



# Curso Multimedia Home Platform 1.1.2

## System Information I. DVB-SI

“Localizando” componentes

Qué llega en la señal

Accediendo a su contenido con DVB-SI

## 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.

## Locators

- ¿ **Cómo referenciar contenidos MPEG ?**: mediante lo que se llaman **Locators** y su representación en formato de String.
- No se usan URLs, aunque la mayoría de las referencias tienen representación en este formato.
- Tipos de Contenido referenciables MEDIANTE LOCATORS:
  - **dvb streaming content**
    - Transport Streams
    - Elementary Streams
    - Services
    - Events
    - Files (no muy usado, Y NO MUY RECOMENDABLE)
  - **Files en un Broadcast File System** (DSMCC Object Carousel)
  - **VideoDrips** (forma eficiente sobre mpeg de representar imágenes, que se pueden usar de fondo en un HBackgroundDevice)

## Locators para contenido DVB Streaming

- Esquema general de representación tipo URL:

**dvb://netID.tsID.serID[.ctag[&ctag]][;evid][path]** Siempre se referirá a un Service

- **netID**: Network ID de quien produjo el contenido, no de quien lo está transmitiendo.
- **tsID**: Transport Stream ID de un TS que transmite la Network
- **serID**: Service ID de un Service dentro del TS
- **cTag**: Es un tag correspondiente a un Elementary Stream concreto que hay sido “tageado” con ese tag (recordamos el **stream\_identifier\_descriptor** en AppSignalling?), o simplemente el id del ES. ¿ por qué puede haber varios?:para poder referirnos a un grupo.
- **evid**: ID de un Evento parte del servicio
- **path**: a un fichero que se está retransmitiendo en un Broadcast File System en ese Service (si solo hay uno) o en ese ES particular

Ejemplos:

DVB Service    dvb://12.43.55

Event            dvb://32.3.44;34

Fichero         dvb://2.45.55/resources/lion.jpg

ES                dvb://45.67.36.23

## Locators para contenido Dvb Streaming

- Volvamos al Fichero: `dvb://2.45.55/resources/lion.jpg`
- **NO es recomendable** utilizar este formato pues asume que el recurso se encuentra en el Object Carousel por defecto y que cuelga del directorio raíz indicado, lo cual es mucho decir...
- En todo caso, para evitar problemas si hay más de un Object Carousel, es preferible indicar el ES concreto donde se supone que está montado el mismo:
  - `dvb://2.45.55.4d/resources/lion.jpg`

## Locators para Ficheros

- Ya hemos visto la opción **dvb**: sin embargo esto no es muy recomendable cuando vamos a acceder a ficheros locales, para ello utilizaremos el objeto **org.dvb.dsmcc.DSMCCObject**, que para simplificar, es un `java.io.File` que nos da acceso a un directorio (por defecto a aquel en el que reside nuestra app). Veremos en detalle los diferentes esquemas de acceso: absoluto y relativo (**¿ recordamos APP Signalling ?**)
- Mediante **org.dvb.dsmcc.DSMCCObject** podemos acceder a un fichero de la forma **`new DSMCCObject("dir/dir/file.txt")`** , sin embargo **si queremos crear un Locator necesitamos el formato: `file:/`**
- Para obtener el formato `file:` simplemente lo obtenemos a partir del método **`getURL()`** de `DSMCCObject`
- Si existen más de un Object Carousel veremos en el capítulo dedicado a DSMCC Files & Directories la forma de proceder.

## Locators para Ficheros

- Salida de varios métodos de DSMCCObject al pasarle los Original path descritos

**Original path** :images/exercise23/mandrill.jpg

AbsolutePath :/home/images/exercise23/mandrill.jpg

CanonicalPath:/home/images/exercise23/mandrill.jpg

Path :images/exercise23/mandrill.jpg

URL :file:/home/images/exercise23/mandrill.jpg

toString :images/exercise23/mandrill.jpg

**Original path** :/home/images/exercise23/mandrill.jpg

AbsolutePath :/home/images/exercise23/mandrill.jpg

CanonicalPath:/home/images/exercise23/mandrill.jpg

Path :/home/images/exercise23/mandrill.jpg

URL :file:/home/images/exercise23/mandrill.jpg

toString :/home/images/exercise23/mandrill.jpg

## Locators para Ficheros

- **OJO: esto no funciona:**

```
DSMCCObject dsmcc = new DSMCCObject("images/mandril.jpg");
javax.tv.locator.Locator loc =
    javax.tv.locator.LocatorFactory.getInstance().createLocator(dsmcc.getURL().toString())
```

- Al menos en Strong 5510 y Engel 6000i con un fichero en local: Ved “/logs/Problema tvLocator.txt”. **Sin embargo con javax.media.MediaLocator SÍ** (en detalle en DSMCC).

