International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.730

(06/2012)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

IPTV multimedia services and applications for IPTV – IPTV middleware

Web-based terminal middleware for IPTV services

Recommendation ITU-T H.730

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
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

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

Recommendation ITU-T H.730

Web-based terminal middleware for IPTV services

Summary

Web-based terminal middleware (WBTM) defines the functional interfaces for high-level resource management over an IPTV terminal device and describes the structure of a web-based presentation engine, which basically supports IPTV multimedia application frameworks in the ITU-T H.76x series of Recommendations. Web-based IPTV terminal middleware is based on ITU-T IPTV functional architecture and ITU-T terminal devices in the H.72x series of Recommendations. ITU-T web-based IPTV terminal middleware is necessary to support basic and advanced interactive IPTV services for IPTV terminal devices.

This Recommendation also describes the general WBTM requirements for IPTV services and any additional functionality for basic and advanced IPTV services. Annex A summarizes the general requirements.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T H.730	2012-06-29	16

Keywords

IPTV services, IPTV terminal device, presentation engine, user agent, web-based terminal middleware.

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/.

Table of Contents

Page

Recommendation ITU-T H.730

Web-based terminal middleware for IPTV services

1 Scope

The purpose of web-based terminal middleware is to define a web-based presentation engine and the functional interfaces for high-level resource management over an IPTV terminal device [ITU-T Y.1901].

This Recommendation defines web-based terminal middleware, its general architecture and interfaces as required by IPTV services.

This Recommendation is based on the ITU-T H.72x series of Recommendations for the specifications of IPTV terminal devices and the ITU-T H.76x series of Recommendations for multimedia interactivity and web-related technologies.

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.720] Recommendation ITU-T H.720 (2008), *Overview of IPTV terminal devices and end systems*.

[ITU-T H.721] Recommendation ITU-T H.721 (2009), *IPTV terminal devices: Basic model*.

[ITU-T H.760] Recommendation ITU-T H.760 (2009), *Overview of multimedia application frameworks for IPTV services*.

[ITU-T H.761] Recommendation ITU-T H.761 v2 (2011), *Nested context language (NCL) and Ginga-NCL*.

[ITU-T H.762] Recommendation ITU-T H.762 (2011), *Lightweight interactive multimedia environment for IPTV services*.

[ITU-T H.763.1] Recommendation ITU-T H.763.1 (2010), *Cascading style sheets for IPTV services.*

[ITU-T H.764] Recommendation ITU-T H.764 (2012), *IPTV services enhanced script language.*

[ITU-T J.200] Recommendation ITU-T J.200 (2010), *Worldwide common core – Application environment for digital interactive television services*.

[ITU-T Y.1901] Recommendation ITU-T Y.1901 (2009), *Requirements for the support of IPTV services*.

[ITU-T Y.1910] Recommendation ITU-T Y.1910 (2008), *IPTV functional architecture*.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 application [b-ITU-T Y.101]: A structured set of capabilities which provide value-added functionality supported by one or more services.

3.1.2 application programming interface (API) [b-ITU-T Y.110]: This is an implementation interface between equipment and a software module and which does not have any physical realization as it is internal to the equipment.

3.1.3 end user [ITU-T Y.1910]: The actual user of the products or services.

NOTE – An end user consumes the product or service. An end user can optionally be a subscriber.

3.1.4 IPTV terminal device [ITU-T Y.1901]: A terminal device which has ITF functionality, e.g., an STB.

3.1.5 IPTV terminal function (ITF) [ITU-T Y.1901]: An end user function associated with:

a) receiving and responding to network control channel messages regarding session set-up, maintenance and tear-down;

b) receiving the content of an IP transport from the network and rendering.

3.1.6 linear TV [ITU-T Y.1901]: A television service in which a continuous stream flows in real time from the service provider to the terminal device and where the user cannot control the temporal order in which content is viewed.

3.1.7 metadata [ITU-T Y.1901]: Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment and management of the described entities.

3.1.8 middleware [b-ITU-T Y.101]: The mediating entity between two information elements. Such an element can be, for example, an application, an infrastructure component, or another mediating entity.

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

3.1.10 terminal device (TD) [ITU-T Y.1901]: An end-user device which typically presents and/or processes the content, such as a personal computer, a computer peripheral, a mobile device, a TV set, a monitor, a VoIP terminal or an audio-visual media player.

3.1.11 user agent [b-W3C WebArch]: One type of web agent; a piece of software acting on behalf of a person.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 script: A script is a program written in a scripting program language. Script is often interpreted from the source code. For example, a program written in ECMAscript is a script. On web-based terminal middleware for IPTV services, scripts are used in order to enable various services in IPTV terminal devices.

3.2.2 scripting program language: A scripting program language is a programming language that allows control of one or more software applications.

3.2.3 WBTM plugin: A WBTM plugin is software that may be added to generic web-based terminal middleware in order to provide additional functionality. WBTM plugins may be resident or be downloaded from the server side. The additional functionality provided by WBTM plugins can

be used by WBTM scripts. A WBTM plugin may be portable for service components, for a resource abstraction layer, for resources, or for none of these.

3.2.4 WBTM plugin API: These APIs are defined by the WBTM to call WBTM plugins to extend the functionalities and abilities provided by WBTM.

3.2.5 WBTM script: A script that is run on web-based terminal middleware in an IPTV terminal device and that enables one or more IPTV services, or a part of an IPTV service. The engine for WBTM scripts is recommended in the multimedia application framework series of Recommendations (ITU-T H.76x series). WBTM scripts may include lines to call additional functions that are implemented by WBTM plugins.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

AM Audience Measurement

API Application Programming Interface

CAS Conditional Access System

CSS Cascading Style Sheet

DHCP Dynamic Host Configuration Protocol

DOM Document Object Model

DRM Digital Rights Management

EPG Electronic Programme Guide

HTML Hypertext Markup Language

IDL Interface Description Language

IGMP Internet Group Management Protocol

LIME Lightweight Interactive Multimedia Environment

MAFR Multimedia Application Framework

MLD Multicast Listener Discovery protocol

NCL Nested Context Language

OS Operating System

PIP Picture In Picture

PVR Personal Video Recorder

RAL Resource Abstraction Layer

RSS RDF Site Summary

RTP Real-time Transport Protocol

SADS Service and Application Discovery and Selection

SCP Service and Content Protection

SI Service Information

SNF Service Navigation Function

SP Service Provider

STB Set-Top Box

TD Terminal Device

TV Television

VoD Video on Demand

WBTM Web-Based Terminal Middleware

5 Conventions

The following conventions are used in this Recommendation:

– The keywords "is required to" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

– The keywords "is prohibited from" indicate a requirement which must be strictly followed and from which no deviation is permitted, if conformance to this Recommendation is to be claimed.

– The keywords "is recommended" indicate a requirement which is recommended but which is not absolutely required. Thus, this requirement need not be present to claim conformance.

– The keywords "is not recommended" indicate a requirement which is not recommended but which is not specifically prohibited. Thus, conformance with this Recommendation can still be claimed even if this requirement is present.

– The keywords "can optionally" indicate an optional requirement which is permissible, without implying any sense of being recommended. This term is not intended to imply that the vendor's implementation must provide the option and the feature can be optionally enabled by the network operator/service provider. Rather, it means the vendor may optionally provide the feature and still claim conformance with this Recommendation.

6 Introduction

Web-based terminal middleware is terminal middleware whose characteristic is that it has one central middleware which orchestrates various applications. This orchestrating middleware, generally called "browser" or "user agent", processes a structured document and an interpretive language, usually called a "script", to enable various services. This typical use of web-based services is described in [ITU-T H.760].

Web-based IPTV terminal middleware supports basic and advanced interactive IPTV services in IPTV terminal devices. It is required to review IPTV service requirements and architecture, as well as IPTV terminal devices. IPTV architecture is described in detail in the ITU-T Y.19x series of Recommendations, and IPTV terminals are described in the ITU-T H.72x series of Recommendations. Web-based IPTV terminal middleware is needed to define the interfaces on IPTV terminal functional architecture and the structure of the presentation engine. The presentation engine basically supports the ITU-T H.76x series of Recommendations.

6.1 Application and middleware in IPTV architecture

Figure 6-1 gives the overview of the functional architecture for an IPTV system.

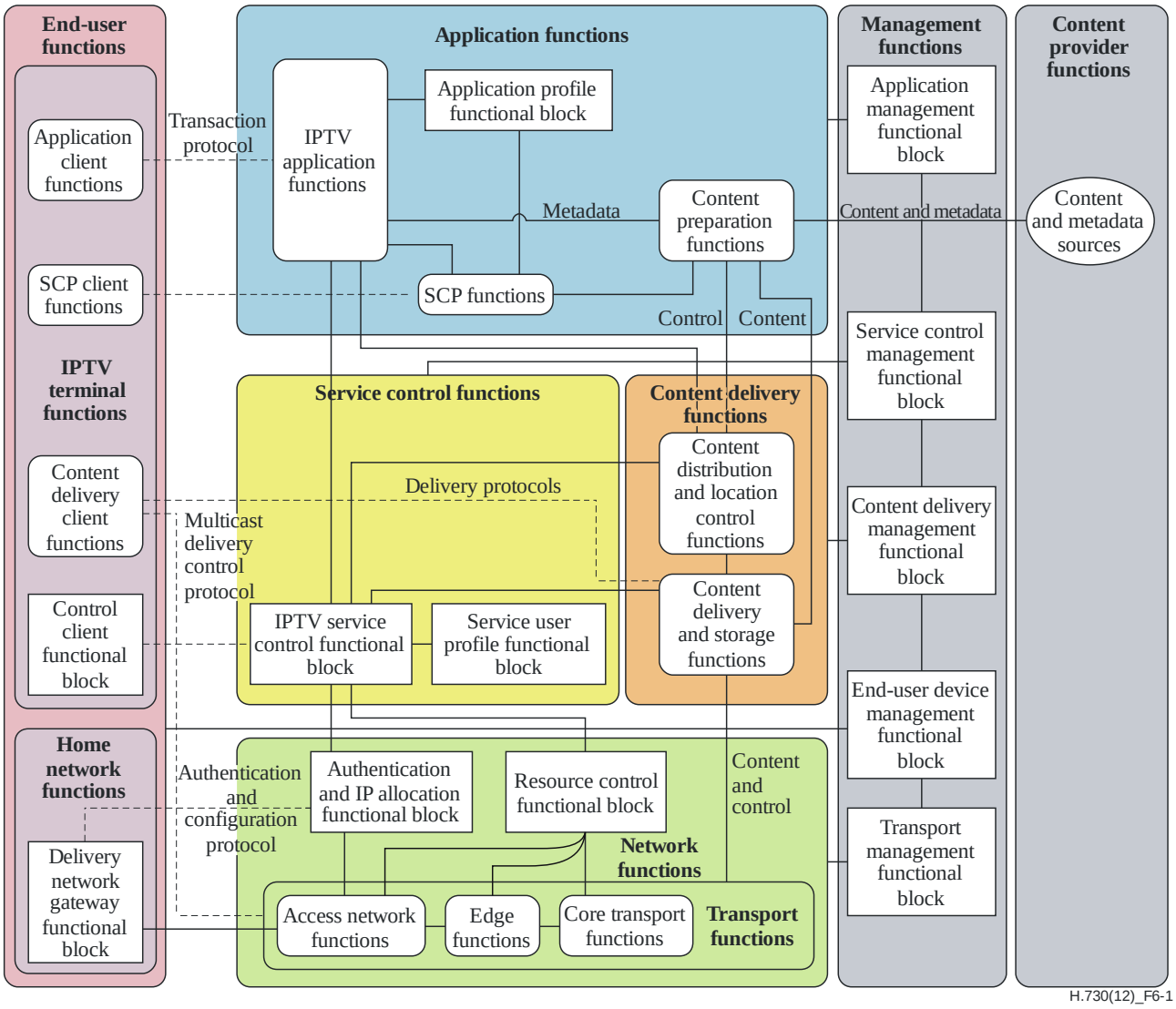

Figure 6-1 – IPTV functional architecture

According to [ITU-T H.720], terminal middleware is located on the terminal side. It is a mediating entity between two information elements in the terminal device, and WBTM could be various engines along with a set of high-level services (e.g., HTML, CSS, Lua and SESL [ITU-T H.764]).

According to [ITU-T H.720], IPTV terminal middleware is hardware-agnostic. As one type of implementation, WBTM in the IPTV terminal device provides various integrated functional blocks and APIs for high-level services, which could be programmed in a WBTM script or by other methods, to implement IPTV services with the integration of IPTV services such as VoD and linear TV.

6.2 Terminal middleware

Terminal middleware is the middleware which sits in the end user's premises and primarily enables IPTV terminal functions, see Figure 6-2. The terminal middleware in the IPTV terminal device is shown on the left-hand side.

