INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2009-2012

IPTV-GSI

TD 482 (GEN/IPTV-GSI)

English only

Original: English

Question(s): 13/16 Milpitas, CA, USA, 24 - 28 September 2012

TD

Source: Editor H.IPTV-Widget

Title: H.IPTV-Widget "IPTV Widget service" (New): updated draft (IPTV-GSI, Milpitas, 24-28 September 2012) – CLEAN VERSION

Summary

This TD contains the updated draft of new Recommendation H.IPTV-Widget "*IPTV Widget Service*" updated during the Q13/16 meeting at the IPTV-GSI event held in Milpitas, 24–28 Sep 2012.

This updated draft reflects the input from contributions:

– IPTV-GSI-C.630 "*H.IPTV-Widget: Proposal to add optional elements and attributes to the Widget configuration document*"; Source: Mitsubishi Electric

– IPTV-GSI-C.639 "*H.IPTV-Widgets: Proposed revisions to seeking alignment in the denomination of widgets*"; Source: Brazil

NOTE – Editorial adjustments were performed by TSB prior to publication of this TD. In particular, table captions and references were added.

Contact:	Marcelo MORENO UFJF Brazil	Tel: +55 32 3229 3311 Email: moreno@ice.ufjf.br
Contact:	Carlos Eduardo BATISTA PUC-Rio Brazil	Tel: +55 21 3527 1500 ext. 3504 Fax: +55 21 3527 1530 Email: cbatista@inf.puc-rio.br
Contact	Fernando Masami Matsubara Mitsubishi Electric Japan	Tel: +81-467-41-2035 Fax: +81-467-41-2287 Email: Matsubara.Masami@eb.mitsubishielectric.co.jp

CONTENTS

Page

List of Tables

Page

Page

List of Figures

Page

Draft new Recommendation H.IPTV-Widget

Packaged IPTV Application (Widget) Service

AAP Summary

[To be added before Consent]

Summary

[To be added]

1 Scope

This document specifies the Packaged IPTV Application (widget) service

TBD

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this document 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.

TBD

[ITU-T H.761] Recommendation ITU-T H.761, *Nested Context Language (NCL) and Ginga-NCL for IPTV services*.

[ITU-T H.762] Recommendation ITU-T H.762, *Lightweight interactive multimedia framework for IPTV services (LIME)*.

[ITU-T H.770] Recommendation ITU-T H.770, *IPTV service discovery up to consumption "Mechanisms for service discovery and selection for IPTV services*.

[ITU-T HSTP.IPTV-HRM.3] ITU-T Technical Paper, *Harmonized security mechanisms for the MAFR series*. (Draft)

3 Definitions

3.1 Terms defined elsewhere

TBD

3.2 Terms defined in this document

This document defines the following terms:

3.2.1 IPTV Widgets: See packaged IPTV apps.

3.2.2 packaged IPTV apps: lightweight applications that are used frequently, such as calendars and news aggregators, with an easily accessible graphical user interface, often staying on the display.

4 Abbreviations and Acronyms

This Recommendation uses the following abbreviations and acronyms.

TD	Terminal device
TMW	Terminal middleware
MAFR	Multimedia application frameworks
NCL	Nested context language
LIME	Lightweight interactive multimedia framework

TBD

5 Introduction

Widget is the definition used for an interactive element of a graphical user interface (GUI) that displays an information arrangement. The origin of the term "widget" comes from the short for "window gadget", and it was first applied to user interface elements on documents from Project Athena in 1988 [b_Athena]. Commonly the term widget is also used to specify an element different from basic GUI components because it provides a single interaction point for the direct manipulation of data in a particular context, but as visual components, widgets can be combined to form an application, or may be used separated, as individual applications. Interactive buttons, sliders, boxes, menus and other components may be called widgets [b_WidgetIntro], but recently *the term is more often used to name a lightweight application* (e.g. stock market monitor, weather forecast, calculator, news aggregator, etc) that is usually packaged into a single file to facilitate its provision, transport and installation.

Widget, as a “packaged web app” [b_W3CWidgets], is an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user’s system. Widgets are used on many environments and with different applicability – they may be found on computers' desktops, mobile devices, web applications and Digital TV platforms as well.