1.- El URL.toString() devuelve : file:/images/...

2.- Ni pasándole file://images....funciona:

```
[3#11:1] !----->Original path :images/exercise_dsm1/mandrill.gif
[3#11:1] !-----
[3#11:1] !----->AbsolutePath :/home/images/exercise_dsm1/mandrill.gif
[3#11:1] !----->CanonicalPath :/home/images/exercise_dsm1/mandrill.gif
[3#11:1] !----->Path :images/exercise_dsm1/mandrill.gif
[3#11:1] !----->URL.toString :file:/home/images/exercise_dsm1/mandrill.gif
[3#11:1] !----->URL.toExternalForm :file:/home/images/exercise_dsm1/mandrill.gif
[3#11:1] !----->toString :images/exercise_dsm1/mandrill.gif
[3#11:1] !-----
[3#11:1] !----- Locator info for uri:file://home/images/exercise_dsm1/mandrill.gif
[3#11:2] javax.tv.locator.MalformedLocatorException: file://home/images/exercise_dsm1/mandrill.gif
```

## Locators para videodrips, un caso muy especial

- ¿ Qué es un VideoDrip ? Es una forma eficiente de presentar una imagen usando una secuencia de Frames MPEG. Consiste en disponer de un I-Frame, donde reside la imagen, seguido de varios P-Frames.
- En realidad un VideoDrip sólo puede referenciarse cuando ya está en memoria de manera que la URL es como sigue:

dripfeed://

Se ve con más detalle en el capítulo dedicado a Java Media.

