



# Curso Multimedia Home Platform 1.1.2

## MHP PLUG-INS

Qué son, por qué existen

Tipos

Ciclo de vida.API

## Curso Multimedia Home Platform 1.1.2

Copyright 2008 © Enrique Pérez Gil

Licensed under the ***Creative Commons Attribution-Non-Commercial-No Derivative Works 3.0 Unported License***. You may not use this file except in compliance with the License. You may obtain a copy of the License at:

<http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>

This is a human-readable summary of the License applied:

(<http://creativecommons.org/licenses/by-nc-nd/3.0/>)

**You are free to Share**, to copy, distribute and transmit the work **Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.

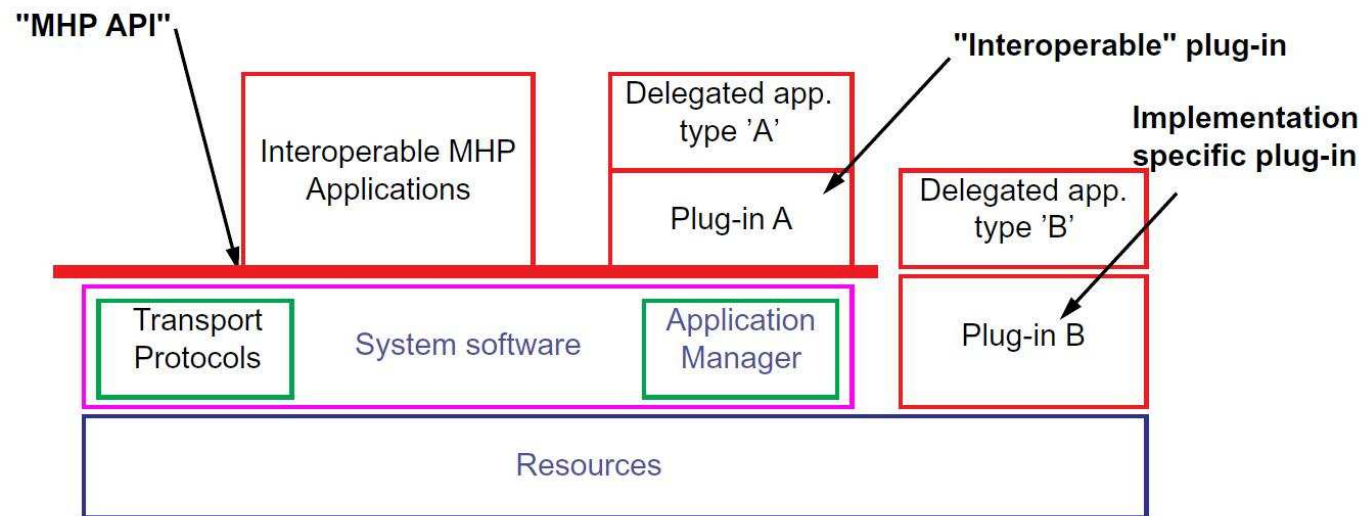
## Introducción

- Al igual que en otros contextos más conocidos (File type managements), MHP ofrece la posibilidad de incluir dentro de una máquina MHP genérica, bloques de software que saben manejar **formatos de aplicaciones** que no son cubiertos por la especificación MHP.
  - **Atención:** hemos dicho Aplicaciones! No vale para entender MPEG-4, por ejemplo.
- Aquellos que deseen implementar este tipo de plug-ins han de especificar muy claramente dichos formatos, de forma que se pueda garantizar la interoperabilidad entre plataformas MHP y otro tipo de estas (si es que aplica). Tanto es así que estos TIPOS han de ser registrados en DVB de manera que se garantice su consistencia en todo el contexto MHP.
- Un ejemplo de Plug-in es el que desarrolló la empresa Espial, para ofrecer la capacidad de representar dentro de un contexto de ejecución MHP aplicaciones de otra naturaleza.
  - <http://www.embeddedstar.com/press/content/2003/12/embedded11871.html>

## Introducción

- Existen dos tipos de plug-in dependiendo del entorno en el que se ejecuten:
  - **Tipo A:** Como una Aplicación MHP, lo cual garantiza portabilidad, por eso también se denominan “**Interoperables**”
  - **Tipo B:** **No es una MHP APP**, se encuentra en la capa del middleware. Usa código específico de la implementación MHP, es decir, no es interoperable, y no garantiza portabilidad.

Nos centraremos en los Interoperables.



MHP 1.1.2, A0068r1

Figure 7: Illustrative plug-in implementation options

## Plug-ins Interoperables

- Un plug-in se proporciona como una aplicación normal, lo único que lo diferencia es que en el **inner loop** de la **AIT** tendremos **al menos un descriptor** que nos indica que esta APP es un plug-in. **OJO:** puede haber varios y por tanto entender más de un tipo de aplicaciones!!
- **plugin\_application\_descriptor**, descriptor\_tag = **0x0F**
- El descriptor nos ofrece dos datos fundamentales: el tipo de aplicaciones que “entiende”, **application\_type**, y qué configuraciones/versiones de este tipo puede manejar, **el bucle**.
- El mecanismo de perfiles/versiones es el mismo que en apps normales MHP, con sus propios datos, claro está.

