



Curso Multimedia Home Platform 1.1.2

MHP Xlet Management

¿ qué es un xlet ?

Ciclo de Vida

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.

Qué es un Xlet

Clase java con constructor público vacío que implementa el interface javax.tv.xlet.Xlet

```
public interface Xlet {
```

```
    public void initXlet(XletContext ctx) throws XletStateChangeException;
```

- Se llama una vez. Me pasan el contexto. Si no puedo lanzo exception y pasará a destroyed, si no, paso a paused

```
    public void startXlet() throws XletStateChangeException;
```

- Estaba en paused y me dicen que pase a Active. Si en ese momento no puedo lanzo Exception y permanezco paused

```
    public void pauseXlet();
```

- Me notifican que me han pausado

```
    public void destroyXlet(boolean unconditional) throws XletStateChangeException
```

- Me dicen que estoy destroyed salvo que unconditional=false y lance la exception, en cuyo caso me quedo en el estado en que estaba

```
}
```

MHP 1.1.2 specs, A068r1

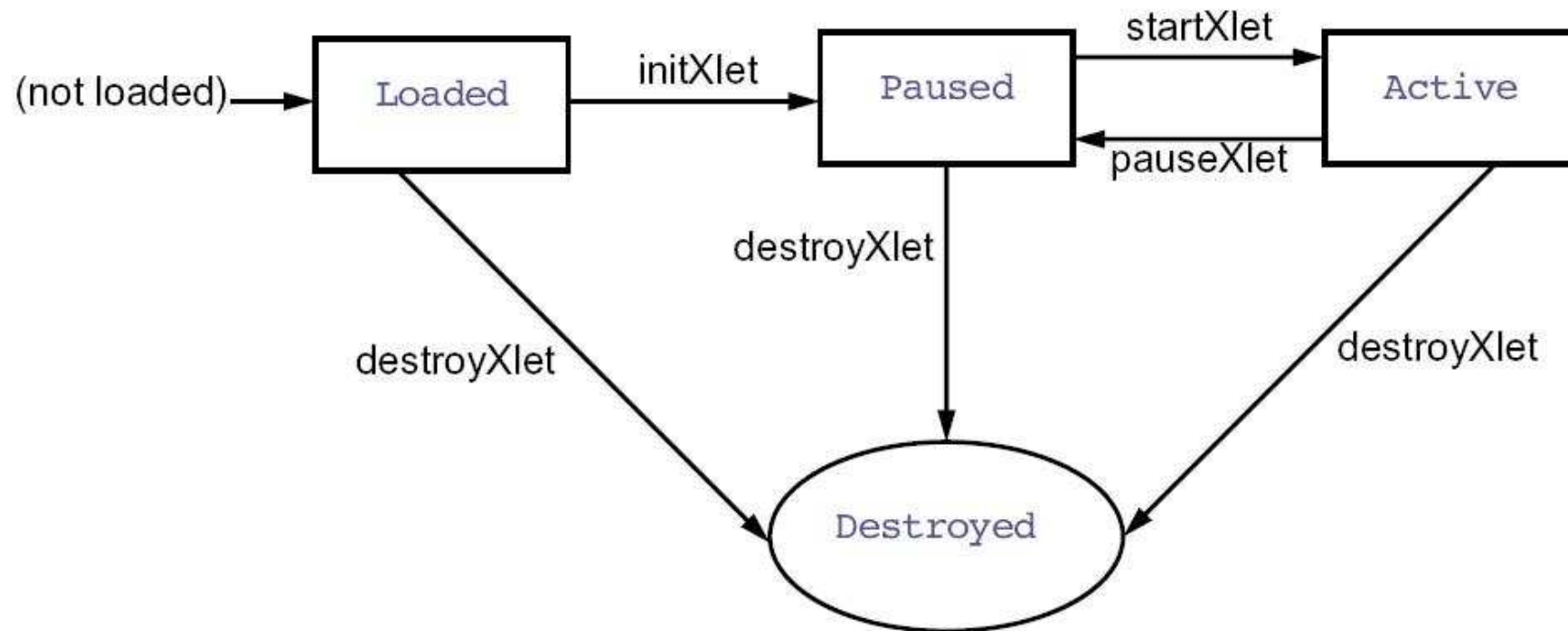


Figure 15: Xlet lifecycle state machine diagram

Antes de seguir, entendamos el “contexto” de las Aplicaciones MHP

- Unidad básica y central: el Service (el canal, vamos).
- Un Service consiste en un conjunto de piezas como señal de audio, video, aplicaciones, información de servicio...etc.
- Un servicio se ejecuta en un **contexto del servicio**. En un MHP STB un servicio se presenta en un **Service Context**, que viene a ser un entorno de ejecución y presentación del servicio, y que en gran medida viene a definir cómo se ejecutan las aplicaciones MHP. Podría incluso darse el caso de que se presentasen simultáneamente 2 servicios, por ejemplo: PIP. El componente ServiceContext permitirá gestionar este tipo de situaciones.