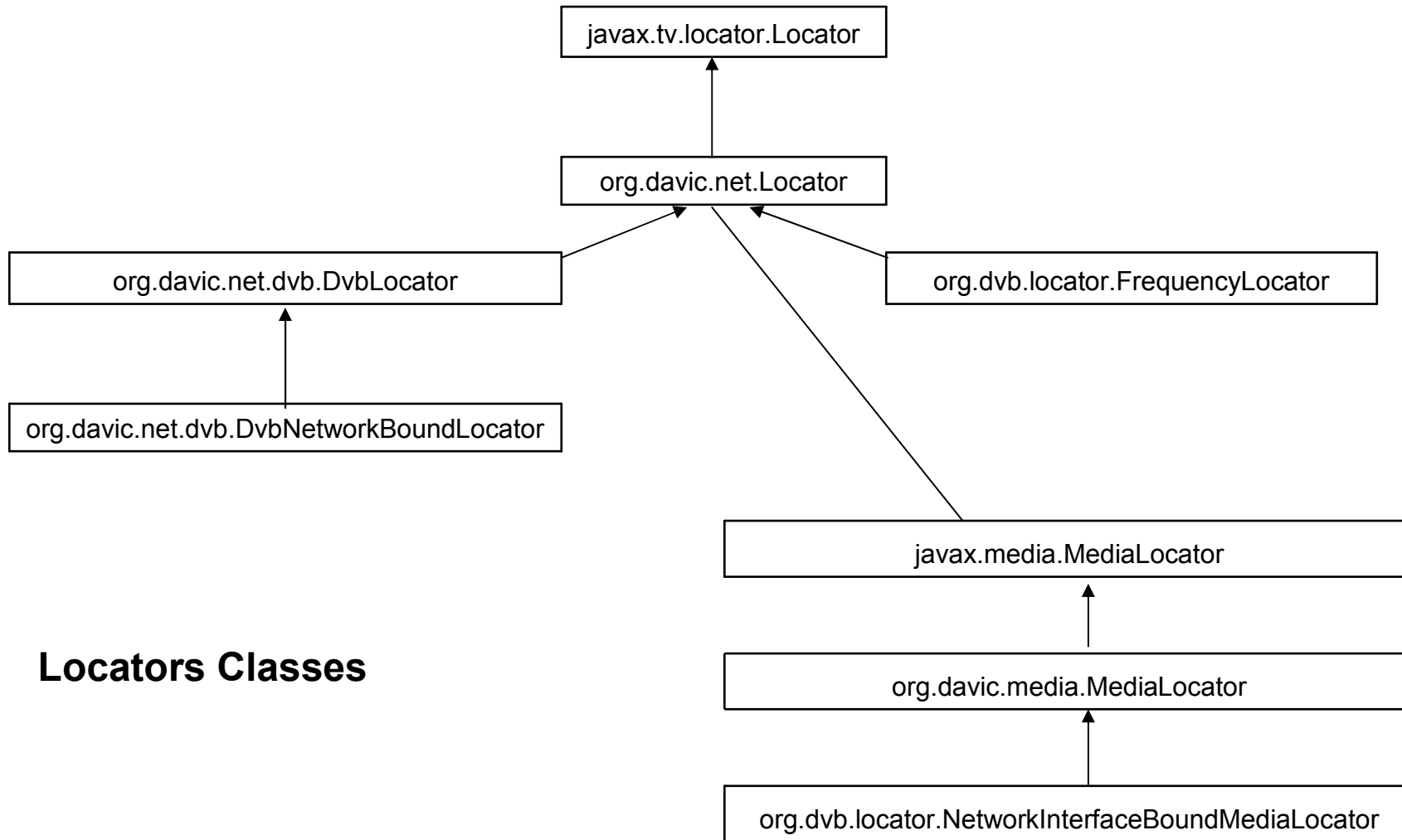
## Locators Classes

- Existen dos grupos diferenciados:

| <p style="text-align: center;"><b>Grupo MEDIA</b></p> <p>Sólo se usa para control de presentación de Media por parte de <b>Java Media Control API: JMF</b></p> <p style="text-align: center;"><b>Para Services, Audio y VideoDrips</b></p>                                      | <p style="text-align: center;"><b>Grupo TV</b></p>  |
|---|---|
| <p><b>javax.<u>media</u>.MediaLocator</b></p>   | <p><b>javax.<u>tv</u>.locator.Locator</b></p> <p> puede referirse a cualquier contenido Media que se distribuya en un DTV: TS, Service, File</p>  |
| <p><b>org.davic.media.MediaLocator</b></p> <pre>public class MediaLocator extends <b>javax.media.MediaLocator</b> {     public MediaLocator(<b>org.davic.net.Locator</b> loc) } </pre>  | <p>Para elementos referenciables como dvb://....</p> <p><b>org.davic.net.Locator</b> (implements javax.tv.locator.Locator)</p> <p>org.davic.net.dvb.DvbLocator (<b>extends</b> Locator)</p> <p>org.davic.net.dvb.DvbNetworkBoundLocator (<b>ext.</b>DvbLocator)</p> |
| <p style="text-align: center;">La clase <b>org.davic.media.MediaLocator</b> conecta el Grupo TV con el Grupo MEDIA puede construirse con un <b>org.davic.net.Locator</b>: así se puede tomar un Locator del Grupo TV y construir fácilmente un Player para verlo con JMF!!!</p> |   |

## Locators Classes

- **org.davic.net.dvb.DvbNetworkBoundLocator** es un caso especial: Se usa cuando por ejemplo disponemos de un deco en el que se recibe el mismo servicio por dos redes distintas: cable y Satélite. Si deseamos especificar que el contenido debe ser tomado por una de ellas en concreto entonces debemos usar esta clase.



## Locators Classes

## Creación de Locators

- **org.davic.net.Locator** sólo dispone del constructor: `Locator(String url)`
- **org.davic.net.dvb.DvbLocator** dispone de constructores que reciben los datos de los TS, Service... Etc (Ved API Java de los constructores para detalle de opciones).
  - `public DvbLocator(int onid, int tsid)`
  - `public DvbLocator(int onid, int tsid, int serviceid)`
  - `public DvbLocator(int onid, int tsid, int serviceid, int eventid)`
  - `public DvbLocator(int onid, int tsid, int serviceid, int eventid, int componenttag)`
  - `public DvbLocator(int onid, int tsid, int serviceid, int eventid, int[] componenttags)`
  - `public DvbLocator(int onid, int tsid, int serviceid, int eventid, int[] componenttags, String filePath)`
  - `public DvbLocator(String url)`
- **javax.media.MediaLocator** (y su descendencia) también puede construirse usando sus constructores.
  - `public MediaLocator(URL url)`
  - `public MediaLocator(String url) // espera URL.toString()`
- **javax.tv.locator.Locator** sólo se puede obtener a partir de una factoría: `javax.tv.locator.LocatorFactory` que recibe **URL Strings** para construirlos.

