



Curso Multimedia Home Platform 1.1.2

Tuning API

Sintonizando otros TS

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

- De todos los APIS que están involucrados en manejar streams de Services sólo JavaTV ServiceContext API y Java Tuning API harán un Tuning para cambiar de TS si lo necesitan.
- En el caso de Java TV ServiceContext se hace “silently” pero en el caso del Tuning API el proceso será explícito.
- Cuando cambiamos de Service dentro del mismo TS la implicaciones son menores que cuando hay que ir a otro TS pues la frecuencia de transmisión es otra y por tanto la disponibilidad de ES del Service que se abandona es anulada.

Veamos el API

Tuning API

- El API de Tuning se encuentra en **org.davic.net.tuning**, el cual “rozamos” cuando hablábamos de DVB SI, de hecho lo que vimos era un nexo entre dvb y davic:
 - Mediante la clase **org.dvb.net.tuning.DvbNetworkInterfaceSIUtil** podemos saber qué Tuner, es decir **org.davic.net.tuning.NetworkInterface**, está usando nuestra SIDatabase

```
public static org.davic.net.tuning.NetworkInterface getNetworkInterface(org.dvb.si.SIDatabase sd)
```

Tuning API. Transport Streams Existentes

- Puesto que el eje central es el Transport Stream, empecemos por la clase que me ofrece todos los TS existentes: **org.davic.net.tuning.StreamTable**
- **StreamTable** ofrece el siguiente escueto aunque no por ello menos útil API:
 - public static **org.davic.mpeg.TransportStream[]**
getTransportStreams(org.davic.net.Locator loc)
 - **El Locator nos debe llevar de forma inconfundible al TS deseado.** Si hay más información (un Service) esta se ignorará.
 - **Puesto que un deco puede disponer de varios interfaces (cable, satélite, terrestre) el método devolverá tantos TransportStreams como interfaces de recepción disponga (cable, terrestre, satélite...),** aunque desde un punto de vista lógico todos sean el mismo, el correspondiente al Locator pasado.
 - public static **org.davic.net.Locator[]** **listTransportStreams**()
 - Nos devuelve **todos los Locators para todos los TransportStreams** que puede recibir en todos los network interfaces.

Tuning API. Transport Streams Existentes

- A partir de lo anterior disponemos de **org.davic.mpeg.TransportStream** que nos ofrece el API siguiente, el cual nos lleva a **org.davic.mpeg.Service**:
 - public int **getTransportStreamId()**
 - public Service **retrieveService(int serviceld)**
 - public Service[] **retrieveServices()**
- Service a su vez nos ofrece “su id” y los **org.davic.mpeg.ElementaryStream** que contiene:
 - public int **getServiceId()**
 - public ElementaryStream **retrieveElementaryStream(int pid)**
 - public ElementaryStream[] **retrieveElementaryStreams()**

Tuning API. Transport Streams Existentes

- En **org.davic.mpeg.ElementaryStream** nos quedamos:
 - public int **getPID()**
 - public Integer **getAssociationTag()**
 - Devuelve el DSMCC Association Tag si existe (forma de identificar ES por DSMCC similar al **stream_identifier_descriptor**)
- No obstante de todo lo anterior lo más útil es quizá conocer el **org.davic.net Locator** de los TS.

Tuning API. `NetworkInterface` / `NetworkInterfaceManager`

- Hemos visto como en el API de `StreamTable` se refiere a los **`NetworkInterfaces`** a través de los cuales somos capaces de recibir las señales y por ende los TS.
- El sistema ofrecerá un solo **`org.davic.net.tuning.NetworkInterface`** por dispositivo de conexión (tarjeta terrestre, cable, sat), pero ¿ **cómo obtenemos los diferentes `NetworkInterfaces` existentes ?**: a través de

`org.davic.net.tuning.NetworkInterfaceManager`

Tuning API. NetworkInterface / NetworkInterfaceManager

- org.davic.net.tuning.NetworkInterfaceManager API
 - public class NetworkInterfaceManager **implements ResourceServer**
 - Implementa ResourceServer -> Vamos a tener que gestionar **recursos caros** muy pronto!
 - public **static** NetworkInterfaceManager **getInstance()**
 - Es un **Singleton** (No es ninguna sorpresa).
 - public NetworkInterface[] **getNetworkInterfaces()**
 - Nos da todos los NetworkInterfaces
 - public NetworkInterface **getNetworkInterface**(org.davic.mpeg.TransportStream ts)
 - Nos da el NetworkInterface **por el que llega** este TS.

Tuning API. `NetworkInterface` / `NetworkInterfaceManager`

- `org.davic.net.tuning.NetworkInterfaceManager` API
 - `public void addResourceStatusEventListener(ResourceStatusListener listener)`
 - `public void removeResourceStatusEventListener(ResourceStatusListener listener)`
 - Estos dos nos permiten saber cuando se han liberado y reservado `NetworkInterfaces` (muy similar al de `ConnectionRCInterfaces` (modems))
 - Los eventos que nos proporciona el Listener de `ResourceStatusListener` listener son los siguientes, los cuales nos ofrecen el `NetworkInterface` reservado o liberado.

`org.davic.net.tuning.NetworkInterfaceReleasedEvent`

`org.davic.net.tuning.NetworkInterfaceReservedEvent`

Tuning API. `NetworkInterface` / `NetworkInterfaceManager`

- El API anterior nos ofrece los `NetworkInterfaces` y un api general de listening relacionado con Gestión de Recursos caros, pero todavía NO HEMOS RESERVADO NADA para hacer TUNING!. Porque como ya os habréis imaginado el usar un dispositivo físico para sintonizar es muy caro.