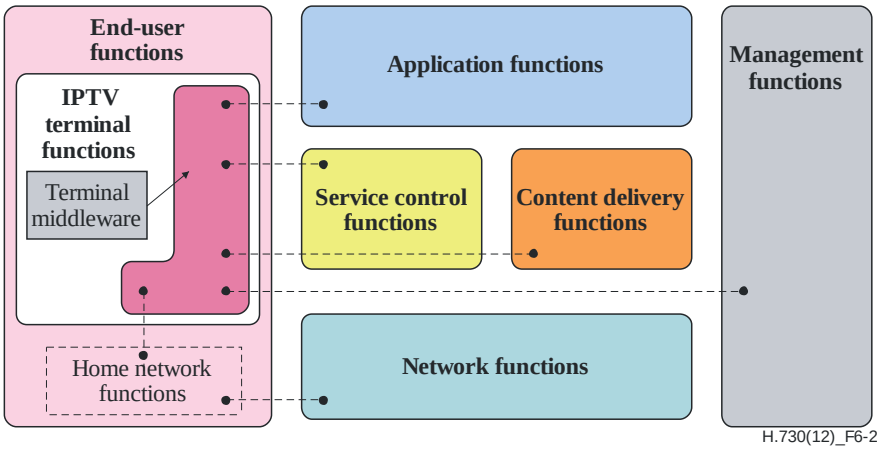

Figure 6-2 – IPTV terminal functions and terminal middleware

6.3 WBTM architectural overview

IPTV services can be deployed with web-based applications as well as middleware-based services for interactive IPTV data services. Many types of IPTV terminal device types can be used, which introduces a wide range of middleware environments: different middleware, application interfaces, content formats, etc. This heterogeneity makes service interoperability difficult. However, most devices have web-based standard user agents, which allow access to web content or support web applications. Therefore, web-based middleware can work as a common cross platform over different devices.

Also, IPTV services need to support seamless capabilities to users, regardless of device type, or across different service providers. Therefore, users can access the same IPTV service independently of the type of IPTV device. In this context, IPTV users and mobile IPTV users can enjoy the same services/applications over different IPTV devices for some types of services, e.g., social networking services, updated content notifications, etc. Therefore, IPTV services need to support interoperability for web-based applications among different devices.

To share such kinds of web applications or content over different IPTV devices, web-based middleware covers some service capabilities, as well as the web-based core engine.

Figure 6-3 gives the layered view of WBTM architecture.

Appendix I describes certain use cases of services using WBTM.

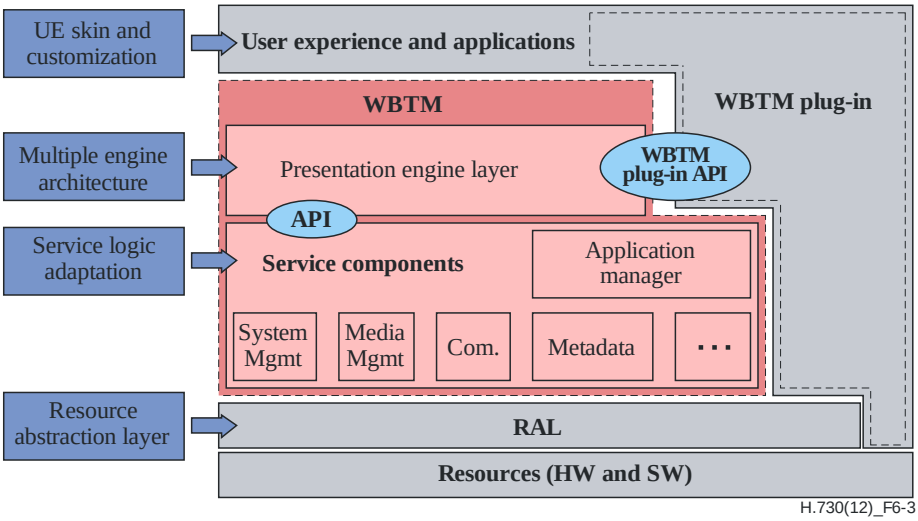

Figure 6-3 – Layered view of IPTV web-based terminal middleware architecture

7 Interfaces

Figure 7-1 shows the IPTV TD functional architecture recommended in [ITU-T H.720]. WBTM is recommended to have interfaces with those functions in IPTV TDs.

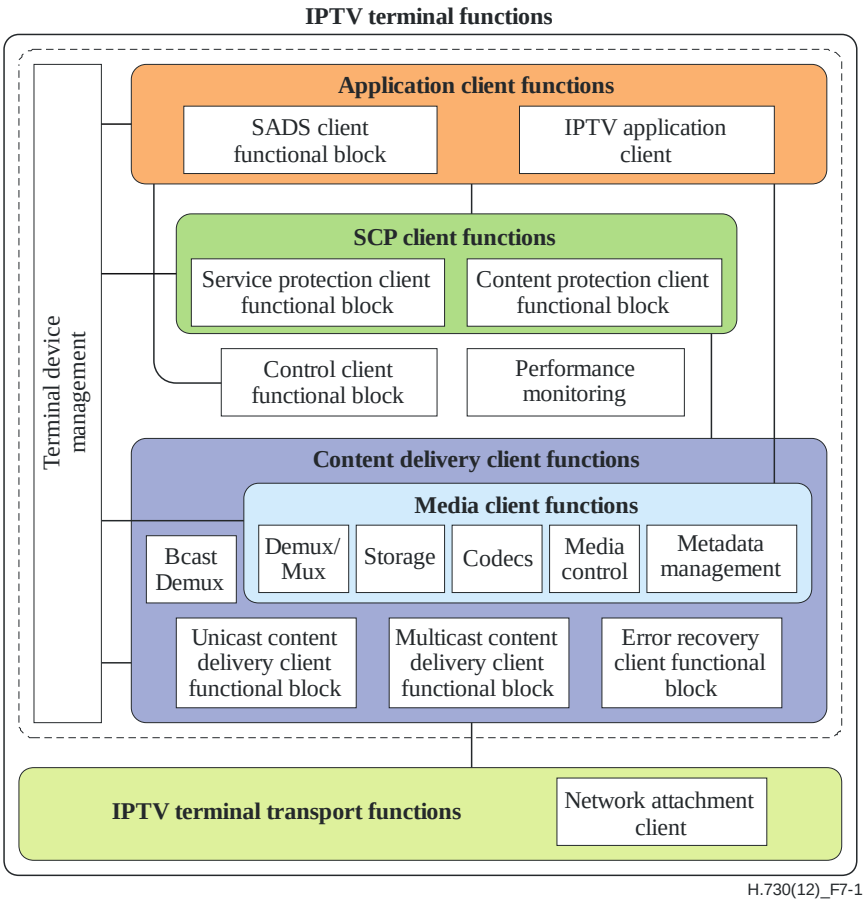

Figure 7-1 – Functional architecture block diagram of IPTV terminal devices [ITU-T H.720]

7.1 IPTV terminal transport functions interfaces

This is the interface between WBTM and the IPTV TD transport functions.

The handling of the IP-based connection between the delivery network gateway and IPTV TD, or between the IPTV network and IPTV TD is performed through this interface.

7.1.1 Network attachment client interface

This is the interface between WBTM and the IPTV TD network attachment client component.

In a web-based service discovery scenario, the network attachment client interface may be relevant. Otherwise, this lies beyond the scope of this Recommendation.

7.2 Content delivery client functions interface

This is the interface between WBTM and IPTV TD content delivery client functions.

Reception and control of content delivery from the content delivery and storage functions is enabled through this interface.

7.2.1 Broadcast demux interface

This is the interface between WBTM and an IPTV TD in a hybrid environment.

IPTV TD supports both IP content reception and non-IP content reception for terrestrial, cable or satellite broadcast services.

7.2.2 Multicast content delivery client functional block interface

This is the interface between WBTM and an IPTV TD multicast client functional block.

Multicast protocols, such as IGMP for IPv4 or MLD for IPv6, are handled through this interface.

7.2.3 Unicast content delivery client functional block interface

This is the interface between WBTM and the IPTV TD unicast content delivery client functional block.

For example, RTP and HTTP (for unicast content on-demand) are supported through this interface.

7.2.4 Error recovery client functional block interface

This is the interface between WBTM and IPTV TD error recovery client functional block.

This lies beyond the scope of this Recommendation.

7.3 Media client functions interfaces

This is the interface between WBTM and IPTV TD media client functions.

7.3.1 Media control interface

This is the interface between WBTM and the IPTV TD media control functional entity.

Control of video and audio components, as well as other components (such as metadata handling, content storage including PVR control, and play/reproduction of content including streaming data), is handled through this interface.

7.3.1.1 Video interface

This is the interface between WBTM and the IPTV TD video control component.

7.3.1.2 Audio interface

This is the interface between WBTM and the IPTV TD audio control component.

7.3.1.3 Other data format interface

This is the interface between WBTM and an IPTV TD for other multimedia data formats that are commonly used, such as text (i.e., closed caption) and graphics.

7.3.2 Storage interface

This is the interface between WBTM and the IPTV TD storage functional entity.

Caching and storage of content and other application data is enabled through this interface.

7.3.3 Metadata management interfaces

This is the interface between WBTM and the IPTV TD metadata management functional entity.

Accessing and retrieving metadata and management of the local cached metadata is enabled through this interface.

7.4 SCP client functions interfaces

This is the interface between WBTM and IPTV TD SCP client functions.

7.4.1 Service protection client interface

This is the interface between WBTM and the IPTV TD for the commonly used service protection function. This interface mediates service protection such as service authentication, service channel protection, etc.

7.4.2 Content protection client interface

This is the interface between WBTM and the IPTV TD for commonly used content protection functions. This interface mediates content protection functions such as content decryption, rights and keys management.

7.5 Application client functions interface

This is the interface between WBTM and IPTV TD application client functions. There are three types of application client functions:

1) Basic functions: applications include software components capable of enabling functional and observable behaviour. Examples include GUI, SNF, VoD controls, SCP applications and other service-related applications.
2) Management functions: some applications are responsible for basic management of the IPTV TD, such as power management and event management.
3) Service support functions: some applications are responsible for support services, including inter alia plugin applications, browser applications, media player applications or user agents in general.

7.5.1 IPTV application client interface

This is the interface between WBTM and IPTV TD application client functions.

This interface accesses the following service-specific functional blocks:

1) On-demand client functional block: This functional block interacts with its server-side counterpart to perform session management, service authorization, presentation of the content metadata, and execution of the service logic for on-demand applications.
2) Linear TV client functional block: This functional block interacts with its server-side counterpart to perform session management, service authorization, presentation of the content metadata, and execution of the service logic for linear TV applications.
3) Other client functional blocks: These functional blocks interact with the other server-side blocks relevant to the delivery and presentation of additional IPTV services and their content (e.g., games, distant learning, audience measurement), and for the support and management of modification, adaptation or integration of services and content (e.g., uploading through a USB client interface of a family picture with a particular template postcard provided from a server, or a video taken with a personal recorder with additional text).

7.5.2 SADS client functional block interface

This is the interface between WBTM and the IPTV TD service and application discovery and selection (SADS) client functional block.

This interface provides the end user's discovery and selection of IPTV services and applications.

7.6 Connection and session management interface

This is the interface between WBTM and the IPTV TD connection and session management.

7.7 Terminal device management interface

This is the interface between WBTM and the IPTV TD terminal device management functional entity.

Setting terminal configuration values, e.g., local configuration, network interface, AV interface, is enabled through this interface.

7.8 Performance monitoring interface

This is the interface between WBTM and the IPTV TD performance monitoring component.

8 Web-based engine structure

This clause identifies the modules that compose the structure of presentation engines for web-based IPTV terminal middleware. Web-based IPTV terminal middleware defines a user agent (e.g., browser and document renderer) as a presentation engine. Web-based middleware is based on web standards with declarative formats as illustrated in Figure 8-1.

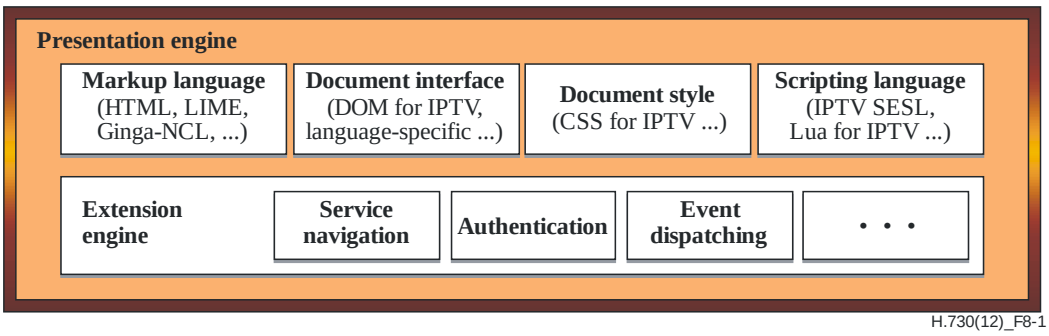

Figure 8-1 – Web-based engine structure

8.1 Markup language

Markup language support for IPTV applications includes a specialization of HTML for IPTV services (currently under study within ITU-T), LIME [ITU-T H.762] and Ginga-NCL [ITU-T H.761].