## Introducción

- Veíamos al principio del curso que la información de lo que existe en el TS va codificada en tablas de manera que cada tipo de información tiene su TIPO y FORMATO concreto de tabla. Toda esta información es lo que conocemos como System Information, y no son solo las tablas principales que mencionamos sino todas las existentes y que pueden residir en diferentes ES.
- Para la definición de los datos manejaremos el concepto de **Descriptor**. Una Tabla tiene asociado un Descriptor que la describe (☺), pero es que dentro de la definición se hará referencia a otros tipos de información que a su vez se definen con sus **Descriptors** concretos. De esta forma, dado que **cada tipo tiene un Descriptor con su código cabecera único**, el sistema es sumamente flexible al poder incluir un descriptor prácticamente donde queramos.
- Toda esta información es absolutamente necesaria para que todo funcione, hay que saber qué hay en la señal, qué Services (y sus aplicaciones), Programas, opciones de Audio, etc...es necesaria para el usuario y por supuesto para el deco que debe usarla (debe saber qué aplicaciones deben seguir funcionando y cuales no....cómo cambiar de canal...etc etc etc) y mostrarla cuando se solicita.
- En la primera parte de este capítulo vamos a realizar un repaso de las tablas más importantes, sin entrar en su detalle de descriptores, pues esto nos llevaría demasiado tiempo, y sería objeto de un curso dedicado a Transmisión, pero sí conviene saber de donde sale la información.

## Introducción

- Importante: La PSI viaja sobre **MPEG Sections**, de manera que el acceso a su contenido mediante filtros “físicos” acelera su recuperación (...Section Filtering)

- En EN 300 468:

### 5.1 SI table mechanism

The SI specified in the present document and MPEG-2 **PSI tables shall be segmented into one or more sections before being inserted into TS packets.**

- **A section is a syntactic structure that shall be used for mapping all MPEG-2 tables and SI tables specified in the present document, into TS packets.**

These SI syntactic structures conform to the private section syntax defined in ISO/IEC 13818-1 [18].

### 5.1.2 Mapping of sections into Transport Stream (TS) packets

**Sections shall be mapped directly into TS packets.**

.....For a more detailed description of the mechanism and functionality, specifically refer to clause 2.4.4 and annex C of ISO/IEC 13818-1 [18].