	No.of Bits	Identifier
plugin_application_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
application_type	16	uimsbf
for( i=0; i<N; i++ ){		
application_profile	16	uimsbf
version.major	8	uimsbf
version.minor	8	uimsbf
version.micro	8	uimsbf
}		
}		

## Plug-ins Interoperables

- ¿ Y como se proporciona la App del tipo específico que entiende el Plug-in ? Esta se va a proporcionar de igual forma que una normal salvo por el hecho de que su **application\_type** será el del **Plug-in**.
- Podría darse la siguiente circunstancia: más de un plug-in soporta el mismo tipo de aplicación. En este caso una App puede incluir en su inner loop un descriptor (sólo uno) del tipo **delegated\_application\_descriptor** que le va a permitir indicar al STB qué plug-ins habrá de intentar usar, y qué orden, para ejecutarla. Si no viene el descriptor o los indicados no están, el deco decidirá con sus propios criterios.
- **delegated\_application\_descriptor**, descriptor\_tag = **0x0E**
- Como se ve disponemos de un bucle con los posibles “plug-ins” identificados.

MHP 1.1.2, A0068r1

	No.of Bits	Identifier
delegated_application_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for( i=0; i<N; i++ ){		
application_identifier	48	uimsbf
}		
}		

## Plug-ins Interoperables

- ¿ Y cómo se ejecuta el nuevo tipo de aplicación ? El Plug-in deberá implementar el interface **org.dvb.application.plugins.Plugin**, que sirve para que el middle le pase la referencia a la App, o mejor dicho a sus datos, y la ejecute. Veamos cómo:
  - public boolean **isSupported**(AppAttributes app);
    - Método siempre disponible.
    - Para saber si la soporta

### Paso 1

- public boolean initPlugin();
  - Siempre antes de llamar a initApplication... Deberá efectuar las operaciones de inicialización que necesite.
  - Por cierto, un plug-in **siempre habrá de tener un constructor por defecto!!!**

## Plug-ins Interoperables

### Paso 2

- `public Xlet initApplication(AppAttributes app)`
- `public javax.tv.xlet.Xlet initApplication(InnerApplication app)`

Solicita que le ofrezca el Xlet a través del cual se ejecutará la App. Si no puede devuelve null.

**OJO:** las APPs habrán de adaptarse el ciclo de vida de los Xlets.

### Paso 3

- `public void terminatePlugin();`
  - Se llamará cuando todas las Apps generadas por el Plug-in hayan sido terminated. Para Liberar recursos.

## Plug-ins Interoperables

- Cuando el método `initApplication` devuelve el `Xlet`, este será usado como si fuese la APP, es decir, **recibirá sus parámetros, tendrá acceso a estructuras de ficheros...etc.**
- Cuando esto sucede es cuando internamente se podrá acceder a los “ficheros” que forman parte de esta App, y que por ejemplo vienen indicados como parámetros de la misma, de manera que se puedan interpretar y “ejecutar”....aplicando el ciclo de vida `Xlet!!!`

<b>ISO/IEC 13818-1</b>	Part 1. Elementary Streams transport definition
<b>ISO/IEC 13818-6</b>	Part 6. Extensions for DSM-CC. Digital Storage Media Command and Control
<b>ETSI EN 300 468</b>	Digital Video Broadcasting (DVB);Specification for Service Information (SI) in DVB systems
<b>ETSI EN 301 192</b>	DVB specification for data broadcasting
<b>ETSI TR 101 202</b>	Implementation Guidelines for Data broadcasting
<b>ETSI TR 101 162</b>	Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems
<b>ETSI TR 102 154</b>	Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in Contribution and Primary Dist
<b>ETSI TR 101 211</b>	Guidelines on implementation and usage of Service Information (SI)
<b>ETSI TR 101 200</b>	Digital Video Broadcasting (DVB); A guideline for the use of DVB specifications and standards
<b>DAVIC</b>	Digital Audio Visual Council. davic 1.4.1
<b>HAVI</b>	Specification of the Home Audio/Video Interoperability (HAVi) Architecture
<b>Interactivetvweb</b>	<a href="http://www.interactivetvweb.org/">http://www.interactivetvweb.org/</a>
<b>Wikipedia DSMCC</b>	<a href="http://en.wikipedia.org/wiki/DSM-CC">http://en.wikipedia.org/wiki/DSM-CC</a>
<b>MHP 1.1.2</b>	Multimedia Home Platform, A068r1 & tam668r23_11xdraft_20061115
<b>MHP 1.1.3</b>	Multimedia Home Platform, A068r3
<b>CDC 1.1</b>	Connected Device Configuration (CDC) 1.1 (JSR=218).
<b>PBP 1.1</b>	Personal Basis Profile 1.1 (JSR 217)
<b>MHP.org</b>	<a href="http://www.mhp.org">www.mhp.org</a>
<b>INTRO MHP 1.1.3</b>	tam1032r1-mhp-iptv-presentation