8.2 Document access interface

Document access interface is the interface specification that provides document access and editing, such as the DOM profile for IPTV (currently under study within ITU-T) or NCL editing commands in [ITU-T H.761].

8.3 Document style

Document style is the style description for web documents, such as the CSS profile for IPTV [ITU-T H.763.1].

8.4 Scripting language

The scripting language is the element that allows for embedding imperative behaviour to web documents, like the service enhanced script language (SESL) profile for IPTV in [ITU-T H.764] or the Lua profile for IPTV currently under study within ITU-T.

8.5 Extension engine

The extension engine provides additional service navigation and discovery according to the service information provided by using the extended script.

Besides the basic requirement of a user agent, extended mark-up elements and scripts are used to expand the user experience. Moreover, extended script could be used to accomplish user interactivity, such as service navigation, authentication and event dispatch.

Moreover, the presentation engine layer could directly access the resource abstraction layer (RAL) [ITU-T H.720] for the resource layer to gain better performance, through a WBTM plugin API, for example. Web applications could be downloaded by the user agent to provide temporary service [ITU-T J.200].

9 WBTM for IPTV services

9.1 WBTM for basic IPTV services

The following functionalities are provided by web-based terminal middleware for basic IPTV services, such as those defined in [ITU-T H.721].

9.1.1 Licensing

– The function to get an IPTV licence: obtain the licence for the specified content.

– The function to get IPTV licence information: obtain information concerning the specified licence.

– The function to get a DRM ID: obtain the identifier of the CAS/DRM client supporting the specified CAS/DRM.

9.1.2 Content initialization

– The function to launch IPTV content: to initialize IPTV content by launching it.

9.1.3 Service registration

– The function to set IPTV service registration information: to set the basic registration information of linear IPTV and VoD services.

– The function to check IPTV service registration information: to confirm the basic registration information of linear IPTV and VoD services.

9.1.4 Communication of licence information

The function to set content package information: set the information for the purchased content package.

The function to update package licence information: update the information of the licences for all package content.

9.1.5 Page-transition control

The function to launch an unmanaged document: changes to a document in the unmanaged state.

The function to get the document's management status: obtains the information on the management status of the document.

9.1.6 Text animation control

The function to display marquee text: displays the strings as marquee, e.g., in the "p" element of HTML specialization for IPTV services currently under study within ITU-T.

9.1.7 Parental control function

The function to check the parental control password: confirms the password for ensuring parental control.

9.1.8 Non-volatile memory access

The function to read and write from the local cache memory: access the non-volatile memory.

9.2 WBTM for advanced IPTV services

The service components for advanced services can be designed as "plugins" that can be added selectively on web-based terminal middleware. These plugins interact with host applications that reside either on web-based terminal middleware or on the server-side to provide enhanced services. Other plugins in web-based terminal middleware, which are called WBTM plugins, interact with functions residing either on service components, on the RAL or on resources. They can be deployed separately, depending on what kind of service is to be offered. The enhanced services that can be provided by WBTM plugins and WBTM scripts on web-based IPTV terminal middleware are listed in Table 1.

Table 1 – List of enhanced services

Presence service:	Provides services using information on the end user's service usage state and statistics.
Communication service:	Provides functionalities for the end user to exchange information such as voice, video and data and includes many types of communication such as telephony, video telephony, video conference, messaging, chatting, etc.
Caller ID service:	Provides features allowing the Caller ID of the incoming call to be displayed on the terminal device.
Personalization service:	Provides services to support personal settings or to recommend preferable content or services.
Widget service:	Provides running environments for widget application.
Multi-device/screen service:	Provides methods and protocols for multimedia presentation over multiple devices.
Feeds reader service:	Provides services to subscribe, get notification on updates, and read contents with RSS and Atom.
Audience measurement service:	Provides services to collect and deliver audience measurement data. WBTM scripts may provide the function to deliver audience measurement data to an audience measurement server, and WBTM plugins may provide the function to collect audience measurement data by communicating with service components or other components in the IPTV terminal device. Appendix III shows an example of web-based terminal middleware architecture with the audience measurement service.
PIP service:	Provides the capability to display different media content on the screen, which is composed of two or more video screen windows.
PVR service:	Provides a personal video record function so that end users can retrieve and play back the recorded video later. For example, the PVR will be able to support a personal channel service which allows generation of the end user's preview schedule, customized according to his/her preferences or lifestyle.

Table 1 – List of enhanced services

Advertising service:	Provides IPTV advertising services which are quite different from traditional linear TV advertising. The IPTV terminal device is capable of supporting additional advertising services such as targeted advertising and on-demand advertising of the full-fledged terminal device model currently under study within ITU-T.

Annex A

General requirements for IPTV WBTM

(This annex forms an integral part of this Recommendation.)

Table A.1 – General requirements for IPTV WBTM

	Item	Note
1 General requirements		
1) Middleware requirements	R-01: IPTV WBTM is required to support terminal device start-up and the initialization function.	Clause 7.1
	R-02: IPTV WBTM is required to support server-side device start-up and the initialization function.	N/A
	R-03: IPTV WBTM, if trick mode is supported, it is required to support play, pause and stop functions.	Clause 7.3.1
2) Middleware recommendations	RR-01: IPTV WBTM is recommended to support application management (e.g., application lifecycles, application states).	Clause 7.5
	RR-02: IPTV WBTM is recommended to be hardware and operating system (OS) independent.	Common part
	RR-03: IPTV WBTM is recommended to support pause in a PVR-capable terminal or system.	Clause 7.3.1
	RR-04: IPTV WBTM is recommended to manage skip forward and skip backward functions (e.g., by time period, interval, location in content).	Clause 7.3.1
	RR-05: IPTV WBTM is recommended to support executing multiple simultaneous applications.	Clause 7.5
	RR-06: IPTV WBTM is recommended to support a means to change the style of EPG.	Clause 7.3.3
	RR-07: IPTV WBTM components are recommended to facilitate service processes interaction among IPTV devices (e.g., a server and its clients, such as a STB, security system, or VoD server).	Common part
	RR-08: IPTV WBTM is recommended to be able to manage the IPTV application profile information.	Clause 7.5
	RR-09: IPTV WBTM is recommended to support presentation capabilities for multimedia data (e.g., audio, video, graphics, text and images) providing the browsing, the synchronization and the interaction of such data with the end user [ITU-T Y.1910].	Clause 7.5

Table A.1 – General requirements for IPTV WBTM

	Item	Note
3) Middleware options	OR-01: IPTV WBTM can optionally support the decoding of service information (SI).	Clause 7.3.3
	OR-02: IPTV WBTM can optionally support a shortcut mechanism for selection.	Clause 7.3.3
	OR-03: IPTV WBTM can optionally support metadata compression.	Clause 7.3.3
2 Middleware application programming interfaces		
1) Middleware requirements	R-01: IPTV WBTM is required to provide an API for stopping and starting the presentation of video and audio.	Clause 7.3.1
	R-02: IPTV WBTM is required to provide an API to communicate with service providers to implement media transmission and media control functions.	Clause 7.3.1
	R-03: IPTV WBTM is required to provide an API to access metadata information.	Clause 7.3.3
2) Middleware recommendations	RR-01: IPTV WBTM is recommended to support an API to access information on removable storage devices (e.g., USB key).	Clause 7.3.2
	RR-02: IPTV WBTM is recommended to support an API for controlling (e.g., selecting, showing and hiding) subtitles and a closed captioning display.	Clause 7.3.1
	RR-03: IPTV WBTM is recommended to include an API to access user preferences (e.g., accessibility features and display settings) available on a removable storage device (e.g., smartcard).	Clause 7.3.2
	RR-04: IPTV WBTM is recommended to provide service enabling APIs.	Clause 7.5
	RR-05: IPTV WBTM APIs are recommended to be open, flexible, granular, self-contained and components-based.	Common part
	RR-06: IPTV WBTM is recommended to provide an API to support a variety of mixed media formats to be presented together (e.g., an HTML text page embedding a video stream).	Clause 7.3.1
	RR-07: IPTV WBTM is recommended to provide an API to support picture-in-picture.	Clause 7.3.1
	RR-08: IPTV WBTM is recommended to provide an API to manage captioning including the selection from a range of languages, speed and verbosity.	Clause 7.3.1
3) Middleware options	OR-01: IPTV WBTM can optionally provide an API for recording programmes locally.	Clause 7.3.1

Appendix I

IPTV service model with WBTM

(This appendix does not form an integral part of this Recommendation.)

This use case illustrates how WBTM can be applied in IPTV services. Several use cases are possible and the example here focuses on two types of service. The first is a general service case illustrating HTML content service with a web portal server. The other example explores an enhanced IPTV service case where an IPTV community portal service provides services amongst IPTV service users.

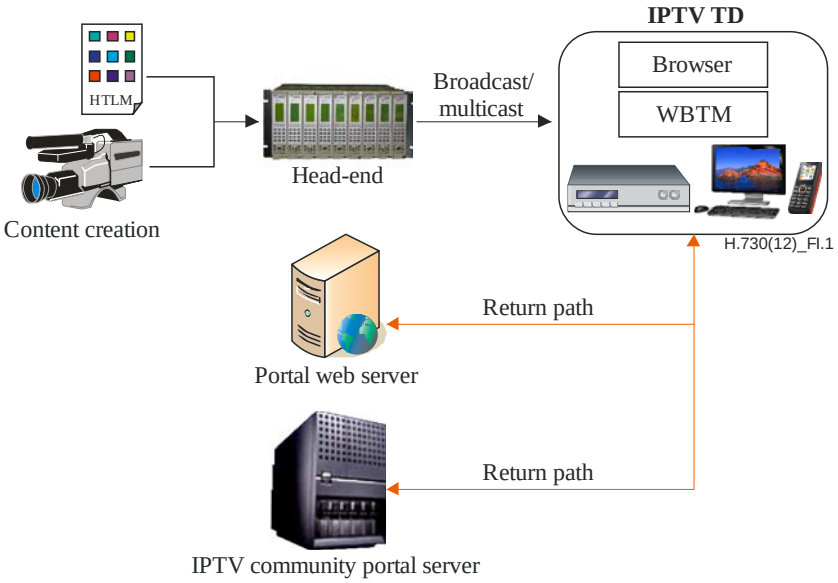

Figure I.1 – Example of IPTV services with WBTM

I.1 Use case: General IPTV service

Figure I.1 shows a conceptual service from web content creation delivery to an IPTV terminal, and then the return path for interactive operation. WBTM can be used with ITU-T IPTV various classes of terminal, e.g., basic terminal mode, fully-fledged mode or mobile terminal mode.

I.2 Use case: Enhanced IPTV service (IPTV community portal service)

The support of better user service experience requires enhanced IPTV service support in IPTV terminals using WBTM (see also clause 8, on service components to support enhanced services). These service components use mixed functional entities and are implementation-dependent. An IPTV community portal service is a popular enhanced IPTV service. The example in Figure I.1 also shows functionalities that are required for an IPTV community portal service.

An IPTV community portal server can have several components: user management, metadata management, content management, media management, etc. The user management component manages a user's login history information and account management. The metadata management component comprises processing for metadata management, electronic programme guide (EPG) management and syndication operation. The content management component is provided for page management, content search and content transformation. The media management component is for processing A/V media content management and video tagging processing.

Appendix II

Audience measurement architecture in web-based terminal middleware

(This appendix does not form an integral part of this Recommendation.)

Figure II.1 shows an example of audience measurement in WBTM. The "AM data delivery" script denotes a WBTM script that delivers audience measurement data (AM data) to aggregation functions [b-ITU-T H.741.0]. This WBTM script may be downloaded from a server. "AM data collection" refers to a WBTM plugin that collects audience measurement data to communicate with media management functionality in service components. Both the AM data delivery WBTM script and AM data collection WBTM plugin provide audience measurement services in IPTV terminal devices with web-based terminal middleware. For example, An AM data delivery WBTM script may be written using functions to access web servers. If the AM data collection WBTM plugin is written using portable interfaces between WBTM and service components, the plugin will also be portable.

Figure II.2 shows another example of audience measurement in WBTM. The "AM data management" WBTM script and the "AM data collection and delivery" WBTM plugin implement the audience measurement service in the IPTV terminal device with web-based terminal middleware. "AM data management" denotes a WBTM script that manages the delivery timing from an IPTV terminal device to the aggregation functions. It sets the timing to run the "AM data collection and delivery" WBTM plugin using the presentation engine. This plugin collects audience measurement data by communicating with the media management functions in the service components and delivers the AM data using the RAL. In this example, delivery protocols between "AM data collection and delivery" WBTM plugin and aggregation functions can be selected independently from the specification of [b-ITU-T H.750] and other IPTV multimedia application framework Recommendations (ITU-T H.76x series).