A widget engine is the software layer that supports running and displaying widgets on a graphical user interface, such as the graphical layer of an IPTV terminal device. Such widgets commonly provide relevant information graphically and/or provide easy access to frequently used functions on a system. A widget toolkit is intended to be used by developers to develop new applications, combining several widgets. Most widget toolkits offer an easy way of development, sometimes allowing non-expert users to develop simple applications.

5.1 IPTV Widgets overview

Packaged IPTV apps, or simply IPTV Widgets, are lightweight applications that are used frequently, such as calendars and news aggregators, with an easily accessible graphical user interface, often staying on the display.

IPTV Widgets may be classified by their functionality. The classification below is a non-exhaustive list of categories that has been collected from [b_WidgetIntro], revised and extended to the IPTV domain:

– Accessory Widgets: self-contained widgets that do not require support from a content provider or from other applications (e.g.: clocks, calculators, offline games)

– Application Widgets: widgets that just present a different interface for a regular application already present in the terminal device (e.g.: mini player, address book, picture frame);

– Information Widgets: widgets that displays processed data downloaded from a content provider (e.g.: news readers, information tickers, weather forecasters).

– Service Widgets: Information widgets that are related to IPTV services (channel-specific EPG, content recommenders, service provider announcers)

Since an IPTV widget may run on different kinds of terminal devices, like set-top boxes, TV sets, and mobile devices, portability is an important issue and should be addressed based on standardized technologies supported by Widget Engines in the terminal device. IPTV Widgets must be developed using the technologies defined in the H.760 series (Multimedia Application Framework), such as HTML, LIME, CSS, ECMAScript, NCL and Lua.

6 IPTV widget service

The widget service takes advantage of functionalities described in H.721 (IPTV terminal device Basic model), as well as protocols and services specified in H.770 (IPTV service discovery) and the family of MAFR specifications including H.761, H.762 (LIME) and H.IPTV-MAFR.6 ECMAScript.

Table 1 illustrates the main features of the widget service.

Table 1 - Main features of the widget service

	ITU-T Widget service scope	Notes
Widget service discovery	Yes	Refer to Clause 7.2 References: H.770
Interactive widgets	Yes	Based on supported MAFR user agent (NOTE 1)
Non interactive widgets	Yes	Based on supported MAFR user agent (NOTE 1)
Media type	A/V, Audio, Image, Text	Depends on client functions of terminal device (NOTE 2)
Access local data	Yes	Depends on client functions of terminal device (NOTE 2) Based on supported MAFR user agent (NOTE 1)
Access remote web data	Yes	Depends on client functions of terminal device (NOTE 2) Based on supported MAFR user agent (NOTE 1)
Widget packaging (for download)	Yes	Refer to Clauses 6.1, 6.2 Based on W3C
Graphical user interface	Yes	Depends on client functions of terminal device (NOTE 2)

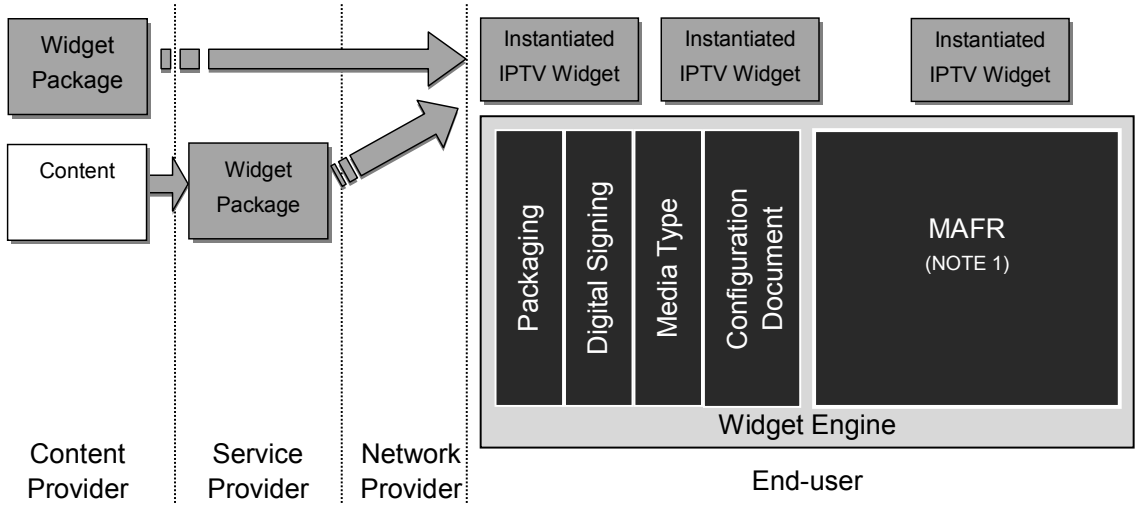
NOTE 1 - Currently supported user agents include ITU-T H.761, H.762, H.764.

