple", "lime", "green", "yellow", "olive", "blue", "navy", "aqua", or "teal".	DTV system dependent: the value defined by the <i>default.focusBorderColor</i>
selBorderColor	Type: color "white", "black", "silver", "gray", "red", "maroon", "fuchsia", "purple", "lime", "green", "yellow", "olive", "blue", "navy", "aqua", or "teal".	DTV system dependent: the value defined by the <i>default.selBorderColor</i>
focusBorderWidth	Type: unsigned integer An integer value specifying the attribute in pixels.	DTV system dependent: the value defined by the <i>default.focusBorderWidth</i>
focusBorderTransparency	Type: percent A real number in the range [0,1] or a real number in the range [0,100] ending with the character "%" (e.g., 30%), specifying the degree of transparency of an object presentation ("1" or "100%" means full transparency and "0" or "0%" means opaque).	DTV system dependent: the value defined by the <i>default.focusTransparency</i>
focusSrc, focusSelSrc	Type: optionalURI An URI or the "null" string	the "null" string
freeze	Type: Boolean "true" or "false" value.	False
transIn, transOut	Type: optionalTransitionList i) A semicolon-separated list of <transition> element identifiers defined in the <transitionBase> element; or ii) A semicolon-separated list of transitions in which each transition is defined by a list of parameters (type, subtype, dur, startProgress, endProgress, direction, fadeColor, horzRepeat, vertRepeat, borderWidth, borderColor) separated by commas. In the lists, there can be zero or more space characters after the commas or the semicolons, which shall be ignored. In a transition list, if a	the "null" string

Table 7-9 – Reserved parameter/attribute and possible values

Parameter/attribute name	Value	Default
	parameter does not exist, it must be declared as the null string.	

NOTE – The parameters that define a transition specified in a transIn or transOut list shall follow the syntax and semantics of the corresponding attributes of the <transition> element defined in clause 7.2.8.2.

Properties that have reserved color string names as values ("white", "black", "silver", "gray", "red", "maroon", "fuchsia", "purple", "lime", "green", "yellow", "olive", "blue", "navy", "aqua" or "teal") follow the CSS1 color standard, as defined in Table 7-10.

Table 7-10 – Reserved names for colour definition

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

The <property> element and its attributes shall be in agreement with Table 7-11.

Table 7-11 – Extended PropertyAnchor module

Elements	Attributes	Content
property	<u>name</u> , value, externable	Empty

7.2.4.2.1 Other default values

- The *value* attribute of a <property> element is optional and defines an initial value for the property declared as *name*. When the value is not specified, the property assumes as its initial value the one defined in homonym attributes of its node's associated descriptor or region, or else a default value. When the *value* is specified, it has precedence over the value defined in homonym attributes of its node's associated descriptor or region.

- If the *left*, *right*, *top*, *bottom*, *width* or *height* properties are not defined and cannot be inferred from property values defined on <property>, <descriptor> and its child elements, or <region> elements, they shall assume "0".
- If the audio properties *soundLevel*, *trebleLevel* or *bassLevel* are not specified they shall assume "100%" as value.
- The values "class (1)" of the device property are reserved to passive classes, and the values "class (2)", "class (3)", are reserved to active classes.