- En un STB MHP tiene preponderancia el hecho de que sea un entorno MHP, esto lo condiciona todo: es MHP quien describe cómo se presentan los servicios etc...Véase: *“Every service that gets presented by an MHP platform is presented within a service context”*.
- El service context:
 - Es un entorno en el cual el Service se presenta.
 - Define los límites del Servicio, describiendo las piezas que lo componen, e incluso permite manejarlo como una unidad.
 - Permite a la plataforma y aplicaciones identificar cuales de las piezas que se están presentando pertenecen al servicio
- En una aplicación **DVB-J** (DVB-J= the Java platform defined as part of the MHP specification) este Service Context se representa por la clase
javax.tv.service.selection.ServiceContext

- Pero ¿ qué nos ofrece un ServiceContext desde el punto de vista de aplicación?
 - **Permite Cambiar de Service!**
 - `javax.tv.service.selection.ServiceContext.select(...)`
 - Un ServiceContext nos informa cuando un Service se ha dejado de presentar y ha llegado otro
 - Mediante este API podemos parar la presentación de un Service...
 - Mediante este API podemos acceder a los “componentes” que configuran el servicio...
 - Este API permite que Aplicaciones que no van unidas a un Servicio se puedan gestionar...
- En otro capítulo profundizaremos en este API, lo importante es “ser conscientes” del nuevo contexto.

Retomamos. Esquema del Ciclo de Vida

- **Antes de ver los estados y flujos en detalle. Importante: la gestión de estados de un Xlet pretende aproximarse al máximo a lo que un telespectador espera:**
 - El tiempo de carga puede/debe ser muy corto
 - El Xlet puede encontrarse en un estado en el que no haga nada.
 - El Xlet puede morir en cualquier momento

Not loaded

- El appManager crea una instancia de la clase Xlet especificada. Si no hay ningún problema en su construcción pasa a estado **Loaded**. Si no es así se descarta cualquier instancia. **El Xlet debe tener un constructor por defecto vacío.**