- **El Stream type será el 0x05: private\_sections**

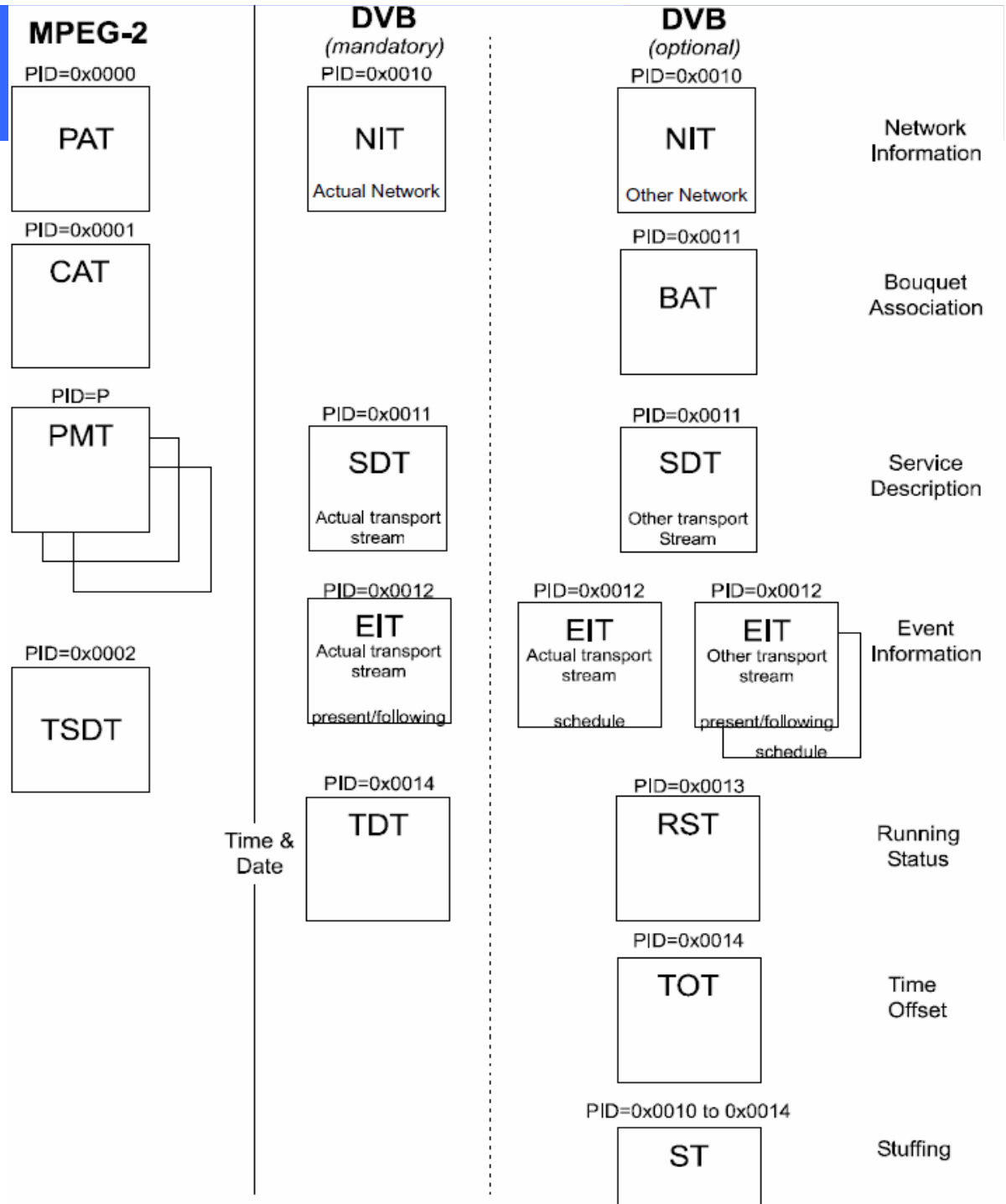
## Introducción

- Antes de estudiar los APIS que nos permiten acceder a esta información, intentemos entender un poco mejor de qué estamos hablando. Recordemos la primera slide de esta primera parte:
  - Para que el receptor sepa, entre otras cosas, a qué canales pertenecen los ES contenidos en un TS, durante el proceso de multiplexación del TS se incluyen ES específicos que contienen esta información: es lo que se llama **Service Information**. Parte de esta información viene del standard MPEG-2 y parte es propia, por ejemplo, de DVB
  - Esta información se incluye en unos tipos de Streams de tipo 0x05 Private Sections.
  - Tablas de información que forman la **System Information**:
    - Program Association table (**PAT**) – definida por el standard MPEG
    - Program Map Table (**PMT**) - definida por el standard MPEG
    - Network Information Table (**NIT**)
    - Conditional Access Table (**CAT**) - definida por el standard MPEG
    - Service Description Table (**SDT**)
    - Event Information Table (**EIT**)
    - Bouquet Association Table (**BAT**)
    - Time and Date Table (**TDT**)
    - Time Offset Table (**TOT**)
  - **PSI**: Juntas, la **PAT**, **PMT**, **CAT** y **NIT** se conocen como **PSI: Program Specific Information (PSI)** y son definidas por Standard MPEG. El resto son específicas de DBV.
  - La información de **PSI** permite la configuración automática del receptor para demultiplexar y decodificar los diferentes Streams de Programas dentro del Multiplex (TS)

# Tablas System Information

- Esquema de Tablas MPEG/DVB

EN 300 468



## Formato de las tablas

- La información que reside en las tablas indicadas o que son parte de las mismas se transmite siguiendo siempre un esquema de formateo común, de manera que los parsers:
  - Saben cómo decodificar cada tipo de información
  - Saben cómo “saltar” la información que no interesa
  - Saben cómo “saltar” la información que no entienden (seguro que hay quien sí sabe)
  - Es un formato que facilita la no emisión de contenido no necesario, de forma que se optimiza el espacio.
  - No es restrictivo en cuanto al orden en que debe enviarse la información, ni siquiera en cuanto a su “colocación” y al igual que en el diseño de BBDD relacionales facilitan la redundancia para optimizar el rendimiento: una tabla se puede colocar en múltiples localizaciones.

¿ pero en qué consiste exactamente ?