7.2.4.2.2 Exception handling

- If two or more <property> elements with the same *name* attribute are defined as child elements of the same <media> element, only the last *value* defined shall be taken into account. The others shall be ignored.
- When the *left*, *right*, *top*, *bottom*, *width* or *height* properties exceed the dimension of the display, only the content portion inside the display dimension shall be displayed.
- If a <bind> element refers to a <property> element with the *externable* attribute equal to "false", the <bind> element shall be ignored by the NCL formatter.
- The <body> and <context> elements should ignore <property> child elements whose *name* attribute has as value the words *top*, *left*, *bottom*, *right*, *width*, *height*, *zIndex*, *plane*, *explicitDur*, *background*, *transparency*, *rgbChromaKey*, *visible*, *fit*, *scroll*, *style*, *soundLevel*, *balanceLevel*, *trebleLevel*, *bassLevel*, *fontColor*, *fontFamily*, *fontStyle*, *fontSize*, *fontVariant*, *fontWeight*, *player*, *reusePlayer*, *playerLife*, *moveLeft*, *moveRight*, *moveUp*, *moveDown*, *focusIndex*, *focusBorderColor*, *focusBorderWidth*, *focusBorderTransparency*, *focusSrc*, *focusSelSrc*, *selBorderColor*, *transIn*, *transOut*, *freeze*, *location*, *size* and *bounds*.
- When the user specifies *top*, *bottom* and *height* information for the same <media> element, spatial inconsistencies can occur. In this case, the *top* and *height* values shall have precedence over the *bottom* value. Analogously, when the user specifies inconsistent values for the *left*, *right* and *width* properties, the *left* and *width* values shall be used to compute a new *right* value.
- When the *left*, *right*, *top*, *bottom*, *width* or *height* properties exceed the dimension of the display, only the content portion inside the display dimension shall be displayed.
- The *reusePlayer* and *playerLife* attributes offer additional support for media-objects' player management, and can be used or be ignored by a particular middleware implementer. The *playerLife* attribute specifies what will happen to a player instance at the end of a media object presentation. Maintaining a player instance demands memory space however decreases the player loading time and the probability for synchronization mismatches, as a consequence. The *reusePlayer* attribute allows using the same player instance to more than one media object presentation, including using a player left in the memory space by the *playerLife* attribute.
- If the *name* attribute of a <property> element is "src" or "id", the <property> element shall be ignored. These properties can be defined only by attributes of the <media> element. For these properties the *externable* attribute shall always have the "false" value.

7.2.4.3 CompositeNodeInterface module

The CompositeNodeInterface module defines the <port> element, which specifies a composite node port with its respective mapping to an interface (*interface* attribute) of one and only one of its components (specified by the *component* attribute).

The <port> element and its attributes shall comply with Table 7-12.

Table 7-12 – Extended CompositeNodeInterface module

Elements	Attributes	Content
port	<i>id</i> , <i>component</i> , <i>interface</i>	Empty

7.2.4.4 SwitchInterface module

The SwitchInterface module allows for the creation of <switch> element interfaces (see clause 7.2.10), which may be mapped to a set of alternative interfaces of internal nodes, allowing a link to anchor on the chosen interface when the <switch> is processed (see [b-NCM Core]). This module introduces the <switchPort> element, which contains a set of *mapping* elements. A *mapping* element defines a path from the <switchPort> to an interface (*interface* attribute) of one of the switch components (specified by its *component* attribute).

Every element representing an object interface (<area>, <port>, <property>, or <switchPort>) shall have an identifier (*id* attribute or *name* attribute).

A reference to an internal switch component shall be made through a <switchPort> element or, by default, to the <switch> element without specifying any <switchPort>. In this case, it is considered as if the reference is made to a default <switchPort> that contains mapping elements to each child object of the switch and referring to its *whole content anchor*.

A <switchPort> element may define a mapping to a subset of the switch's components. When a link is bound to a <switchPort> element and all the rules bound to mapped components are evaluated as false, the <defaultComponent> element shall be chosen; if the <defaultComponent> element is not defined no component shall be selected for presentation.

The <switchPort> element, its child elements, and its attributes shall comply with Table 7-13.

Table 7-13 – Extended SwitchInterface module

Elements	Attributes	Content
switchPort	<i>id</i>	mapping+
mapping	<i>component</i> , <i>interface</i>	Empty

7.2.5 Layout functionality

The Layout functionality has a single module, called Layout, which specifies elements and attributes that may define how objects will be initially presented inside regions of output devices. Indeed, this module may define initial values for homonym NCL properties defined by <media>, <body>, and <context> elements (see clause 7.2.3).

7.2.5.1 Layout module

A <regionBase> element, which may be declared in the NCL document <head>, defines a set of <region> elements, each of which may contain another set of nested <region> elements, and so on, recursively.

The <regionBase> element may have the *id* attribute, and <region> elements shall have the *id* attribute. As usual, the *id* attribute uniquely identifies the element within a document and shall follow the NCName production [Namespaces in XML].

Each <regionBase> element is associated with a class of devices where presentation will take place. In order to identify the association, the <regionBase> element defines the *device* attribute, which may have the values: *class (i)* (or "systemScreen (*i*)" or "systemAudio(*i*)"), where *i* is an integer greater than zero. The "systemScreen (*i*)" and "systemAudio(*i*)" values must be avoided, since they are deprecated, and they will likely be unsupported in future versions of NCL. The chosen class defines