NOTE 2 - Currently supported terminal devices include H.721

6.1 Widget service end-to-end architecture

Description of end to end configuration

(Contributor's note: Description will be provided)



NOTE 1 - Currently supported user agents include H.761, H.762, H.IPTV-MAFR.6

Figure 6.1 – End to end configuration

6.2 IPTV widget service discovery

Widget services are discovered using the framework established in H.770. Widget service metadata can be accessed under the "Other services" category defined in H.770. Metadata semantics specific to the Widget service is described in Table 2.

Table 2 - Metadata semantics specific to the Widget service

Element / Attribute	Description	Example
Widgets Services		
@DomainName	The unique identifier given to the widget provider	WidgetsAnyCompany.net
@Version	Version of the offering record	3
Widget		
@Id	Identifies an instance of the widget allowing the service provider to provide multiple versions of the widget	231
Name	Name of the specific widget	CalendarWidget
Description	Textual description of the widget for potential display. Several textual descriptions are possible, each of them	Schedule / calendar service on your IPTV

Element / Attribute	Description	Example
Widgets Services		
	in a specific language.	
Author	Textual description of the widget author (person, or company).	Any company
License	Usage conditions.	(Usage conditions)
Language	Supported language(s).	en (English)
minver	Minimum version of the widget engine required to run the widget.	(minimum version to run the widget)
maxver	Maximum version of the widget engine required to run the widget.	(maximum version to run the widget)
Rating	Popularity rating of the widget.	2 (popularity ranking)
Downloads	Cumulative number of downloads.	1222 (number of downloads)
Added	Date of addition.	2008.6.1 (date of addition)
Updated	Date of latest update.	2009.1.1 (latest update date)
Picked	Flag set by service provider to promote one or more widgets	1
Tag	Text related to the widget without any predefined taxonomy	(tags describing widget)
Locator	Location (e.g. URI or IP addresses) to access the widget	calendarwidget.AnyCompany.com
Widget		
@Id		
Name		
…		

Widget service metadata can be extended by using a separate namespace (XMLNS) with the definition of other XML elements and attributes. Elements outside the widget service namespace are simply ignored during processing.

6.3 Protocols for widget service

(Contributor's note: Text will be provided. Harmonized with W3C and H.721, H.770)

7 IPTV Widget Engine

A widget engine is the software layer that provides features so that widgets can run on a certain user platform (i.e. an IPTV terminal device). Also referred as "widget user agents" [b_W3CWidgetsLand], they make use of a number of technologies that compose an environment to be used by developers to create lightweight applications. To achieve interoperability, Widget Engines must rely on a set of standards and protocols that enable the rapid development of single-purposed applications that handle multimedia resources (such as images, audio, video etc.), also being able to communicate over a network (using HTTP, for instance).

An IPTV Widget Engine is the entity responsible for instantiating widget(s) in the client side (i.e. the IPTV terminal device). Figure 7-1 illustrates the basic concept of a widget engine.

An IPTV Widget Engine is based on the MAFR technologies defined by ITU-T H.760 series. The IPTV Widget Engine handles the widget's instantiation (unpacking and configuring), verifies the widget's authenticity and integrity, allocates the proper resources for the widget's execution (players, API etc.), and controls the widget's execution lifecycle. As presented in Figure 7-1, an IPTV Widget Package shall contain NCL (H.761) and/or LIME (H.762) application(s), along with the resource files (media, scripts etc.) within its scope.