MHP 1.1.2 specs, A068r1

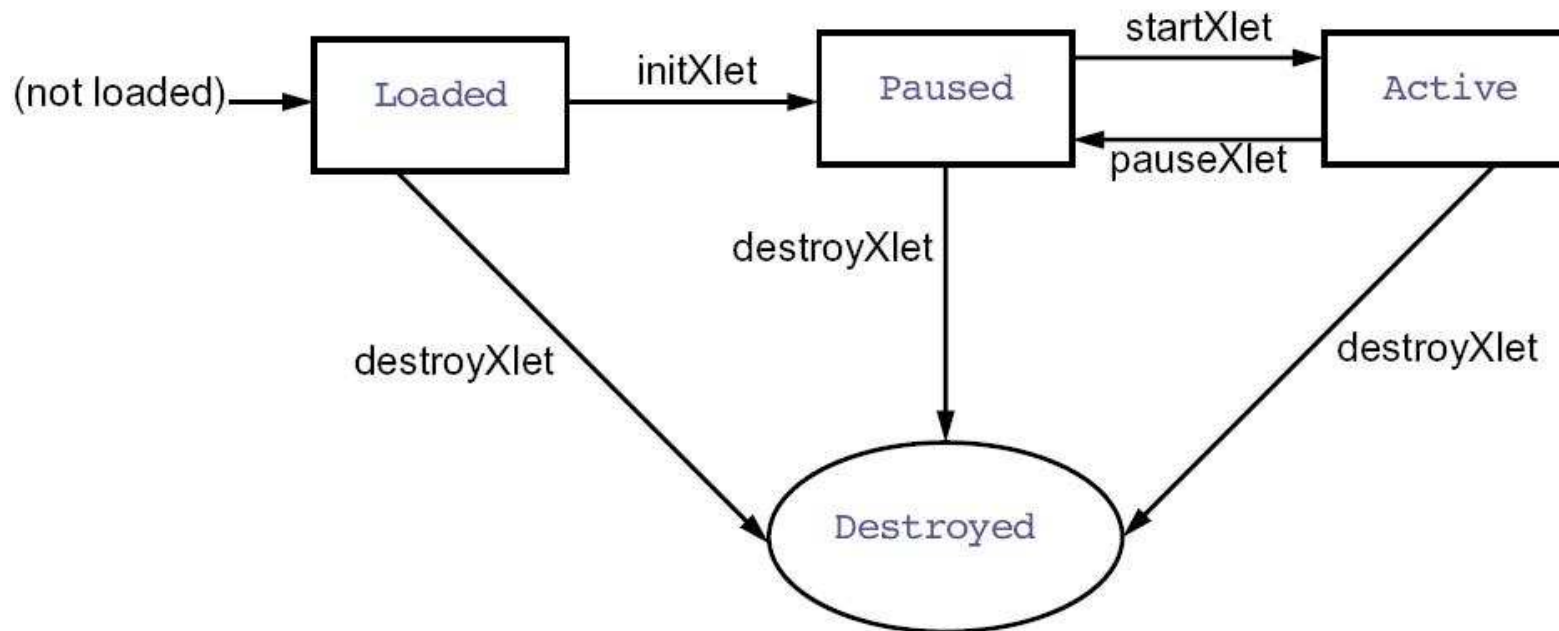


Figure 15: Xlet lifecycle state machine diagram

Not loaded

- **ATENCIÓN:** En el proceso de instanciación de un XLET os podría interesar asegurar la disponibilidad de determinadas classes, por ejemplo, si estas se van a bajar dinámicamente después cuando sean usadas realmente.
- Bastaría con definir en ellas un constructor vacío e instanciarlas como variables del Xlet, aunque luego no las uséis.
- OJO: se trata de asegurar disponibilidad no de cargar procesos!!!

Loaded

- Puede pasar a **Paused**
 - En cuando el Application Manager pueda llamará al método **initXlet(XletContext)** después del cual, si no se ha producido ningún problema, el Xlet pasará a estado **Paused**.

```
public void initXlet(XletContext ctx) throws XletStateChangeException;
```
 - Es en la llamada a **initXlet** donde se recibe un objeto (**que se debe guardar**) del tipo **XletContext**.
 - La llamada a **initXlet** sólo es efectuada una vez para cada instancia del XLET.
- Puede pasar a **Destroyed**
 - Si se produce una excepción **XletStateChangeException** en la llamada al método **initXlet** entonces la instancia permanece en estado **Loaded**, pero su única transición posible será a **Destroyed**, lo cual puede hacer ella misma usando **XletContext.notifyDestroyed()** o bien esperar a que ocurra.

- **Muy importante:** La inicialización de las estructuras de datos, procesos, etc deberá hacerse en el initXlet, NUNCA en el constructor. Este debe estar vacío.

Paused

- El Xlet debe permanecer sin actividad, y es **MUY IMPORTANTE** que **libere al máximo sus accesos a recursos**, si quiere tener posibilidades de sobrevivir.
- Algunas acciones concretas a tener en cuenta;
 - Debe asegurarse que no existe ninguna **HScene** suya visible.
 - No deberá crear nuevas **HScenes** ni hacerlas visibles.
 - **No deberá pintar.**
 - Parada de Threads
- **Cualquier aplicación que NO actúe correctamente será considerada como “NO cooperante” y el AppManager podrá decidir destruirla.**