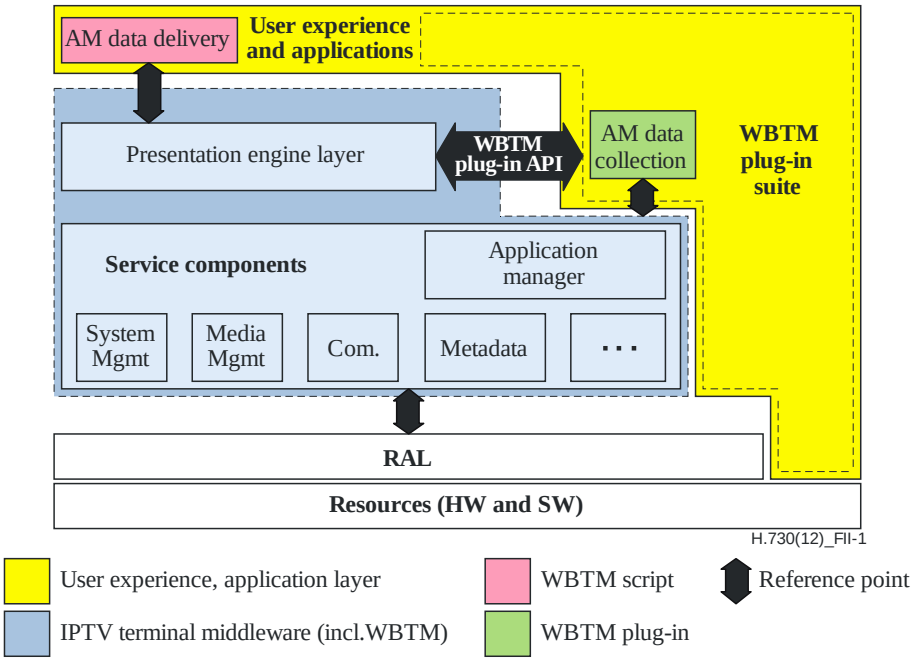

Figure II.1 – Audience measurement architecture in WBTM

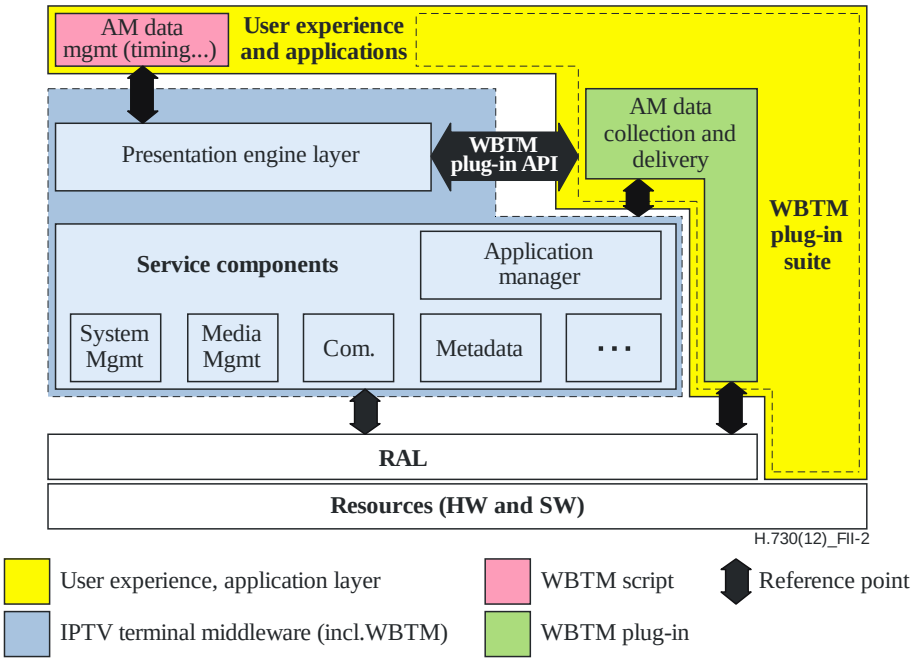

Figure II.2 – Another web-based terminal middleware architecture with AM service

Appendix III

Examples of script and plugin operation modes in WBTM

(This appendix does not form an integral part of this Recommendation.)

This appendix describes the relationship between applications and IPTV terminal middleware, and between "script" and WBTM.

III.1 Relationship between application and IPTV terminal middleware

In Figure III.1, the highest layer refers to user experience and applications, and IPTV terminal middleware comprises all other parts except for the resources (HW & SW) layer and RAL. In web-based terminal middleware for IPTV devices, a script is an application located in the highest layer of the IPTV terminal software architecture. The highest layer application may consist of several scripts (WBTM and non-WBTM ones), as well as non-script applications (WBTM plugin).

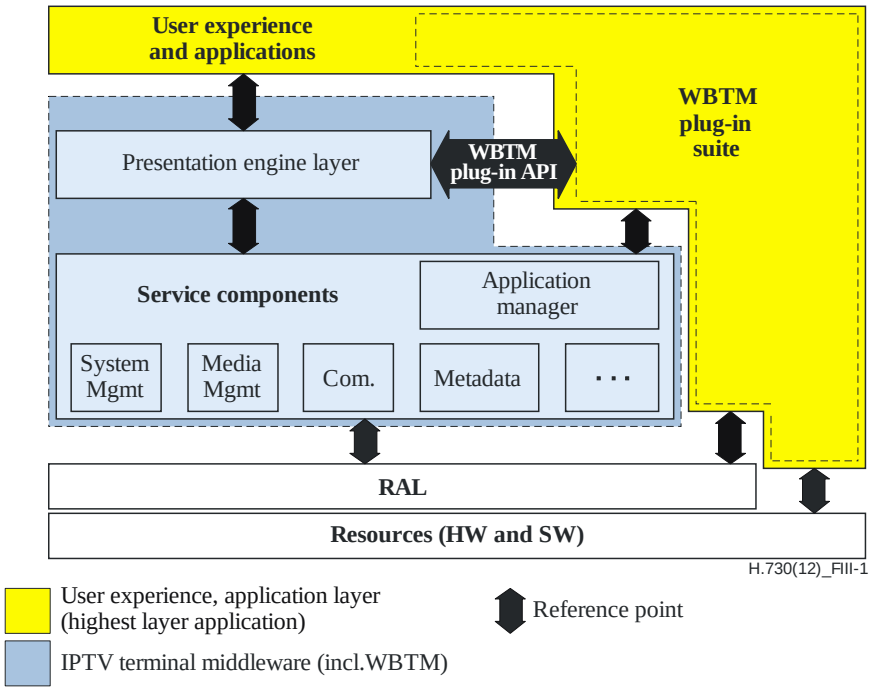

Figure III.1 – IPTV terminal middleware and the highest layer

Figure III.2 shows five types of applications in the highest layer application, denoted A1 to A5. The applications may be resident or downloadable. In the figure, A1 is a WBTM script and A2 is a WBTM plugin application (which may itself contain scripts) that uses WBTM functions and service component functions, while A3, A4 and A5 are WBTM plugin applications that directly use service component functions, RAL functions, and functions in resources, respectively. One should note that while both A4 and A5 use functions that are not provided by IPTV terminal middleware, A4 is portable (since it is based on RAL) and A5 is not portable (implementation-specific). All WBTM plugins are IPTV terminal applications.

As WBTM scripts have the highest portability, it is important that they be clearly specified in ITU-T Recommendations. In order to extend the ability of A1 while keeping the highest portability, the ability of WBTM is extended by the means of WBTM plugins. In Figure III.2, WBTM can only use functions provided by service components.

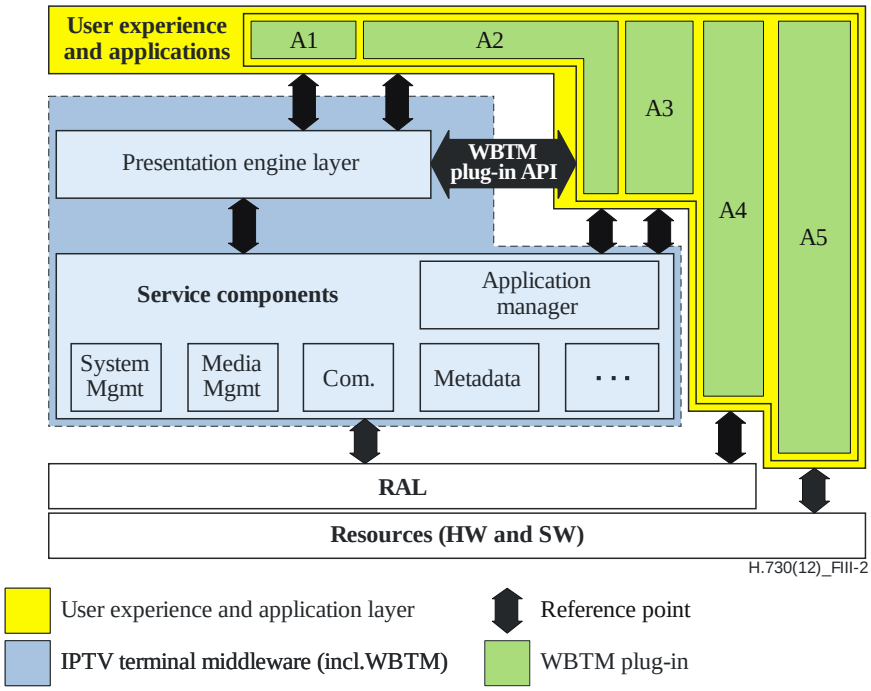

Figure III.2 – Relationship between applications in the highest layer application and the lower layers

Figure III.3 shows the use of WBTM plugins to gain extended portability. Reference point R3 is the interface with RAL, and R4 is the interface with the resources layer. Application A6 is a WBTM plugin application with WBTM scripts, which is called by the WBTM via a WBTM script and uses RAL functions through interfaces R1 and R3. Application A6 also uses functions in service components through R2 and R5. Though A6 uses functions provided by RAL, A6 keeps the highest portability by use of R1 and R3.

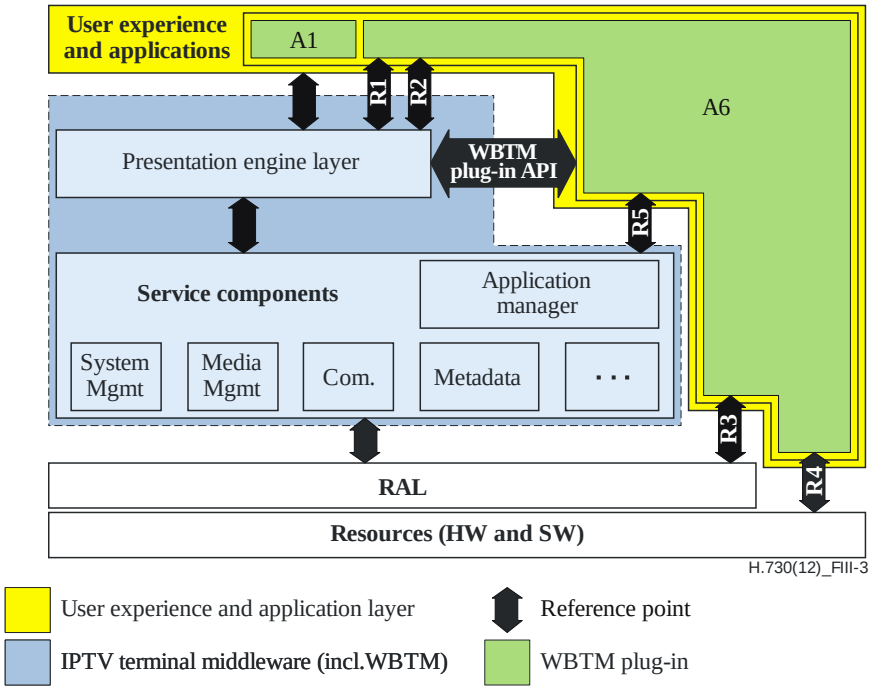

Figure III.3 – Reference points from WBTM to other components

III.2 Relationship between WBTM and WBTM plugin

Figure III.4 shows the location of both WBTM and the WBTM plugin in an IPTV terminal device. The WBTM plugin can be called from the WBTM script as A7 and A8. WBTM script A7 is a WBTM script that needs WBTM plugin P1, and P1 uses one or more functions in the service component. These functions are not included in the generic function of WBTM. WBTM script A8 needs WBTM plugin P2. P2 uses one or more functions in the service component as P1 does and uses functions in the resource abstraction layer. As P1 is independent from the RAL, it is more portable than P2.