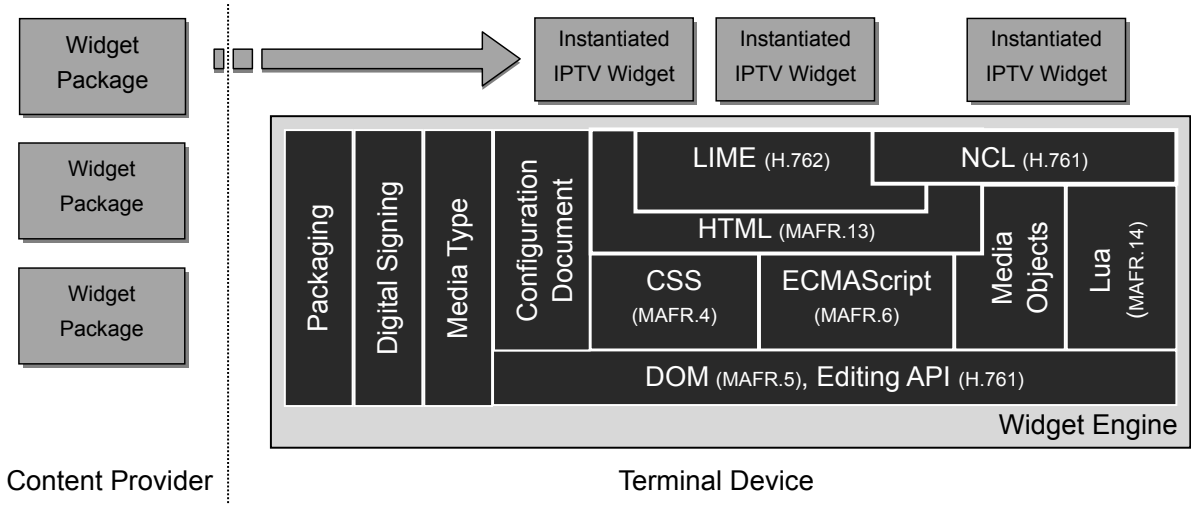


Figure 7.1 – Widget and widget engine

In an IPTV scenario, due to its flexibility and openness, it is foreseeable the presentation concurrency between third-party widgets and service platform provided content. This possibility may lead to discussions that are out of ITU-T Recommendations' scope. The definition of good practices and/or system restrictions to deal with that subject is up to business relationships.

8 IPTV Widgets

IPTV Widgets are desirable to be lightweight, easy to use, fast, accessible, portable, interoperable, customizable, secure and shareable. These goals can be used to define a set of guidelines for the development of widgets. The following guidelines are independent of the MAFR technology chosen to develop a widget:

– Packaging – a widget must be packaged using a standard format recognizable by IPTV widget engines. Widget developers must have in mind that this package will be distributed to different devices, locations, languages and depending of the business model, may be required to be signed.

– Metadata and Configuration – widget developers shall have the tools to inform the widget engine the configuration information of a widget, comprising: metadata elements about a widget, such as its title, some form of identification, and versioning information; metadata containing authorship information; a bootstrapping mechanism in order to enable the widget user agents to automatically instantiate a widget; environment configuration parameters.

– Security –The development of widgets incurs on dealing with security issues, so that the execution of the widget does not harm the terminal device nor compromises data privacy. Using functionalities provided by the supporting Widget Engine, the developer must

guarantee that the widget handles sensitive data confidentially and that it verifies data authenticity and integrity when needed.

8.1 IPTV Widget Packaging

Widget packaging refers to the technologies related to the encapsulation of all locally necessary resources and metadata required by the widget into a single file so that it can be distributed and deployed. The *de facto* standard for packaging widgets is the Zip file format [b_ZIP].

An IPTV Widget package includes the application file (e.g. an NCL or LIME application document), a configuration file and many media files (images, videos etc) compressed into a ZIP file, using the Zip Deflate compression algorithm. The package file extension is recommended to be ZIP's default ".zip" or, as an alternative, ".iwp", which stands for ITU-T IPTV Widget Package. In this case, the media type must be "application/x-iptv-widget".

Any IPTV Widget package must contain one configuration file, which is an XML document containing metadata information and configuration parameters to be interpreted by the widget engine. The configuration file must be placed in the root level of the package, and it must be named `config.xml`. The next subclause describes the information provided by an IPTV Widget configuration file.

8.2 IPTV Widget Metadata/Configuration

IPTV Widget Metadata/Configuration is related to how the information regarding a widget is stored inside the package, and how this data is made accessible. Information necessary for IPTV widget configuration is provided in the form of an XML document containing relevant metadata.

NOTE: We need to define the widget namespace.