Paused

- A Paused se llega desde
 - **Loaded**: después de initXlet
 - **Active**: una vez que el AppManager llama a pauseXlet() sin problemas
 - **Active**: después de que se ha llamado explícitamente al método XletContext.notifyPaused()
 - **Paused**: Si lanzamos la excepción XletStateChangeException en startXlet(). entonces permanecerá en Paused.
- Puede ir a **Active**
 - Puede **solicitar** pasar a Active mediante la llamada XletContext.resumeRequest();
 - Después de que se le llame al método startXlet() sin problemas

Paused

- Puede ir a Destroyed
 - Si llamamos a `XletContext.notifyDestroyed()` en la llamada `startXlet()` por ejemplo o en `notifyPaused()`;
 - Después de que el `appManager` llame a `destroyXlet(boolean unconditional)`

Active

- Se llega desde **Paused** únicamente
- Puede ir a **Paused**
 - Porque el AppManager llama a `pauseXlet()`
 - Después de llamar explícitamente al método `XletContext.notifyPaused()`
- Puede ir a **Destroyed**
 - Porque el AppManager llama a `destroyXlet(boolean unconditional)`
 - Después de llamar explícitamente al método `XletContext.notifyDestroyed()`

Destroyed

- A Destroyed se llega desde
 - **Loaded**: problemas en la inicialización (init...)
 - **Loaded**: llamando directamente a `XletContext.notifyDestroyed()` en `initXlet(...)`
 - **Paused**: llamada a `XletContext.notifyDestroyed()` o por llamada a `destroyXlet()`
 - **Active**: llamada a `XletContext.notifyDestroyed()` o por llamada a `destroyXlet()`
- Puede **evitar pasar a Destroyed** cuando se le llama a `destroyXlet(...)` si llega el parámetro `unconditional=false` y lanza la excepción `XletStateChangeException`

Destroyed

- Acciones a realizar antes de que se destruya el Xlet
 - Threads creados han de finalizar voluntariamente (no hay Stop())
 - Parar, liberar los recursos y cerrar todos los JMF players que se hayan creado
 - Parar y destruir todos los objetos JavaTV service selection que se hayan creado
 - Liberar todos los recursos “caros” que se hayan creado, p.e. NetworkInterfaceControllers si efectúan tuning.
 - Flush de las Images usando el método Image.flush().
 - Los Xlets no provocarán ningún retraso innecesario en su método destroyXlet
 - De-registrar todos los event listeners.

Resumen de estados válidos/llamadas

- Recomendación: leed Table 72: Valid lifecycle states of DVB-J application instances (A0068r1)

MHP 1.1.2 A0068r1

Table 73: States for valid state management calls

| Call | State |
|-----------------|-------------|
| notifyDestroyed | all states |
| notifyPaused | active only |
| resumeRequest | paused only |

API Javax.tv.xlet.XletContext

- public static final String **ARGS** = "javax.tv.xlet.args"
 - Clave para obtener los parámetros de inicialización
XletContext.getXletProperty(XletContext.ARGS) los devolverá como un array de Strings. **NO HAY (PARAMETRO, VALOR) sino lista de Strings definidos como parámetros.**
- public void **notifyDestroyed()**;
 - **Notifica** que está Destroyed. El appManager **NO** llamará a destroyXlet.
 - Se deberá liberar recursos antes de llamar.
- public void **notifyPaused()**;
 - **Notifica** que ha pasado de Active a Paused. El appManager **NO** llamará a pauseXlet
- public Object **getXletProperty**(String key)
 - acceso a propiedades del contexto.
- public void **resumeRequest()**;
 - **Desea** pasar a Active

Modos de Arranque

- Veamos en detalle los posibles valores que la **AIT** utiliza para determinar el modo en que una aplicación deberá arrancar (**o parar**). Esto es la base del ciclo de vida de las apps

MHP 1.1.2 A068r1

Table 78: DVB-J application control code values

| code | identifier | semantics |
|--------------|------------|--|
| 0x00 | | reserved_future_use |
| 0x01 | AUTOSTART | The file system element(s) (e.g. an Object Carousel module) containing the class implementing the Xlet interface is loaded, The class implementing the Xlet is loaded into the VM and an Xlet object is instantiated, and the application is started subject to usual restrictions, etc. |
| 0x02 | PRESENT | Indicates that the application is present in the service, but is not autostarted. |
| 0x03 | DESTROY | When the control code changes from AUTOSTART or PRESENT to DESTROY, the destroy method of the Xlet is called (with the unconditional parameter set to false) by the application manager and the application is allowed to destroy itself gracefully. |
| 0x04 | KILL | When the control code changes from AUTOSTART or PRESENT to KILL, the destroy method of the Xlet is called (with the unconditional parameter set to true) by the application manager. |
| 0x05 | | reserved_future_use |
| 0x06 | REMOTE | This identifies a remote application that is only launchable after service selection. |
| 0x07 to 0xFF | | reserved_future_use |