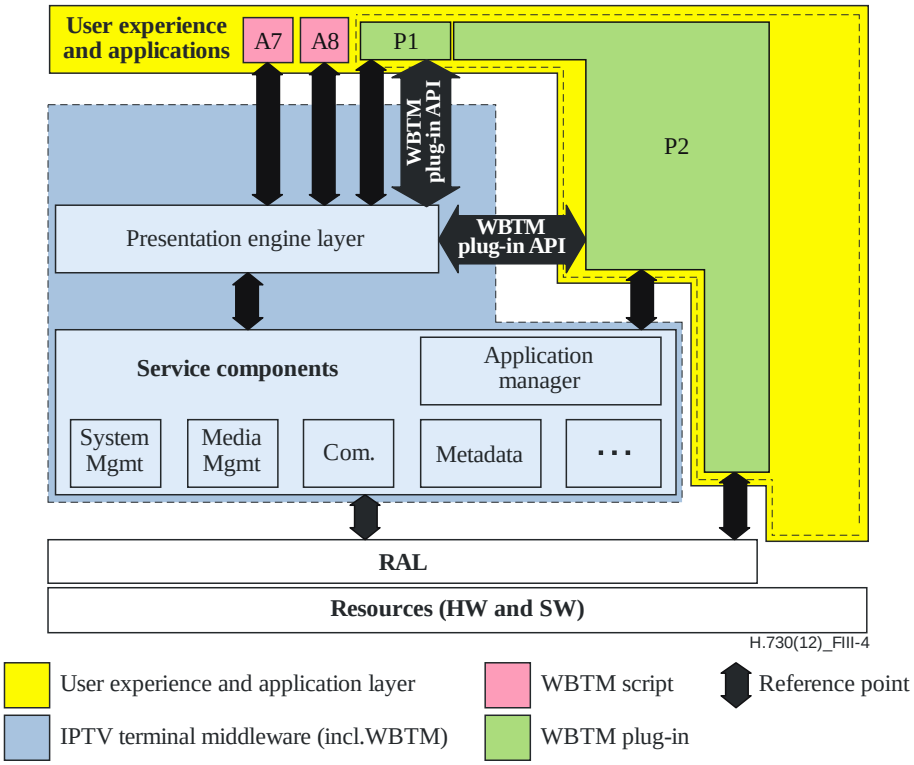

Figure III.4 – WBTM plugin and WBTM

Figure III.5 shows another example of a WBTM plugin. The difference between Figure III.4 and Figure III.5 is the WBTM plugin. The interface between WBTM script A8 and WBTM is the same in figures III.4 and III.5. This means that WBTM plugins P2 and P3 provide the same interface for WBTM scripts. The difference between P2 and P3 is the interfaces between WBTM and its lower layer. As P3 directly uses RAL and the resources layer, P3 may be faster than P2. However, P3 is dependent on the resources layer, hence it is less portable than P2.

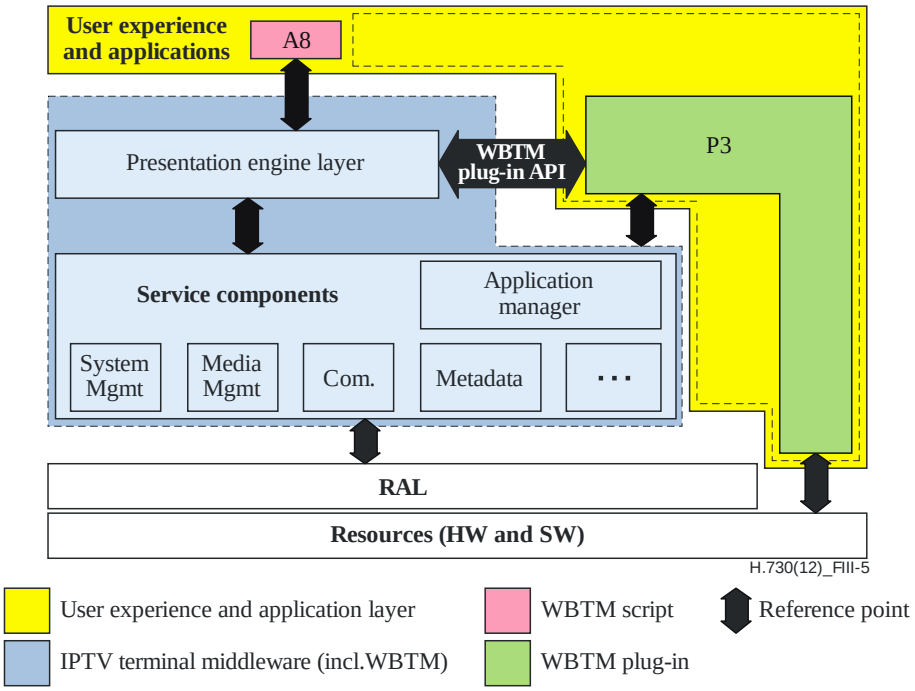

Figure III.5 – Another WBTM plugin that calls RAL and WBTM

Figure III.6 shows the most complicated example of the relationship between WBTM and WBTM plugins. A9 is a WBTM script which is executed by the WBTM presentation engine and it calls WBTM plugin P4 via the WBTM script APIs. P4 is a WBTM plugin using a WBTM script, service component and RAL. Since P4 contains the WBTM script A10, A10 calls the WBTM plugin P5 via the WBTM script APIs to gain the additional abilities implemented by P5.

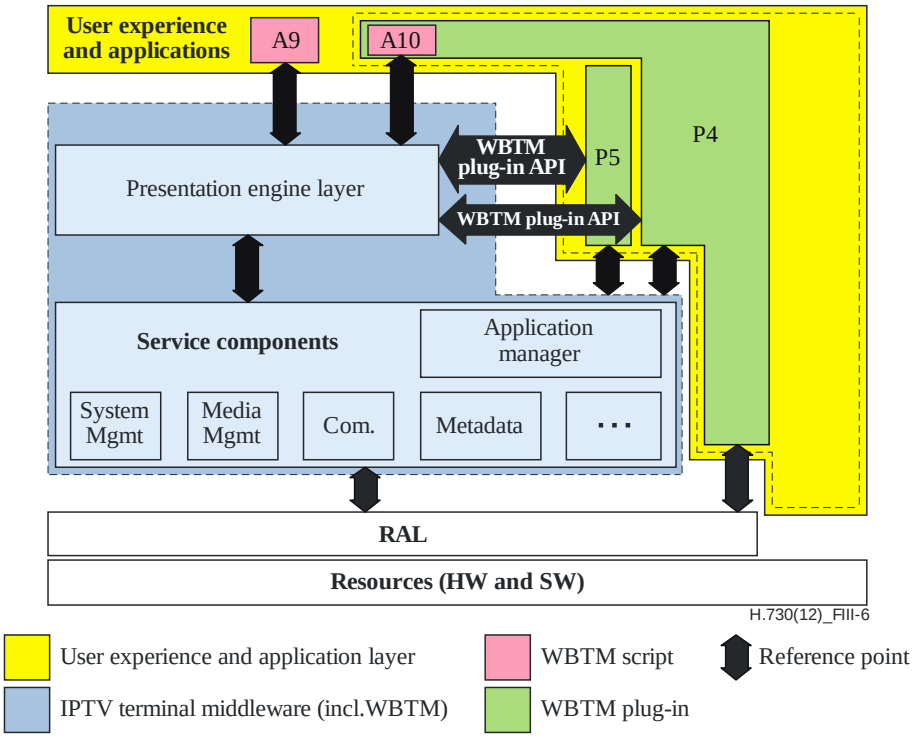

Figure III.6 – WBTM script calls WBTM plugin

Appendix IV

Examples of WBTM overall workflow with the ITU-T H.76x series of Recommendations

(This appendix does not form an integral part of this Recommendation.)

IV.1 Use case 1: Service display workflow

The user agent in the presentation engine extends its ability with the help of an extension engine to meet the needs of IPTV services' requirements. In Figure IV.1, the web document programmed using a Recommendation of the ITU-T H.76x series is located in the user experience and applications layer, and can be executed by the user agent. The end user experiences the IPTV service via the web documents, which contain EPG information, executed and displayed by the user agent. In order to provide the IPTV service, the terminal device will implement the following steps shown in Figure IV.1:

Step 1: The user agent in the WBTM downloads the web documents from the web server in the IPTV service platform.

Step 2: The user agent analyses the web documents and executes the scripts (e.g., SESL [ITU-T H.764], Lua profile for IPTV (currently under study within ITU-T), etc.).

Step 3: The user agent calls the functional block "Media Management" defined in the service components layer to communicate with the streaming server in the IPTV service platform and the streaming server feeds back the required video stream.

Step 4: The media management component receives and decodes the video from the streaming server.

Step 5: The decoded video output from the media management component and web documents displayed by the user agent are sent together to the output service component.

Step 6: The output service component refreshes the display on the TV.

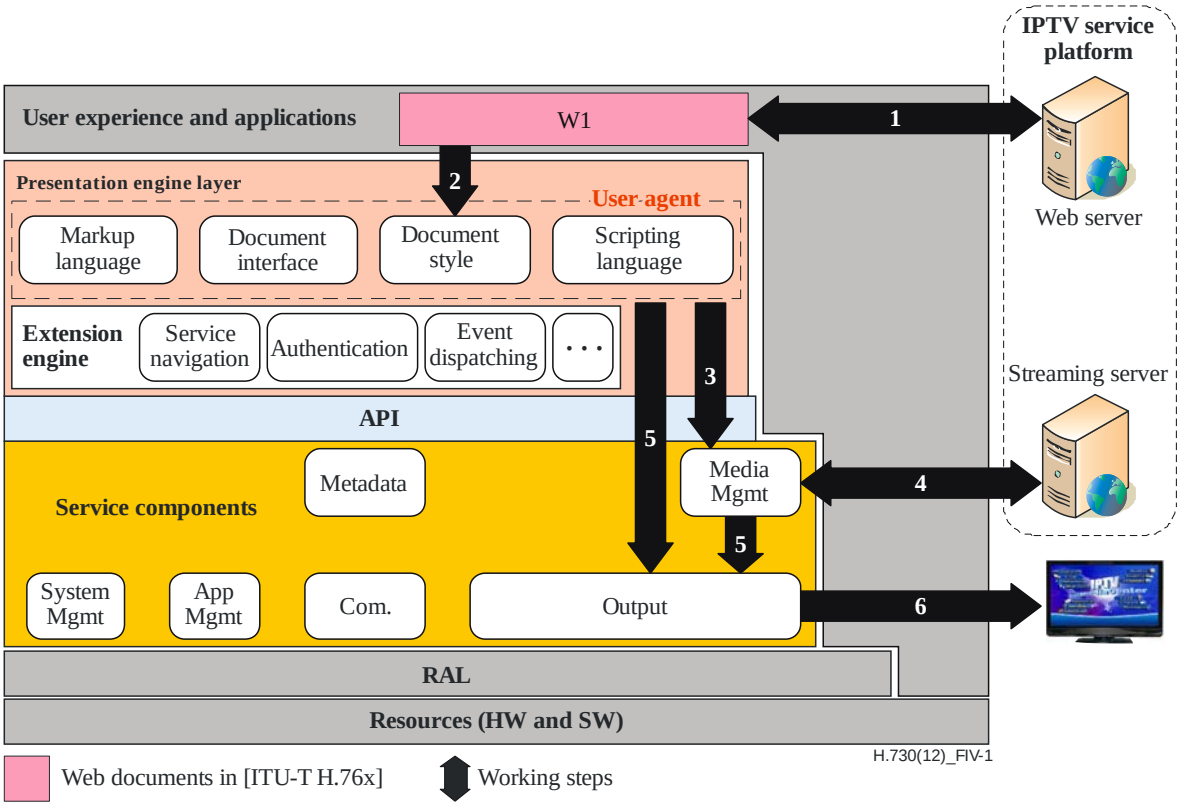

Figure IV.1 – Service display working flow using WBTM

IV.2 Use case 2: User interactive working flow with media management

Clause 7.3 defines the interfaces between WBTM and IPTV TD media client functions and clause 7.3.1 defines how to control video/audio/other media.

Web-based IPTV middleware defines a user agent (e.g., browser, document renderer, etc.) as a presentation engine, and web standards with declarative formats are used. Since the scripting language specifications that allow embedding imperative behaviour to web documents are part of the web-based presentation engine, a detailed interface definition is required by scripting languages (e.g., IPTV SESL [ITU-T H.764] and the Lua profile for IPTV (currently under study within ITU-T)) between WBTM and IPTV TD media client functions:

Media control interface: Media control interfaces are required for providing STB capabilities that control the life cycle of the media and provide methods like pause, resume, fast-forward, fast-rewind, etc.

Video interface: Video interfaces are required for providing video-related methods that support functionalities like media codec selection, video output control, etc.

Audio interface: Audio interfaces are required for providing audio-related methods that support functionalities like audio channel switching, audio codec selection, etc.