Tuning API. NetworkInterface

- Veamos el API de `org.davic.net.tuning.NetworkInterface`:
 - `public org.davic.mpeg.TransportStream getCurrentTransportStream()`
 - TS actualmente tuned o null si no hay ninguno o si está haciendo tuning en este momento.
 - `public org.davic.net.Locator getLocator()`
 - Locator del **TS** actualmente tuned o null si no hay ninguno o si está haciendo tuning en este momento.
 - `public synchronized boolean isReserved()`
 - Si está reservado. (muy útil).
 - `public boolean isLocal()`
 - Si el tuner es local al deco o se está usando uno remoto (viene de otras épocas)

Tuning API. NetworkInterface

- API de org.davic.net.tuning.NetworkInterface
 - public **org.davic.mpeg.TransportStream[] listAccessibleTransportStreams()**
TS disponibles a través de este NI (array vacío si no hay ninguno)
 - public **int getDeliverySystemType()**
 - DeliverySystemType.CABLE_DELIVERY_SYSTEM (1), TERRESTRIAL_DELIVERY_SYSTEM(2), SATELLITE_DELIVERY_SYSTEM(0)
 - public void **addNetworkInterfaceListener(NetworkInterfaceListener listener)**
 - public void **removeNetworkInterfaceListener(NetworkInterfaceListener listener)**
 - **Suscripción a eventos** relacionados con este NetworkInterface Concreto. Estos son:
NetworkInterfaceTuningEvent: Ha comenzado a sintonizar
NetworkInterfaceTuningOverEvent : Ha terminado. (getStatus() SUCCEEDED = 0, FAILED = 1)

Tuning API. NetworkInterface

- El API de **org.davic.net.tuning.NetworkInterface** nos ofrece solo información, no podemos todavía efectuar ninguna acción relacionada con Tuning. Esto lo haremos finalmente con la clase: **org.davic.net.tuning.NetworkInterfaceController**

Tuning API. NetworkInterfaceController

- Veamos el API de org.davic.net.tuning.NetworkInterfaceController de forma temporal
 - **Paso 0**: Hemos de implementar el **ResourceClient** Interface
 - **Paso 1**: **Construimos** el objeto. Hemos de pasarle nuestro ResourceClient (**recursos caros**).

```
public NetworkInterfaceController(ResourceClient rc)
```

Tuning API. NetworkInterfaceController

- **Paso 2:** Hemos de efectuar la reserva del **NetworkInterface** que se va a usar para sintonizar a un nuevo **TS**. Existen varias posibilidades:

```
public synchronized void reserve(NetworkInterface ni, Object requestData)
```

- Reservamos el **NetworkInterface concreto** que queremos.

```
public synchronized void reserveFor(org.davic.net.Locator locator, Object requestData)
```

- Reservamos el **NetworkInterface sobre el que se transmite el TS del locator** pasado. Es decir, es una indirección.
- Imaginemos que conocemos el Locator del Servicio.

- Si la reserva resulta OK se lanzará el **NetworkInterfaceReservedEvent** de NetworkInterfaceManager Listener API.

Tuning API. NetworkInterfaceController

- **Paso 3:** Una vez que tenemos el recurso reservado y antes de realizar el Tuning hemos **de suscribirnos al gestor de Eventos del NetworkInterface** reservado, pues los métodos de Tuning son asíncronos y será mediante la notificación de estos eventos como sabremos como ha ido la operación de sintonización:

```
getNetworkInterface().addNetworkInterfaceListener(NetworkInterfaceListener listener)
```

```
public interface NetworkInterfaceListener extends java.util.EventListener {  
    public void receiveNIEvent(NetworkInterfaceEvent anEvent);  
}
```

- Los eventos que se recibirán serán en realidad los dos tipos siguientes que veremos a continuación.

NetworkInterfaceTuningEvent

NetworkInterfaceTuningOverEvent

Tuning API. NetworkInterfaceController

- **Paso 4:** Ya estamos en disposición de realizar el **Tuning** al TS deseado a través de las llamadas **asíncronas** siguientes:

```
public synchronized void tune(org.davic.net.Locator locator)
```

```
public synchronized void tune(TransportStream ts)
```

- Después se lanzará en primer lugar el evento siguiente de **NetworkInterfaceListener** para decirnos que ha comenzado a sintonizar:

NetworkInterfaceTuningEvent

- Y después llegará el evento que nos dice que ha dejado de sintonizar porque o bien ha conseguido el objetivo o no se ha podido, lo cual sabremos gracias el método del evento:
int getStatus() SUCCEEDED = 0, FAILED = 1.

NetworkInterfaceTuningOverEvent

Tuning API. NetworkInterfaceController

- **Paso 5:** Una vez finalizado el proceso es el momento de liberar los recursos:
public synchronized void **release()**
- Lo cual, si la liberación se hace OK lanzará el **NetworkInterfaceReleasedEvent** de NetworkInterfaceManager Listener API.

Implicaciones de hacer TUNING

- Cambiar de TS implica muchas cosas, como por ejemplo que los Object Carousels dejarán de estar disponibles....
- Salvo que sea absolutamente necesario se recomienda no cambiar de TS, y si se hace, conviene re-acceder a los recursos relevantes de antes del cambio puesto que las caches y recursos caros puede que no se hayan reseteado convenientemente (DVB-SI, Section Filtering, Object Carousels...)

Seguridad

- Recordemos que las unsigned no pueden usar el Tuning API y que las signed sólo si lo piden en el PRF de la forma:

```
<!ELEMENT tuning EMPTY><
```

```
<!ATTLIST tuning value (true|false) "true" >
```

Ejercicios Bloque TUNING-1

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