Common metadata included in the widget configuration document is shown in Table 3. In order to support proprietary extensions of widget metadata, additional elements or attributes to those shown in the table can be included using a separate namespace. If the widget engine supports the corresponding element or attribute, then it is required to process accordingly; otherwise the widget engine is required to ignore.

Table 3 – Common metadata included in the widget configuration document

Element / @Attribute	Description	Value example (explanation)	Option-ality
widget			M
@id	URI that uniquely identifies the widget. Recommended to be an accessible URI for reference purposes, update checking, notifications, etc.	http://example.org/exampleWidget	O
@width	Preferred width for the widget exhibition area.	200	O
@height	Preferred height for the widget exhibition area.	320	O
@viewmodes	List of attributes denoting the preferred way of displaying the widget to the end user. The list contains one or more values,	windowed floating	O

Element / @Attribute	Description	Value example (explanation)	Option-ality
	separated by space characters, ordered by preference where the first value denotes the preferred way. Possible values relevant to IPTV terminal devices: windowed, floating, fullscreen, minimized.		
@version	A string that identifies the version of the widget.	2.1	O
name	A human readable string used, for instance, to identify the widget in a menu. Several names are possible, each of them in a specific language. Use attribute xml:lang to specify the name language.	Example Widget	O
description	A human readable string for a textual description of the widget. Several textual descriptions are possible, each of them in a specific language. Use attribute xml:lang to specify the description language.	Example service on your IPTV	O
author	Textual description of the widget author (person, or company).	Any Company	O
@href	Author's Website URI	http://www.anycompany.com	O
@email	Author's Email address	widgets@anycompany.com	O
license	String containing the copyright information for the widget	Copyright 2010 by Any Company This program is free software; you can redistribute…	O
language	Supported language(s).	en (English)	O
icon@Src	Icon image file for the widget	/media/icon.png	O
content			M
@src	Path to the main application source file	/main.ncl (relative path from the root of the package file system)	M
@type	Type of the main application source file	application/x-ginga-ncl	O
tag	Text related to the widget without any predefined taxonomy	(tags describing widget)	O
feature	URI to be used by the user agent to request a binding to a runtime component.	Runtime components needed by the widget (e.g., a video decoder).	O

Element / @Attribute	Description	Value example (explanation)	Option-ality
@name	URI attribute	http://www.anycompany.com	O
@required	Boolean value to specify if the component is required to function properly.	"true" : Widget needs to support the feature to function "false" : Widget can function without supporting the feature The default value when the attribute is omitted is "false".	O
param	Defines a parameter for the corresponding feature using name-value pair.	param is a child of the feature	O
@name	String to denote the name of the parameter	name= "Update frequency" Useful widget information (e.g., stock or weather widget)	O
@value	Value of the parameter	value= "Once per hour"	O
preference	Persistent storage for preferences using a name-value pair		O
@name	String to denote the name of the preference	name= "Home menu"	O
@value	Value of the preference	value= "My preferred skin"	O
@readonly	Boolean value to allow, disallow overwriting	"true" : can not be overwritten at runtime "false" : can be overwritten The default value when the attribute is omitted is "false".	O

The extension of the configuration document format can be done by using a separate namespace (XMLNS) with the definition of other XML elements and attributes. Elements outside the widget namespace are simply ignored during processing.

The following is an example illustrating how to include proprietary metadata using an external namespace. In this example, in addition to the standard namespace an external namespace ("http://somecompany.com/") enables the use of proprietary metadata. Proprietary metadata (i.e. not included in Table 1 of clause 8.2) can be used as shown in the fragment below. The attribute "src" is defined in the standardized namespace, while the proprietary attribute "role" is defined in the proprietary namespace.

```
<widget xmlns="http://itu.int/namespace/widgets"
   xmlns:ex="http://somecompany.com/">

      <icon src="big.png" ex:role="big"/>
      <ex:datasource>{'a':'b','c':'d'}</ex:datasource>

      <content src="widget.bml"/>
</widget>
```

Contributor's note: the above example includes the namespace
http://itu.int/namespace/widgets
which has not been agreed by ITU

8.3 IPTV Widget Security

IPTV Widget Security shall follow [ITU-T HSTP.IPTV-HRM.3], which defines the encryption methods for both the protection of MAFR content (the IPTV Widget itself) and the protection of data MAFR content handles (the data used by an IPTV Widget).