In this example, after the user watches the web documents and decoded the video stream on a TV, he selects a VoD programme to play. During the VoD's playback, the user presses the "PAUSE" key on the remote control to pause the VoD. Figure IV.2 illustrates the flow of how the WBTM handles the procedure to interact with the user; this is described below:

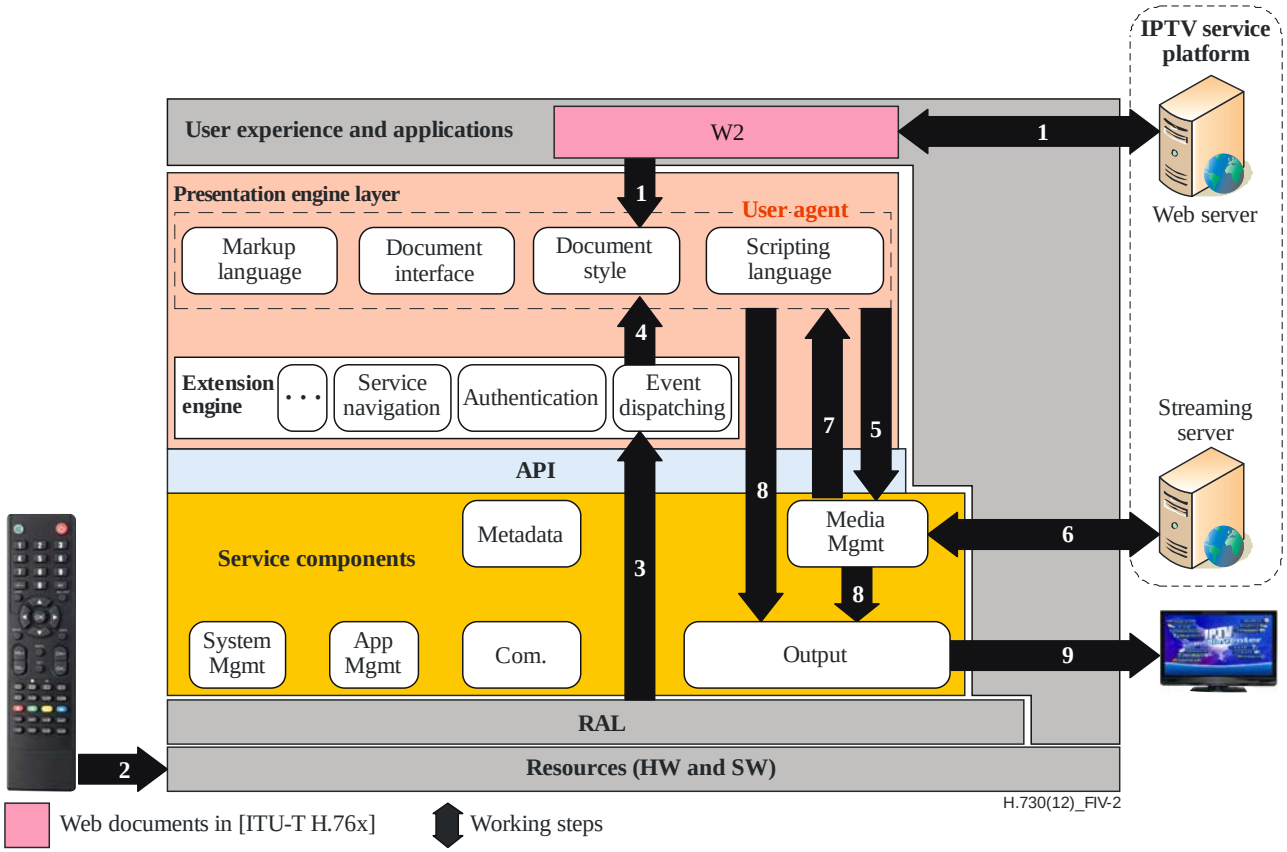

Figure IV.2 – Working flow of user interaction handled by WBTM

Step 1: The user agent downloads the web documents with WBTM scripts from the web server in the IPTV service platform to play the VoD.

Step 2: The user presses the "PAUSE" key on the remote.

Step 3: The key pressed status is sent to the "Event Dispatching" block in the extension engine through the service components layer.

Step 4: The user agent in the presentation engine layer receives the "PAUSE" key press event and executes the WBTM scripts (e.g., SESL, Lua, etc.) to handle the event.

Step 5: The user agent calls the media management service component.

Step 6: The media management service component communicates with the streaming server in the IPTV service platform to pause the video stream.

Step 7: The media management service component feeds back the successful communication result to the user agent.

Step 8: The media management service component refreshes the video display and the user agent refreshes the web document display according to the operation result.

Step 9: The output service component refreshes the display on the TV.

Appendix V

An implementation example for WBTM with interface description language

(This appendix does not form an integral part of this Recommendation.)

This appendix provides an example of an implementation where WBTM can work on a terminal device using an interface description language (IDL).

V.1 Use case 1: WBTM interface list

The example in Figure V.1 shows the overall function list to support interactive IPTV applications, which are implemented via WBTM.

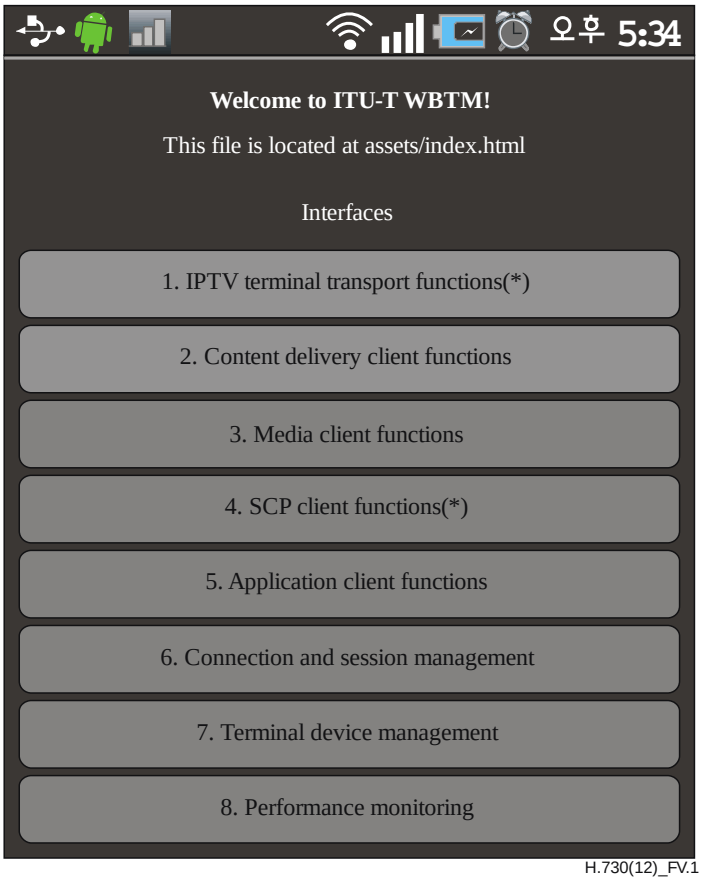

Figure V.1 – WBTM interface list

V.2 Use case 2: Media client functions interfaces for WBTM

Figure V.2 shows an example of media client that supports media-related information.

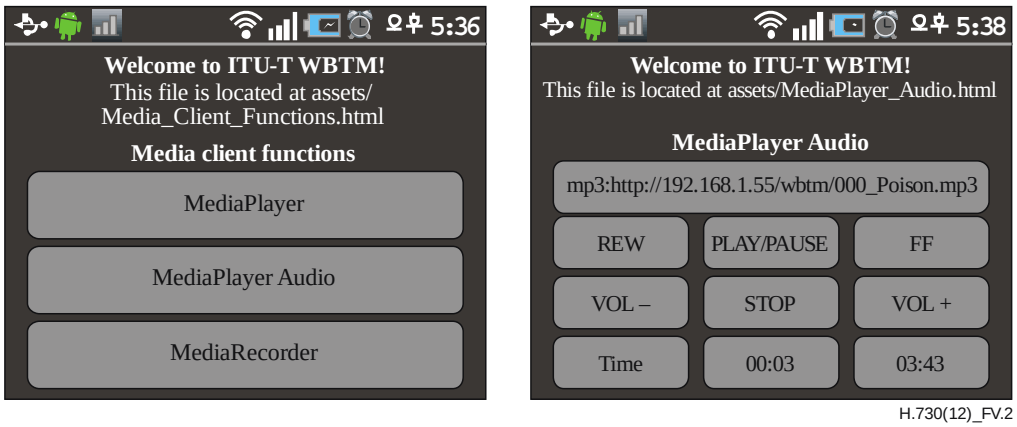

Figure V.2 – Display of media clients function interface

Figure V.3 illustrates the IDL sample for the implementation of the above functions.

```
function play()       //AudioPlayer play
{
     MediaPlayer.play();
}

function pause()          //Pauses playback
{
     MediaPlayer.pause();
}

function stop()        //AudioPlayer stop
{
     MediaPlayer.stop();
}

function start()        //AudioPlayer stop
{
     AudioPlayer.audioPlay();
     duration();
}

function stop()     //AudioPlayer stop
{
     on = 1;
     AudioPlayer.audioStop();
}

function FF()       //AudioPlayer fast-foward (5 seconds)
{
     AudioPlayer.audioFF();
}

function REW()      //Audioplayer rewand (5 seconds)
{
     AudioPlayer.audioREW();
}

function volumeUp( )      //Sets Audio Volume up
```

```
{
     AudioPlayer.audioVolumeUp();
}

function volumeDown()    //Sets Audio Volume down
{
     AudioPlayer.audioVolumeDown();
}

function audioPath(arg)     //the audiopath of the file
{
     AudioPlayer.audioPath(arg);
}

function gerDuration()        //Get the duration of the file
{
     time = string(AudioPlayer.audioDuration());
     }

function getCurrentPosition()     //Get the current playback position
{
     time = string(AudioPlayer.audioCurrentPosition());
     }
```

Figure V.3 – IDL for the example in Appendix III

Appendix VI

An example of the interface description language for WBTM APIs

(This appendix does not form an integral part of this Recommendation.)

The interface description language (IDL) description in this appendix describes the interfaces of the WBTM APIs; this is provided for a better understanding of WBTM.

VI.1 Interfaces

VI.1.1 IPTV terminal transport functions interfaces

VI.1.1.1 Network attachment client interface

DiscoveryFinder: Find one or more terminal device.

```
interface DiscoveryFinder
{
     public DiscoveryServiceCollection find(in string strName,  in int
nTimeoutInMS); // Find a device of special name in local network, with in time
out
     public DiscoveryServiceCollection find(in int nTimeoutInMS); // Find all
device in local network, with in time out
     public void done();// Stop finding
};
```

VI.1.2 Content delivery client functions interface

VI.1.2.1 Broadcast demux interface

VI.1.2.1.1 Broadcast

```
interface Broadcast
{
     // Broadcasting Type define
     const long BROADCAST_TYPE_TERRERRESTRIAL  = 0; // Non IP - TERRERRESTRIAL
     const long BROADCAST_TYPE_CABLE           = 1; // Non IP - CABLE
     const long BROADCAST_TYPE_SATELLITE       = 2; // Non IP - SATELLITE
     const long BROADCAST_TYPE_UNICAST         = 3; // Over IP - UNICAST
     const long BROADCAST_TYPE_MULTICAST       = 4; // Over IP - MULTICAST
     boolean OpenBroadCast();  // Open and Connect to Broadcasting
     void CloseBroadCast();  // Close or disconnect from Broadcasting
     void setBroadCastType(in long nType); // Set Broadcasting type
     void getBroadCastType();  // Get Broadcasting type
     void ChannelUp();  // Channel Up
     void ChannelDown();// Channel Down
     void setChannel(in ChannelInfo sToChannel); // Set Channel by Channel
information
     void setChannel(in string strChNum); // Set Channel by Channel number
     void SetDefaultChannel();// Set to default channel
     ChannelInfo getCurrentChannel();// Get current channel information
     string getCurrentChannelName();// Get current channel Name
     boolean StartChannelScan();// Channel scanning start
     void StopChannelScan();// Channel scanning start
     int getChannelCount();// Get available channels count
};
```

VI.1.2.1.2 Channel manager

```
interface ChannelManager
{
     attribute  Vector       m_vecChannels;    // Channel vector
     attribute  int     m_ChannelIndex;   // Current channel index
     int AddChannel(in ChannelInfo sCh); // Add Channel information to channel
vector - return channel count
     int getChannelCount(in ChannelInfo sCh); // Get Channel count of channel
vector
     boolean RemoveChannel(in string nChannelNum); // Remove channel information
from channel vector
     ChannelInfo FindChannel(in string nChannelNum); //Find Channel Information
via channel number
     boolean StartChannelScan();//Channel Scan Start
     boolean StopChannelScan();//Channel Scan Stop
     boolean ClearChannel();//Clear channel information from channel vector
     ChannelInfo ChannelUp();//Channel up
     ChannelInfo ChannelDown();//Channel Down
     ChannelInfo getCurrentChannel();//Get current channel information
     ChannelInfo  SetDefaultChannel();//set to default channel
};
```