En general

- Sólo se lanzan apps compatibles con la versión de MHP del STB. Esto se sabe por el signalling de la App (lo veremos en detalle en el capítulo APP Signalling)
- Sólo una aplicación con un identificador único, Organisation id & App id, puede estar ejecutándose simultáneamente.

Aplicaciones auto-start

- Se arrancan automáticamente cuando el usuario selecciona el Servicio, El STB monitoriza el signalling del broadcast para determinar su finalización.
- Pueden aparecer nuevas como auto-start o bien alguna de las existentes cambia su modo, entonces, el receptor la lanzará dependiendo de los recursos...

Aplicaciones Present

- Cuando se selecciona un canal será necesario ofrecer al usuario la posibilidad de ejecutar aquellas aplicaciones marcadas como Present
- Dado que un STB MHP **no está obligado a disponer de un Launcher, el Broadcast deberá ofrecer una aplicación MHP de tipo auto-start que de acceso a la ejecución de las aplicaciones no auto-start disponibles.**

Aplicaciones Present

- Interesante:
 - Engel tdt6000i: MHP 1.0.2
 - Dispone de aplicación residente en el DECO, con lo que no se ejecuta la lanzadera del Broadcast.
 - La lanzadera aparece como opción !!!
 - Strong 5110: MHP 1.1.2
 - No dispone de aplicación residente en el DECO. Se arranca sola la auto-start lanzadera del Broadcast.
- Mediante el “**application listing and launching API**” definido en el anexo S, org.dvb.application, se permite a una aplicación arrancar otra MHP (por supuesto, sujeta a los requerimientos de seguridad). Lo veremos en el capítulo correspondiente.

Parada de Aplicaciones sin cambio de Canal

- Por la aplicación en sí.
- Cuando una aplicación mata a otra aplicación.
- Application Signalling realizados por el Broadcaster: KILL/DESTROY. Un ejemplo de este caso puede ser el de aplicaciones ligadas a un evento.
 - **Quedaos con la diferencia entre KILL / DESTROY: destroy pasa el parámetro unconditional = false y KILL a true**
- Si el terminal está mal de recursos, puede decidir, sin intervención del usuario, matar una aplicación.

Persistencia de Aplicaciones en el flujo de cambio de Servicios

- Cuando se cambia de canal: sólo se mantendrá su ejecución en el nuevo si la aplicación es “**bound_to_service**” = **false** y **está signalled** en la nueva AIT. **Recordemos que cada Servicio ofrece sus aplicaciones.**
- Cuando se cambia de canal: si una aplicación está “**bound_to_service**” = **true** el STB la parará sin esperar al signalling del nuevo canal. **Esto favorece que las auto-start del nuevo se ejecuten antes.**
- Si en el nuevo servicio está marcada como **auto-start** se reiniciará y no se mantendrá ninguna información volátil del servicio anterior.

Persistencia de Aplicaciones en el flujo de cambio de Servicios

- Carousel File System: Cuando una aplicación sobrevive a un cambio de servicio, no obtendrá acceso inmediato al Carousel File systems en el nuevo, sino que,
 - en primer lugar estará un tiempo desconectada, y
 - en segundo lugar tendrá acceso a los Carousel File Systems identificados unitariamente en el Broadcasting, con independencia del servicio en el que inicialmente obtuvo acceso a ellos. El sistema MHP se encargará de reactivar el acceso a los file systems, cuando los encuentre disponibles.

Stored Channels & Cached APPS

- Cuando toda una aplicación o parte de ella se carga desde caché, en la práctica funciona igual que una app cargada desde broadcasting, la única diferencia es que se ha cargado con mayor velocidad.