TBD

Appendix I
Proprietary Widget Engines and Development Platforms

I.1 Yahoo! Connected TV

The Yahoo! Connected TV platform (http://connectedtv.yahoo.com/developer) aims to harmonize Internet applications with the television, by offering different interface mechanisms since that it is a different environment. The platform offers a "conversion" of popular Web services such as EBay, Yahoo! Finance, Flickr, Twitter, and also news aggregation from sources like CBS and USA Today. Video on demand is also available, provided by a Netflix widget.

The platform offers a widget dock, which is accessible by the remote control, and from where the widgets are selected and loaded (Figure I.1). Users of the system may have a profile and their own widget collection, and each has also the possibility of personalizing the appearance and some other parameters of the widget.

Yahoo! Widget Engine, the core of the Yahoo! Connected TV platform is to be embedded in a variety of consumer electronics products, including television sets manufactured by Samsung, Sony, LG Electronics and VIZIO. It evolved from the Konfabulator (http://widgets.yahoo.com), which was a popular PC widget platform, and it was re-engineered for consumer electronic devices, such as Digital TV receivers.

The Yahoo! Widget Development Kit (WDK), provides all necessary tools for the development of Connected TV widgets, using an API called Widget Channel API. The API was developed in collaboration between Yahoo! and Intel, providing access to handling JavaScript and XML (Konfabulator) and HTML rendering.

Figure I.1 – Flickr widget on Yahoo! Connected TV

I.2 Verizon FiOS TV

The FiOS TV (https://www22.verizon.com/fiosdeveloper/General/Resource.aspx) platform is based upon the FiOS TV API, which offers the basic functionalities of a receiver (such as tuning, EPG data manipulation, basic graphic rendering, etc). The FiOS TV platform offers a framework for the development of TV widgets, aiming to offer the available functionalities of a Digital TV receiver through a high level framework, called FiOS TV Widget Application Framework, which is based on the Lua language, thus containing a Lua Virtual Machine. The framework incorporates Lua APIs organized upon four sets of functionalities: Graphics, Events, Timers and Webservices. Figure I.2 shows the software stack for the FiOS TV platform.

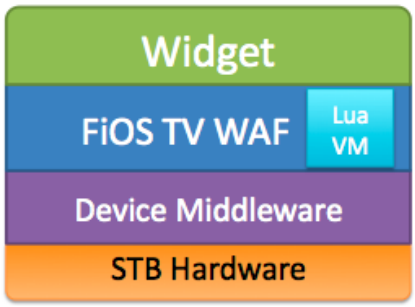


Figure I.2 – FiOS TV Software Stack

TV receivers using the FiOS TV platform are able to access a "Widget Bazaar" (Figure I.3), from where is possible to access the available widgets on a system, and obtain new ones (buying or for free). The Widget Bazaar enables content creators to develop and add widgets to the Widget Bazaar, using sophisticated interaction models, more complex than those offered by platforms based on XML and RSS technologies.

Figure I.3 – Verizon Widget Bazaar

Verizon is developing a SDK for the development of widgets, and plans to release it soon, with an emulation environment (FiOS TV Lua Emulator, Figure I.4) for development on personal computers. It simulates the FiOS TV receiver and provides a simulated TV screen and remote control, including tools for debugging the application.

Figure I.4 – FiOS TV Lua Emulator

Appendix II
ITU-T IPTV Widgets versus W3C Widgets

[Any text to go in the Appendix?]

Table II.1 – Title missing

	W3C	ITU-T IPTV scope
What is a widget?	Application	Lightweight Application
Widget service	Not defined	Yes, see Clause 6
Interactivity	Yes	Yes, but also widgets without interactivity
Single purpose	Yes	Not necessarily, so leave it out of the definition
Display data	Yes	Yes, but could be audible data, therefore use "render" data
Access local data	Yes	Yes via example
Access remote web data	Yes	Yes via example
Packaged for download	Yes	Yes
Easily accessible	No mention	Yes
Graphical user interface	No mention	Yes