VI.1.2.1.3 Channel information

```
interface ChannelInfo
{
     attribute  string    mstrChannelName;       // Channel Name
     attribute  string    mstrURL;                    // Channel attribute
Service URL
     attribute  string     mstrChannelNo;       // Channel Number
     attribute   int         mnChannelFreq;     // Channel Frequency
     attribute   int     mnChannelMajorCh;      // Major Channel Number
     attribute   int     mnChannelMinorCh;      // Minor Channel Number

     int getChannelFreq(); //Get Channel frequency
     void setChannelFreq(in long mnChannelFreq);  // Set Channel Frequency
     int getChannelMajorCh(); // Get Major Channel Number
     void setChannelMajorCh(in long mnChannelMajorCh); // Set Major Channel
Number
     int getChannelMinorCh() ; // Get Minor Channel Number
     void setChannelMinorCh(in long mnChannelMinorCh) ; // Set Minor Channel
Number
     string getChannelName();   // Get Channel Name
     void setChannelName(string mstrChannelName) ; // Set Channel Name
     string getURL() ; // Get URL
     void setURL(string mstrURL) ; // Set URL
     string getChannelNo();    // Get Channel Num,ber
     void getChannelNo (string chNum) ; // Set Channel Number
}
```

VI.1.2.2 MulticastClient – Multicast content delivery client functional interface

```
interface MulticastClient
{
     attribute MulticastSocket   mSocket;         // MulticastSocket
     attribute int              mPort ;      // Port for Multicast
     attribute string           mServerAddress ; // Server IP Address
     attribute  InetAddress         mAddress;     // IP Address container
     attribute boolean          bJoin;        // Is Joined to multicast server
     void joinGroup();  //Join to Mulicast group
     void leaveGroup();  //Leave Mulicast group
```

```
    void leaveGroup();  // Leave Mulicast group
    DatagramPacket getDataPacket(in long nDataLen);   // Get DataPaket
    boolean isJoinned();  // Is Joinned to Multicast
};
```

VI.1.2.3 Unicast content delivery client functional block interface

This is a normal media player's function. A media stream or file can be played.

See clause VI.1.3 for the media client function.

VI.1.3 Media client functions interfaces

VI.1.3.1 MediaPlayer

MediaPlayer can be used to control the playback of audio/video files and streams.

```
interface MediaPlayer
{
    void getCurrentPosition();  //Gets the current playback position
    void getDuration();  //Get the duration of the file
    long getVideoHeight();  //Returns the height of the video
    long getVideoWidth();  //Returns the width of the video
    boolean isLooping();  //Checks whether the MediaPlayer is looping or non-
looping
    boolean isPlaying();  //Checks whether the MediaPlayer is playing
    void mediaPause();  //Pauses playback
    void seekTo(in long msec);  //Seeks to specified time position
    void setLooping(in boolean looping);  //Sets the MediaPlayer to be loopint
or non-looping
    void mediaVolumeUp();  //Sets the MediaVolume 1 step up
    void mediaVolumeDown();  //Sets the MediaVolume 1 step dwon
    void mediaPlay();  //MediaPlayer play
    void mediaStop();  //MediaPlayer stop
    void mediaFF();  //MediaPlayer fast-foward (5 seconds)
    void mediaREW();  //Mediaplayer rewand (5 seconds)
    void MediaPath(in string path);   //the Mediapath of the file, or the
http/rtsp URL of the stream you want to play
};
```

VI.1.3.2 Subtitle

```
interface SubTitle
{
    boolean      mbVisible = true;  //visible subtitles
    void subtitleVisible(in boolean visible); //true is visible. false is
invisible
    void subtitleFontSize(in long fontSize); //Subtitles font size
    void subtitleLocation(in long X, in long Y); //Subtitles locate position
    void subtitleDefaultPosition();  //Subtitles default position
    void subtileFF(in long time);  //movement synchronization subtitles Fase-
Forward
    void subtileREW(in long time);  //Movement synchoronization subtitles
Rewind
};
```

VI.1.3.3 MediaInfo

```
interface MediaInfo
{
     long long fileInfoDate_Created();  //The time the Media was Created to the
media provider Units are seconds since 1970
     long long fileInfoDate_Modified();  //The time the Media was last modified
Units are seconds sincd 1970
     string fileInfoDisplay_Name();  //The display name of this file
     long long getSize();  //The size of the file in bytes
     string getTitle();  //The title of the content
     string getArtist();  //Audio file info Artist
     string getAlbum();  //Audio file info Album
     string getFileName();  //Audio file info filename
     sCodecInfo videocodecInfo();  //Return video codec info
     sCodecInfo audiocodecInfo();  //Return audio codec info
};
```

VI.1.3.4 AudioPlayer

AudioPlayer class can be used to control playback of audio files and streams

```
interface AudioPlayer
{
     void getCurrentPosition();  //Gets the current playback position
     void getDuration();  //Get the duration of the file
     boolean isLooping();  //Checks whether the AudioPlayer is looping or non-
looping
     boolean isPlaying();  //Checks whether the AudioPlayer is playing
     void Pause();  //Pauses playback
     void seekTo(in long msec);  //Seeks to specified time position
     void setLooping(in boolean looping);  //Sets the player to be loopint or
non-looping
     void VolumeUp();  //Sets Audio Volume   up
     void VolumeDown();  //Sets Audio Volume   dwon
     void Play();  //AudioPlayer play
     void Stop();  //AudioPlayer stop
     void FF();  //AudioPlayer fast-foward (5 seconds)
     void REW();  //Audioplayer rewand (5 seconds)
     void audioPath(in string path);   //the audiopath of the file, or the
http/rtsp URL of the stream you want to play
};
```

VI.1.3.5 MediaRecorder

Used to record audio and video. The recording control is based on a simple state machine.

```
interface MediaRecoder
{
     void getRecodertime();  //Gets the Recodertime
     void getDuration();  //Get the duration of the file
     long getVideoHeight();  //Returns the height of the video
     long getVideoWidth();  //Returns the width of the video
     boolean isRecording();  //Checks whether the MediaRecorder is recording
     void pauseRecorder();  //Pauses playback
     void stopRecorder();  //MediaRecorder stop
     void playRecorder(in string file, in long type);  //Default MediaRecorder
play
     void playAudioRecorder(in string file, in long audioSource, in long
outputFormat, in long audioEncoder, in long maxDuration, in long maxFileSize);
//Detail setting for          AudioRecording by user
     void playMediaRecorder(in string file, in long audioSource, in long
```

```
videoSource, in long outputFormat, in long audioEncoder, in long vidoeEncoder,
in long width, in long height, in long videoFrameRate, in long maxDuration, in
long maxFileSize);  //Detail setting for MediaRecording by user
     // Defines the audio source. These constants are used with
setAudioSource(int)
     const long AudioSource.DEFAULT           = 0;
     const long AudioSource.MIC               = 1;
     const long AudioSource.VOICE_UPLINK      = 2;
     const long AudioSource.VOICE_DOWNLINK    = 3;
     const long AudioSource.VOICE_CALL        = 4;
     const long AudioSource.CAMCORDER         = 5;
     const long AudioSource.VOICE_RECOGNITION = 6;
     //Defines the video source. These constants are used with
setVideoSource(int)
     const long VideoSource.DEFAULT = 0;
     const long VideoSource.CAMERA 1 = 1;
     const long VideoSource.CAMERA 2 = 2;
     //Defines the output format. These constants are used with
setOutputFormat(int)
     const long OutputFormat.DEFAULT   = 0;
     const long OutputFormat.THREE_GPP = 1;
     const long OutputFormat.MPEG_4    = 2;
     const long OutputFormat.RAW_AMR   = 3;
     //Defines the audio encoding. These constants are used with
setAudioEncoder(int)
     const long AudioEncoder.DEFAULT = 0;
     const long AudioEncoder.AMR_NB  = 1;
     //Defines the video encoding. These constants are used with
setVideoEncoder(int)
     const long VideoEncoder.DEFAULT   = 0;
     const long VideoEncoder.H263      = 1;
     const long VideoEncoder.H264      = 2;
     const long VideoEncoder.MPEG_4_SP = 3;
};
```

VI.1.4 SCP client functions interfaces

VI.1.4.1 DRM function interface

```
interface DRM
{
     struct  DRMInfo{
          char SID[64];       //sellerID
          char SCID[128];  //SellerContentID
          char UID[64];       //UserID
          char UPW[64];     //Password
     };
     void setDRMInfo(in string SID, in string SCID, in string UDI, in string
UPW);
};
```

VI.1.4.2 CAS function interface

```
interface CAS
{
     long authorityCheck(in long CW); // decoding by CW
};
```

VI.1.5 Application client functions interface

VI.1.5.1 appExecuter – IPTV application client interface

```
interface appExecuter
{
     boolean isRunning(in string fileName); //Check isRunning
     void execute(in string filePath, in string fileName);  //Execute program
     void terminate(in string fileName);   //Kill the executed program
};
```

VI.1.5.1.1 Browser – Global function for WBTM client

```
interface Browser
{
     void navigate(in string strURL); // Go to the URL web site
     void closeApplication();  //Close the browser or exit web application.
     void goBack();  //Go back to history of URL stack.
     boolean canGoBack();  //Return true if this WebView has a back history
item.
     void zoomIn();  //Perform zoom in in the Browser
     void zoomOut();  //Perform zoom out in the Browser
     void  goBackOrForward(in long steps);  //Go to the history item that is the
number of steps away from the current item.
     void  goForward();  //Go forward in the history of this Browser.
     void setVerticalScrollBarEnabled(in boolean b);  //Specify whether the
vertical scrollbar has overlay style.
     string getUrl();  //Get the url for the current page
     boolean  canGoBackOrForward(in long steps);  //Return true if the page can
go back or forward the given number of steps.
     boolean  canGoForward();  //Return true if this Browser has a forward
history item.
     void  clearCache(in boolean includeDiskFiles); //Clear the resource cache.
     void  clearFormData();  //Make sure that clearing the form data removes the
adapter from the currently focused textfield if there is one.
     void  clearHistory();  //Tell the Browser to clear its internal
back/forward list.
     string  getOriginalUrl();  //Get the original url for the current page.
     int  getProgress();  //Get the progress for the current page.
     float  getScale();  //Return the current scale of the Browser
     string  getTitle();  //Get the title for the current page.
     void  stopLoading();  //Stop the current load.
     void  reload();  //Reload the current url.
};
```

VI.1.5.1.2 WebSettings

```
interface WebSettings
{
     enum ZoomDensity { FAR,          // 240dpi
                                      MEDIUM,    // 160dpi
                                      CLOSE};      // 120dpi}
     int getDefaultFontSize();  //Get the default font size.
     boolean  getAllowFileAccess();  //Returns true if this Browser supports
file access.
     int  getCacheMode();  // Return the current setting for overriding the
cache mode
     string  getCursiveFontFamily();  // Get the cursive font family name
     boolean  getDatabaseEnabled();  // Returns true if database storage API is
enabled
     string  getDatabasePath();  // Return the path to where database storage
API databases are saved for the current Browser
     int  getDefaultFixedFontSize();  // Get the default fixed font size. The
default is 16.
     string  getDefaultTextEncodingName();  // Get the default text encoding
name
     WebSettings.ZoomDensity  getDefaultZoom();  // Get the default zoom density
of the page
     boolean  getJavaScriptEnabled();  // Return true if javascript is enabled.
The default is false.
     void  setAppCacheEnabled(in boolean flag);   //Tell the Browser to enable
Application Caches API.
     void  setCacheMode(in long mode);   //Override the way the cache is used.
The way the cache is used is based on the navigation option
     void  setCursiveFontFamily(in string font);  //Set the cursive font family
name
     void  setDatabaseEnabled(in boolean flag);  //Set whether the database
storage API is enabled
     void  setDefaultZoom(in WebSettings.ZoomDensity zoom);  //Set the default
zoom density of the page
     void  setDatabasePath(in string databasePath);  //Set the path to where
database storage API databases to be saved
     void  setDefaultFixedFontSize(in long size);  //Set the default fixed font
size.
     void  setDefaultFontSize(in long size);  //Set the default font size
     void  setJavaScriptEnabled(in boolean flag);   //Tell the Browser to enable
javascript execution.
     void  setStandardFontFamily(in string font);   //Set the standard font
family name.
     void  setSupportZoom(in boolean support);   //Set whether the Browser
supports zoom
     boolean  supportZoom();  //Returns whether the Browser supports zoom
};
```