Veremos el detalle de Stored Channels y Cached Apps en el capítulo correspondiente

Monitorización de la AIT

- Los cambios en la AIT se deben detectar por el terminal en menos de un segundo, de manera que el Broadcast puede decidir cómo se ha de comportar una app en cualquier momento y la respuesta es prácticamente inmediata. (recordamos la PMT ?)

Dominio de una Aplicación

- ¿ Qué es el **dominio de una aplicación** ? Aquellos servicios en los que la aplicación se puede ejecutar, bien porque está signalled en la AIT o porque está en el **External Application Permission Descriptor** (lo veremos en detalle en Signalling de Apps): es una parte de la AIT que indica qué aplicaciones externas tienen permitido continuar siendo ejecutadas en el servicio, pero no ejecutarse de nuevas en el mismo.

Ahora a trabajar!!

Previo

- **INSTRUCCIONES CURSO-PRESENCIAL 1.0**

Ejercicios Bloque CICLO-1

¿ **Cómo podemos obtener los parámetros de nuestro Xlet ?**

- `XletContext.getXletProperty(XletContext. ARGS)`
- como un array de String

¿ **Hay más properties ?** Si. Las vemos,

MHP System Properties

- A través de `System.getProperty(...)` y `XletContext.getXletProperty(..)` se accede a las propiedades que el sistema ofrece para saber, por ejemplo, el **user.dir**.
- En la Especificación existen otras propiedades. Se detallan a continuación con su descripción. (Algunas ya las conocemos de los Profiles y sus opciones....)

MHP System Properties

| Property | Description | FROM X=Xletc S=System |
|-----------------------------------|--|-----------------------------|
| javax.tv.xlet.args | String param name to obtain XLET init params from context. | X |
| dvb.org.id | APP OrgID | X |
| dvb.app.id | APP AppID | X |
| user.dir | User dir | S |
| path.separator | Path separator in filenames (will always be '/' in OCAP) | |
| dvb.persistent.root | Root directory for persistent storage | S |
| dvb.returnchannel.timeout | Timeout period (in seconds) for return channel connections | S |
| dvb.display.aspect_ratio | shall indicate the shape of the physical display that the MHP terminal believes is used by its default HScreen | S |
| dvb.caller.parameters | Parameters passed when launched from AppProxy | X |
| dvb.installer.parameters | parameters to be available to the applications when started | X |
| dvb.security.pkcs11.defaultSlotId | dvb.security.pkcs11.defaultSlotId | S |
| mhp.profile.enhanced_broadcast | Indicates whether the enhanced broadcast profile is supported (will always be YES since this is the minimum profile) | S |
| mhp.profile.interactive_broadcast | Indicates whether the interactive broadcast profile is supported (YES if it is NO or null otherwise) | S |
| mhp.profile.internet_access | Indicates whether the Internet access profile is supported (YES if it is NO or null otherwise) | S |
| mhp.eb.version.major | Major version number of the supported enhanced broadcast profile or null if not supported | S |
| mhp.eb.version.minor | Minor version number of the supported enhanced broadcast profile or null if not supported | S |
| mhp.eb.version.micro | Micro version number of the supported enhanced broadcast profile or null if not supported | S |
| mhp.ib.version.major | Major version number of the supported interactive broadcast profile or null if not supported | S |
| mhp.ib.version.minor | Minor version number of the supported interactive broadcast profile or null if not supported | S |
| mhp.ib.version.micro | Micro version number of the supported interactive broadcast profile or null if not supported | S |
| mhp.ia.version.major | Major version number of the supported Internet access profile or null if not supported | S |
| mhp.ia.version.minor | Minor version number of the supported Internet access profile or null if not supported | S |
| mhp.ia.version.micro | Micro version number of the supported Internet access profile or null if not supported | S |
| mhp.option.ip.multicast | Has a value of SUPPORTED if the system supports IP multicast in transport streams | S |
| mhp.option.dsmcc.uu | Has a value of SUPPORTED if the system supports DSM-CC User-To-User protocol over the return channel | S |