## Formato de las tablas

- A nivel lógico el esquema que se suele seguir para ofrecer la información respecto a una funcionalidad es el que sigue:

### Formato de un Descriptor Principal

- Cabecera de la tabla/descriptor: contiene el Código del Descriptor
- Zona con información concreta (y formato concreto) del descriptor
- Bucle con Descriptores de partes que son comunes a todo lo que viene después (haciendo un símil: variables o funciones globales!)
- Bucle con las partes “Principales” de que consta la información, por ejemplo, si la Tabla principal fuese la de la PMT (Program Map Table) cada ciclo podría contener un Descriptor que me diera la referencia a cada Elementary Stream de que consta.
- Dentro de cada Bucle “Principal” podemos encontrarnos a su vez bucles internos con más sets de información que a su vez contienen otros Descriptores

- Ya sabemos cómo pero no qué. Antes de pasar a ver cómo se accede a la información, veamos qué información hay (o al menos una pequeña parte!!)

## **Documentos a tener a mano**

- ETSI EN 300 468  
Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems
- ISO-IEC 13818-1  
Information technology - Generic coding of moving pictures and associated audio information: Systems
- ETSI EN 301 192  
Digital Video Broadcasting (DVB); DVB specification for data broadcasting



## PMT

- Describe para un servicio/canal todos los ES que lo componen indicando **el tipo de cada uno**. Además indica cual de estos ES contiene el **MPEG Program Clock Reference: PCR (sirve para sincronizar el reloj con el deco!)**
- PCR: Program Clock Reference
  - Muy importante para mantener la sincronización del programa y el receptor.
  - Se guarda en una zona opcional de los paquetes del PES.

## PMT

- Formato de la PMT: Program Map Table, (table\_id= 0x02)
- Como véis tenemos un bucle de descriptores Globales y otro de detalle de los elementos que componen la información elemental de la PMT, que consiste básicamente en Streams/Tipo de que consta el Service. (ved iso 13818-1:2000, en 300468 )
- Ved el capítulo App Signalling para analizar el contenido en relación a la transmisión de aplicaciones.
- Ved igualmente el capítulo DSMCC Files...para el contenido en relación a la transmisión de elementos DSMCC.

ISO 13818-1

Table 2-28 – Transport Stream program map section

| Syntax                     | No. of bits | Mnemonic |
|----------------------------|-------------|----------|
| TS_program_map_section() { |             |          |
| table_id                   | 8           | uimsbf   |
| section_syntax_indicator   | 1           | bslbf    |
| '0'                        | 1           | bslbf    |
| reserved                   | 2           | bslbf    |
| section_length             | 12          | uimsbf   |
| program_number             | 16          | uimsbf   |
| reserved                   | 2           | bslbf    |
| version_number             | 5           | uimsbf   |
| current_next_indicator     | 1           | bslbf    |
| section_number             | 8           | uimsbf   |
| last_section_number        | 8           | uimsbf   |
| reserved                   | 3           | bslbf    |
| PCR_PID                    | 13          | uimsbf   |
| reserved                   | 4           | bslbf    |
| program_info_length        | 12          | uimsbf   |
| for (i = 0; i < N; i++) {  |             |          |
| descriptor()               |             |          |
| }                          |             |          |
| for (i = 0; i < N1; i++) { |             |          |
| stream_type                | 8           | uimsbf   |
| reserved                   | 3           | bslbf    |
| elementary_PID             | 13          | uimsbf   |
| }                          |             |          |
| reserved                   | 4           | bslbf    |
| ES_info_length             | 12          | uimsbf   |
| for (i = 0; i < N2; i++) { |             |          |
| descriptor()               |             |          |
| }                          |             |          |
| }                          |             |          |
| CRC_32                     | 32          | rpchof   |
| }                          |             |          |

## NIT

- ¿Qué es una **network**?
  - Se refiere una entidad física (Satélite, Emisor Terrestre...) a través de cuya infraestructura se transmiten una serie de TS.
- Network Information Table (**NIT**)
  - Describe como los TS están organizados y algún otro tipo de información acerca de propiedades físicas. Tiene el Nombre de la Red y su ID. *Este ID es el de la network que está emitiendo ahora el TS pero puede no ser el de la network original si se está re-emitiendo*
  - Ofrece una lista de TS que contiene la network **y los parámetros necesarios para sintonizarlos**.
  - Puede haber dos tipos de **NIT**: **NIT-actual**, que se corresponde con la network actualmente emitiendo, y **NIT-other** que ofrece información acerca de otras networks, de manera que se puedan sintonizar. Cada una tiene su **table\_id**:
    - NIT-actual: 0x40
    - NIT-other: 0x41