Appendix III
IPTV Widget Examples

III.1 H.761 Widget example

A simple widget example is presented in this clause. The configuration file defines a start NCL application that exhibits news – the content is reloaded when the user presses the "BLUE" key and the application exits when the "RED" key is pressed. The configuration file defines a specific value for the "*service.language*" variable, which shall be used by the execution environment - all configuration parameters set by the configuration file must be also referenced by the application (the NCL application must include the same media objects for the associated parameters). In the example below, the widget (e.g. *news-widget.iwp*) contains the following directory tree:

```
–   main.ncl
–   config.xml
–   css/
–   css/style.css
–   media/
–   scripts/
–   scripts/counter.lua
```

config.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns       = "http://www.w3.org/ns/widgets"
        id          = "http://example.org/exampleWidget"
        version     = "3.0"
        height      = "300"
        width       = "200">
```

```
  <name>
    Example Widget
  </name>
  <description>
    A sample widget.
  </description>
  <author href  = " http://www.itu.int/ITU-T/gsi/iptv/"
          email = "moreno@ice.ufjf.br">Marcelo Moreno</author>
  <icon src="icons/icon.png"/>
  <content src="main.ncl" type="application/x-ginga-ncl"/>
  <license>
        Copyright.2010. International Telecommunication Union
  </license>
</widget>
```

main.ncl file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="test_widget" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
   <head>
      <!-- widget size and position relative to those defined by config.xml -->
      <regionBase>
         <region id="main_region" left="10" top="10" width="300" height="400"/>
      </regionBase>
      <!-- Stylesheet -->
      <descriptorBase>
         <descriptor id="main_content" region="main_region"
                     style="css/styles.css"/>
      </descriptorBase>
      <connectorBase>
         <causalConnector id="onKeySelectionStopStartSet">
            <connectorParam name="keyCode"/>
            <connectorParam name="var"/>
            <simpleCondition role="onSelection" key="$keyCode" />
            <compoundAction operator="seq">
               <simpleAction role="stop" max="unbounded" qualifier="par"/>
               <simpleAction role="start" max="unbounded" qualifier="par"/>
               <simpleAction role="set" value="$var" max="unbounded"
                             qualifier="par"/>
            </compoundAction>
         </causalConnector>
         <causalConnector id="onKeySelectionStopN">
            <connectorParam name="keyCode"/>
            <connectorParam name="var"/>
            <simpleCondition role="onSelection" key="$keyCode" />
            <simpleAction role="stop" max="unbounded"/>
         </causalConnector>
      </connectorBase>
      <!-- Rules used for internationalization of the content -->
      <ruleBase>
         <rule id="r_en" var="service.language" comparator="eq" value="eng"/>
         <rule id="r_pt" var="service.language" comparator="eq" value="por"/>
      </ruleBase>
   </head>
   <body>
      <port id="main_port" component="html_content"/>
      <!-- 'settings' node, repeats the parameters from config.xml -->
      <media type="application/x-ncl-settings" id="settings">
         <property name="service.language"/>
      </media>
      <!-- Lua script counting number of reloads -->
```

```
      <media id="lua_counter"  src="scripts/counter.lua">
         <property name="inc"/>
      </media>
      <!-- Content based on the presentation environment language -->
      <switch id="html_content">
         <defaultComponent component="html_en" />
         <bindRule rule="r_pt" constituent="html_pt" />
         <media id="html_pt" src="http://www.ginga.org.br/noticias.php"
                type="text/html" descriptor="main_content"/>
         <media id="html_en" src="http://www.ginga.org.br/news.php"
                type="text/html" descriptor="main_content"/>
      </switch>
      <!-- Ends the widget when users presses the red button -->
      <link xconnector="onKeySelectionStopN">
         <bind component="html_content" role="onSelection">
            <bindParam name="keyCode" value="RED"/>
         </bind>
         <bind role="stop" component="html_content"/>
      </link>
      <!-- The blue key reloads the news pages and increases the counter -->
      <link xconnector="onKeySelectionStopStartSet">
         <bind role="onSelection" component="html_content">
            <bindParam name="keyCode" value="BLUE"/>
         </bind>
         <bind role="stop" component="html_content"/>
         <bind role="start" component="html_content"/>
         <bind role="set" component="lua_counter" interface="inc">
            <bindParam name="var" value="1"/>
         </bind>
      </link>
   </body>
</ncl>
```

counter.lua file:

```
counter = 0
local counterEvt = {
    class = 'ncl',
    type  = 'attribution',
    property = 'counter',
}

function handler (evt)
    if evt.class ~= 'ncl' then return end
    if evt.type ~= 'attribution' then return end
    if evt.property ~= 'inc' then return end

    counter = counter + evt.value

    event.post {
        class    = 'ncl',
        type     = 'attribution',
        property = 'inc',
        action   = 'stop'
    }

    counterEvt.value = counter
    counterEvt.action = 'start'; event.post(counterEvt)
    counterEvt.action = 'stop';  event.post(counterEvt)
end

event.register(handler)
```