VI.1.5.1.3 URLUtil

```
interface URLUtil
{
     string  decode(in string strUrl, in string encode); //Decode url
     string  guessUrl(in string inUrl); //Cleans up (if possible) user-entered
web addresses
     boolean  isAboutUrl(in string url);  //True iff the url is an about: url.
     boolean  isAssetUrl(in string url);  //True iff the url is an asset file.
     boolean  isContentUrl(in string url);  //True iff the url is a content:
url.
     boolean  isDataUrl(in string url);  //True iff the url is a data: url.
     boolean  isFileUrl(in string url);  //True iff the url is a local file.
     boolean  isHttpUrl(in string url);  //True iff the url is an http: url.
```

```
    boolean  isHttpsUrl(in string url);  //True iff the url is an https: url.
    boolean  isJavaScriptUrl(in string url);  //True iff the url is a
javascript: url.
    boolean  isNetworkUrl(in string url);  //True iff the url is a network url.
    boolean  isValidUrl(in string url);  //True iff the url is valid.
    string  stripAnchor(in string url);  //Strips the url of the anchor.
};
```

VI.1.5.2 SADS client functional block interface

VI.1.5.2.1 Starter

```
interface Starter
{
    void getInstance();  // Gets this sigleton instance of the Service starter
    void goHome();  // Browser to HomeScreen of  WBTM. You can choose  one of
the Contents provider and service category.
};
```

VI.1.6 Connection and session management interface

VI.1.6.1 CookieManager

```
interface CookieManager
{
    boolean acceptCookie () ;  //Return whether cookie is enabled
    string getCookie(in string url);  //Get cookie(s) for a given url so that
it can be set to "cookie:" in http request header.
    boolean hasCookies ();  //Return true if there are stored cookies.
    void removeAllCookie ();  //Remove all cookies
    void removeExpiredCookie ();  //Remove all expired cookies
    void removeSessionCookie ();  //Remove all session cookies, which are
cookies without expiration date
    void setAcceptCookie (in boolean accept);   //Control whether cookie is
enabled or disabled
    void setCookie (in string url, in string value);   //Set cookie for a given
url.
};
```

VI.1.6.2 CookieSyncManager

```
interface  CookieSyncManager
{
    void resetSync () ;  // resets sync manager's timer
    void run () ;  //Starts executing the active part of the class' code.
    void startSync () ;  //startSync() requests sync manager to start sync
    void stopSync () ;  //stopSync() requests sync manager to stop sync
    void sync () ;  //sync() forces sync manager to sync now
};
```

VI.1.7 Terminal device management interface

VI.1.7.1 NetworkSetting

```
interface NetworkSetting
{
     attribute boolean     mbDHCP;  // is DHCP or Not
     attribute string      mstrIPAddress;  // Local Ip Address
     attribute string      mstrSubnetMask;  // Subnet Mask
     attribute string      mstrGetWay;  //Gateway IP
     attribute boolean     mbAutoDNS;  // Get DNS Address automaticaly
     attribute string      mstrPrimDNS;  // Primary DNS
     attribute string      mstrSecDNS;  // Secondary DNS
     boolean IsDHCP();  // Get Information about DHCP or Not
     void SetDHCP(in boolean bdhcp);   // Set DHCP or Not
     void setIPAddress (in string strIP);  // Set IP Address
     string getIPAddress ();  // Get IP Address
     void setSubnetMask(in string strIP);  // Set Subnet Mask
     string getSubnetMask ();  // Get Subnet Mask
     void setGateway(in string strIP);  // Set Gateway
     string getGateway();  // Get Gateway Address
};
```

VI.1.7.2 StorageManager

This class represents the storage manager which keeps track of the storage devices attached to the system.

```
interface StorageManager
{
     const long STORAGETYPE_HD  = 0;
     const long STORAGETYPE_USB = 1;
     long getStorageType();//Return StorageType
     void onRemove(in long storageType);  //Remove Storage
     void onAttach(in long storageType);  //Attach Storage
     const long STORAGE_ADDED   = 0;
     const long STORAGE_REMOVED = 1;
     const long STORAGE_CHANGED = 2;
     long addStorageManagerListener(in StorageManagerListener listener);
//Return STORAGE_ADDED
     long removeStorageManagerListener(in StorageManagerListener listener);
//Return STORAGE_REMOVED
     sequence<string> getMountList();  //Return MountList
     octet getTotalVolume();  //Return TotalVolume
     octet getFreeVolume();  //Return FreeVolume
     octet getUsedVolume();  //Return UsedVolume
     const long FAT16 = 0;
     const long FAT32 = 1;
     const long NTFS  = 2;
     const long EXT2  = 3;
     const long EXT3  = 4;
     const long RAISERFS = 5;
     long getFileSystem(in long storageType);  //Return storage filesystem
};
```

VI.1.7.3 MicController

```
interface MicController
{
     const long RET_OK = 0;
     const long RET_NOT_SUPPORTED = 1;
     const long RET_FAIL = 2;
     void instance();  //Create MikeController object.
     void setVolume(in long level);  //Set microphone volume to level.
     void getVolume();  //Get current microphone volume.
     void setMikeEnabled(in boolean enable);  //Set microphone Enable or
disable.
     void isMikeEnabled();  //Check is microphone enabled.
     void MikeMute();  //Check whether themicrophone mute is on or off.
     void setMikeMute(in boolean on);  //Sets the microphone mute on or off
};
```

VI.1.7.4 VolumeController

```
interface VolumeController
{
     void getInstance();  //Create VolumeController object.
     void setVolumeLevel(in long value);  //Set speaker volume to the level.
     void setMute(in boolean isMute);  //Set Mute
     boolean isMute();  //Check is muted
     void getCurrentVolumeLevel();  //Get current volume level.
     void systemVolumeUp();  //Set SystemVolume 1 step up.
     void systemVolumeDown();  //Set SystemVolume 1 step down.
     void mediaVolumeUp();     //set MediaVolume 1 step up.
     void mediaVolumedown();  //set MediaVolume 1 step down
     void getMaxVolumeLevel();  //Get maximum volume leavel.
     void getMinVolumeLevel();  //Get minimum volume level.
};
```

VI.1.7.5 PowerManager

```
interface PowerManager
{
     void   goToSleep(in long long time);  //Force the device to go to sleep.
Overrides all the wake locks that are held.
     void  isScreenOn();  //Returns whether the screen is currently on. The
screen could be bright or dim.
     void reboot();  //Reboot the device. Will not return if the reboot is
successful.
};
```

VI.1.7.6 System Clock

```
interface SystemClock
{
     long long currentThreadTimeMillis(in boolean realTime);  //Returns
milliseconds running in the current thread.
     long long elapsedRealtime();  //Returns milliseconds since boot, including
time spent in sleep.
     boolean setCurrentTimeMillis(in long long millis);  //Sets the current wall
time, in milliseconds. Requires the calling process to have appropriate
permissions.
     void sleep(in long long ms);  //Waits a given number of milliseconds(of
uptimeMillis) before returning.
};
```

VI.1.8 Performance monitoring interface

VI.1.8.1 MemoryInfo

```
interface MemoryInfo
{
     long long totalMemory();  //Returns the total amount of memory which is
available to the running program.
     long long freeMemory();   //Returns the amount of free memory resources
which are available to the running program.
     long long maxMemory();  //Returns the maximum amount of memory that may be
used by the virtual machine, or Long.MAX_VALUE if there is no such limit
     long long usedMemory();  //Returns the amount of Usedmemory resources which
are available to the running program.
};
```

VI.1.8.2 ProcessorInfo

```
interface ProcessorInfo
{
     struct Processor
       {
          string mName;
          long   mID;
       };
     long getTotalProcessor();  // Count of All Processor
     int  CPUUsageRate();  // Get CPU Usage Rate
     int    ProcessorRate(in long ProcessorID);  // Get Processor's Usage Rate
     void   KillProcessor(in long ProcessorID);  // Kill Processor
     void   KillAllProcessor(in string strProcessorName);  // Kill Processor vis
Processor name
     typedef sequence<Processor> GetWorkingProcessorList();  //Get Progessor
List
};
```

VI.2 Web-based engine structure

VI.2.1 Markup language

VI.2.1.1 MLDocument

```
interface MLDocument
{
     void load(string URL);  // load url
     boolean isLoading();  //Check whether current page is loading
     string getURL();  // Get current url
     boolean isComplete();// Check whether current page loads complete
     void stopLoad();  //stop loading page
     boolean isStopping();  //Check whether current page stop loading
     ResourceError getError();  //Get Loading error information
};
```

VI.2.1.2 ResourceError

```
interface ResourceError
{
     string domain();  //get the domain
     int errorCode();  //get errordode
     string localizedDescription();  //get error code descriptione
};
```

VI.2.2 Document access interface

VI.2.2.1 DOM node

```
interface DOMNode
{
     string getNodeName(string strNode); //get node name
     void setNodeValue(string strValue);  //setting node value of current node
     string nodeValue(string strNode);  //get node value
     string nodeType(string strNode);  //get node type
     DOMNode parentNode(string strNode);  //get parent node
     DOMNodeList childNodes();  //get child node list of current node
     DOMNode firstChild();  //get first child node of current node
     DOMNode lastChild();  //get last child node of current node
     DOMNode previousSibling();  // get the previous sibling node
     DOMNode nextSibling();  //get  the next sibling node
     boolean removeChild (DOMNode oldChild);  //remove child node of current
node
     boolean appendChild( DOMNode  newChild)  //append child node of current
node
     boolean hasChildNodes();  //return true if node has children
     boolean hasAttributes();  //return ture if node has attributes
};
```

VI.2.2.2 DOM node list

```
interface  DOMNodeList
{
     DOMNode item(UINT index);  //get DOMNode of DOMNodeList
     int length();  //get length of DOMNodeList
};
```

VI.2.2.3 DOM document

```
interface DOMDocument
{
     void Load(string strFile);  // load DOM file
     DOMNodeList getElementsByTagName(BSTR tagName);
          // returns a NodeList of all a elements with a specified name
     DOMElement getElementById(BSTR elementId);
          //returns  the first element with the specified id
};
```

VI.2.2.4 DOM element

```
interface DOMElement
{
     string getAttribute(string  name);  // gets an attribute value by name
     void setAttribute(string name, string value);  // adds a new attribute.
     void removeAttribute(string name);  // removes a specified attribute
};
```

VI.2.3 Document style

VI.2.3.1 Style sheet

```
interface StyleSheet
{
     CSSRule ownerRule();  //get Parent(Owner)'s  Rule
     CSSRuleList* rules();  //get  rulelist
     int  addRule(string selector, string style, int index);  //Add rule to
style sheet
     int addRule(string selector, string style);  //Add rule to style sheet
     void removeRule(unsigned index);  //remove rule from style sheet
     boolean isLoading();  //Check whether style sheet is loading
     boolean isComplete() ;  // Check whether style sheet loads complete
};
```

VI.2.3.2 Rule

```
interface Rule
{
     boolean isStyleSheet();  //check  sytlesheet
     boolean isCSSStyleSheet();  //check CSS style sheet
     boolean isXSLStyleSheet();  //check XSL style sheet
     boolean isStyleSheetList() ;  //check style sheet list
     boolean isMediaList();  // check media list
     boolean isMediaRule();  // check media rule
     boolean isRuleList();  // check rulelist
     boolean isRule();  // check rule
     boolean isStyleRule();  // check style rule
     boolean isCharsetRule();  // check charset rule
     boolean isImportRule();  //check imported rule
     boolean isFontFaceRule();  // check fontface rule
     boolean isPageRule();  // check page rule
     boolean isUnknownRule();  // check unknown rule
     boolean isValue();  // check  a value
     string getValue();  //get  value
};
```

VI.2.3.3 Rule list

```
interface RuleList
{
     int length() ;  //get  item length
     Rule item(unsigned index);  //get rule item of list
     int insertRule(Rule*, unsigned index);  //insert a new rule
     void deleteRule(unsigned index);  // delete rule
     void append(CSSRule );  // append a new rule
};
```

VI.2.4 Scripting language

VI.2.4.1 Script

```
Script
{
     JSValue executeScript(string script, boolean forceUserGesture);
          // execute script string
};
```

VI.2.4.2 Script value

```
ScriptValue
{
     enum JSType
     {
          UnspecifiedType   = 0;
          NumberType        = 1;
          BooleanType       = 2;
          UndefinedType     = 3;
          NullType          = 4;
          StringType        = 5;
     };
     boolean   isUndefined();  //check  undefined type
     boolean isNull();  //check  null type
     boolean isBoolean();  //check boolean
     boolean isNumber();  // check number
     boolean isString();  // check string
     boolean getBoolean();  //get boolean value
     double getNumber();  //get number value
     string getString();  //get string value
};
```

Bibliography

[b-ITU-T H.741.0] Recommendation ITU-T H.741.0 (2011), *IPTV application event handling: Overall aspects of audience measurement for IPTV services*.

[b-ITU-T H.750] Recommendation ITU-T H.750 (2008), *High-level specification of metadata for IPTV services*.

[b-ITU-T Y.101] Recommendation ITU-T Y.101 (2000), *Global Information Infrastructure terminology: Terms and definitions*.

[b-ITU-T Y.110] Recommendation ITU-T Y.110 (1998), *Global Information Infrastructure principles and framework architecture*.

[b-W3C WebArch] W3C Recommendation (2004), *Architecture of the World Wide Web, Volume One*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2013