MHP System Properties (y 2)

| Property | Description | FROM X=Xletc S=System |
|--|--|-----------------------------|
| mhp.option.dvb.html | Has a value of SUPPORTED if the system supports DVB-HTML applications | S |
| mhp.stored.services | Flag showing how much memory is reserved for stored services and applications (if any memory is reserved). This does not reflect how much of that memory is actually free. (MHP 1.1. only) | S |
| mhp.smartcard.reader | Has a value of SUPPORTED if the system has a smart card reader accessible using the smart card API (MHP 1.1. only) | S |
| mhp.option.memory.card | Has a value of SUPPORTED if the system supports memory card devices | S |
| mhp.option.opentype | Has a value of SUPPORTED if the system supports OpenType font technology | S |
| mhp.option.highdef | Has a value of SUPPORTED if the system supports high-definition TV | S |
| mhp.option.http.1.1 | HTTP 1.1 over the interaction channel | S |
| mhp.option.dsmcc.uu | DSMCC user-to-user over the interaction channel | S |
| org.havi.ui.HVersion.HAVI_SPECIFICATION_VENDOR | Vendor name of the HAVi specification. Will always be DVB for MHP systems | S |
| org.havi.ui.HVersion.HAVI_SPECIFICATION_NAME | Name of the HAVi specification that is implemented. Will always be MHP for MHP systems | S |
| org.havi.ui.HVersion.HAVI_SPECIFICATION_VERSION | Version of the HAVi specification that is implemented. This will be the same as the version number of the MHP specification that is implemented by the receiver | S |
| org.havi.ui.HVersion.HAVI_IMPLEMENTATION_VENDOR | Name of the vendor for the HAVi implementation (e.g. the middleware vendor) | S |
| org.havi.ui.HVersion.HAVI_IMPLEMENTATION_VERSION | Version of the HAVi implementation | S |
| org.havi.ui.HVersion.HAVI_IMPLEMENTATION_NAME | Name of the HAVi implementation | S |

Ejercicios Bloque CICLO-2

¿ Qué debo y NO debo hacer en cada método del ciclo de vida ?

- Inicializaciones: si no cuestan mucho en `initXlet`, si no, en `startXlet`.
- En el `initXlet` no se deben reservar recursos caros que no necesite usar aún
- Se debe de esperar a `startXlet` para reservar recursos caros o estructuras pesadas. Un Xlet puede que no arranque nunca!
- El constructor por defecto no hace nada.
- `startXlet` debe retornar inmediatamente, la forma habitual de reservar recursos, etc **es mediante un Thread que se arranca en `startXlet`.**
- Salvo **`startXlet`** ningún otro método **del ciclo de vida** debería arrancar otro Thread, especialmente `initXlet` o `destroyXlet`.
- Es muy importante ser riguroso con la gestión de recursos “Caros” y “amable” con el resto de Xlets (veremos más adelante Resource Management)

¿ Qué debo y NO debo hacer en cada método del ciclo de vida ? (y 2)

- Cuando se comparten recursos con otras aplicaciones, por ejemplo, un HScreen, no debemos modificar parámetros que no nos importen.
- En pauseXlet se deben liberar tantos recursos caros como podamos. La aplicación debe dejar de pintar, todos los componentes deben ocultarse y no deberá de pintar nada en la pantalla, si lo hace podrá ser considerada hostil y ser destruida
- Nunca llamar a System.exit
- DestroyXlet debería parar todos los Threads creados por la aplicación.
- DestroyXlet debería cancelar todas las peticiones asíncronas en curso.(ya veremos ese tipo de peticiones)
- DestroyXlet e idealmente pauseXlet deberían de liberar todos los componentes gráficos que se hayan creado. Para hacerlo correctamente se debe llamar a `java.awt.Graphics.dispose()`. El middle puede que no los libere correctamente.

¿ Qué debo y NO debo hacer en cada método del ciclo de vida ? (y 3)

- Tengamos en mente que una aplicación puede pasar a Paused o Destroyed en cualquier momento y que hemos de estar preparados para liberar recursos.
- NUNCA capturar ThreadDeath Exceptions: se deben liberar recursos en destroyXlet()
- No usar los Finalizers, puede que no se llamen.
- No confiar en que se pueden cargar clases en una llamada a destroyXlet
- CAPTUREN LAS EXCEPTIONS QUE LANZA EL MIDDLEWARE y reaccionen ante ellas para mantener la aplicación estable

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