III.2 H.762 Widget example

An example of a simple H.762 widget is presented in this clause. The configuration file `config.xml` follows the conventions defined in clause 7.2 including necessary elements and attributes. In this example the files included in the Widget package are listed below. Note that this particular widget example does not require any external script file.

- `config.xml`
- `startup.bml`
- `object-element-test1.lime`
- `media/`
- `media/lime-large.jpg`
- `media/lime-small.jpg`
- `media/ITU-icon.jpg`
- `scripts/`

The content of the files in this example with the exception of the media files are presented below.

config.xml file

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<widget id="123" width="200" height="200" version="1">
  <Name>Widget sample in LIME</Name>
  <Description>Simple LIME Widget sample,
               with two JPEG files.
  </Description>
  <Author href="http://www.itu.int/ITU-T/gsi/iptv/"
          email="john.doe@ties.itu.int">
      John Doe
  </Author>
  <License>Copyright.2010. International Telecommunication
           Union
  </License>
  <Language>English, Japanese</Language>
  <Icon src="icon/ITU-logo.jpg" />
  <Content src="startup.bml" type="application/x-lime" />
  <Tag> images, LIME <Tag/>
</Widget>
```

startup.bml file

```
<?xml version="1.0" encoding="EUC-JP" ?>
<!DOCTYPE bml PUBLIC "-//ARIB STD-B24:1999//DTD BML Document for
IPTV//JA" "http://www.arib.or.jp/B24/DTD/bml_x_x_iptv.dtd">
<?bml bml-version="100.0" ?>
<bml>
<head>
<title>startup -- bml version</title>
<script><![CDATA[

function onload() {
```

```
browser.launchDocument("lime/object-element-test1.lime", "cut");
}

]]></script>

</head>
<body id="body" onload="onload();">

</body>
</bml>
```

object-element-test1.lime file

```
<?bml bml-version="100.0" ?>

<bml>

<head>
<title>Object element test 1:JPEG</title>
</head>

<body style="background-color-index:7">

<div style="left:0px;top:0px;width:960px;height:500px">
<object data="../images/lime-large.JPG" type="image/jpeg"
style="width:369px;height:252px;left:300px;top:150px;"/>
<object data="../images/ITU-Logo.JPG" type="image/jpeg"
style="width:189px;height:77px;left:20px;top:10px;"/>
</div>
</body>
</bml>
```

III.3 MAFR.13/MAFR.6 Widget example

TBD

III.4 MAFR.14 Widget example

TBD

Bibliography

[b_Athena] George A. Champine, Daniel E. Geer, and William N. Ruh, ‘‘*Project Athena as a Distributed Computer System*,’’ IEEE Computer 23(9), pp. 40-50 (September 1990).

[b_H.IPTV-MAFR.5] Draft new Recommendation "*Document Object Model (DOM) for IPTV Services*".

[b_H.IPTV-MAFR.14] Draft Recommendation ITU-T H.IPTV-MAFR.14 "*Lua script language for IPTV*"

[b_W3C XHTML] W3C Recommendation. XHTML 1.0 (2002), *Extensible HyperText Markup Language*

[b_W3C RDF] W3C Recommendation. RDF (1999), *Resource Description Framework (RDF) Model and Syntax Specification*.

[b_W3CWidgets] W3C Working Draft (2009) - *Widgets 1.0 Requirements*. http://www.w3.org/TR/2009/WD-widgets-reqs-20090430.

[b_W3CWidgetsLand] W3C Working Draft (2008) *Widgets 1.0: The Widget Landscape* (Q1 2008). http://www.w3.org/TR/2008/WD-widgets-land-20080414/

[b_WidgetIntro] C Kaar, "*An introduction to Widgets with particular emphasis on Mobile Widgets Computing*", October 2007.

[b_ZIP] .ZIP File Format Specification. PKWare Inc., September 2007. http://www.pkware.com/documents/casestudies/APPNOTE.TXT

_____