## NIT

- Formato de la Network Information Table, (table\_id= 0x40)

**Table 3: Network information section**

| Syntax                          | Number of bits | Identifier |
|---------------------------------|----------------|------------|
| network_information_section() { |                |            |
| table_id                        | 8              | uimsbf     |
| section_syntax_indicator        | 1              | bslbf      |
| reserved_future_use             | 1              | bslbf      |
| reserved                        | 2              | bslbf      |
| section_length                  | 12             | uimsbf     |
| network_id                      | 16             | uimsbf     |
| reserved                        | 2              | bslbf      |
| version_number                  | 5              | uimsbf     |
| current_next_indicator          | 1              | bslbf      |
| section_number                  | 8              | uimsbf     |
| last_section_number             | 8              | uimsbf     |
| reserved_future_use             | 4              | bslbf      |
| network_descriptors_length      | 12             | uimsbf     |
| for(i=0;i<N;i++){               |                |            |
| descriptor()                    |                |            |
| }                               |                |            |
| reserved_future_use             | 4              | bslbf      |
| transport_stream_loop_length    | 12             | uimsbf     |
| for(i=0;i<N;i++){               |                |            |
| transport_stream_id             | 16             | uimsbf     |
| original_network_id             | 16             | uimsbf     |
| reserved_future_use             | 4              | bslbf      |
| transport_descriptors_length    | 12             | uimsbf     |
| for(j=0;j<N;j++){               |                |            |
| descriptor()                    |                |            |
| }                               |                |            |
| }                               |                |            |
| CRC_32                          | 32             | rpchof     |
| }                               |                |            |

EN 300 468

## CA

- Para que el STB sepa cómo descifrar un ES es necesario informarle de los mecanismos de CA. Esto se hace mediante la publicación de la **CAT:Conditional Access Table (siempre en el ES con PID: 0x0001)**, la cual proporciona información acerca de los sistemas CA (cifrado) que se usan dentro de un TS.
- Nos permite definir sistemas de CA como por ejemplo, de los tipos:
  - **Entitlement Control Message (ECM)**: Información de acceso condicional a nivel de Stream
  - **Entitlement Management Message (EMM)**: Información de acceso condicional que indican nivel de autorización o los servicios de decodificadores específicos.
- Sin embargo no sólo en la CAT encontraremos los **Conditional Access Descriptors** (siguiente slide) , también en la **PMT asociados a un ES de un programa**

## CA

- CAT Format , (table\_id= 0x01)
- En el bucle descriptor() se encontrarán los **CA\_descriptor** (siguiente slide)

| Syntax                          | No. of bits | Mnemonic      |
|---------------------------------|-------------|---------------|
| CA_section() {                  |             |               |
| <b>table_id</b>                 | <b>8</b>    | <b>uimsbf</b> |
| <b>section_syntax_indicator</b> | <b>1</b>    | <b>bslbf</b>  |
| '0'                             | <b>1</b>    | <b>bslbf</b>  |
| <b>reserved</b>                 | <b>2</b>    | <b>bslbf</b>  |
| <b>section_length</b>           | <b>12</b>   | <b>uimsbf</b> |
| <b>reserved</b>                 | <b>18</b>   | <b>bslbf</b>  |
| <b>version_number</b>           | <b>5</b>    | <b>uimsbf</b> |
| <b>current_next_indicator</b>   | <b>1</b>    | <b>bslbf</b>  |
| <b>section_number</b>           | <b>8</b>    | <b>uimsbf</b> |
| <b>last_section_number</b>      | <b>8</b>    | <b>uimsbf</b> |
| for (i = 0; i < N; i++) {       |             |               |
| descriptor()                    |             |               |
| }                               |             |               |
| <b>CRC_32</b>                   | <b>32</b>   | <b>rpchof</b> |
| }                               |             |               |

ISO-13818-1

## CA

- Conditional Access Descriptor (descriptor\_tag=9)

ISO 131818-1

| Syntax  | No. of bits                              | Mnemonic  |
|---|--|---|
| <pre>CA_descriptor() {<br/>    descriptor_tag<br/>    descriptor_length<br/>    CA_system_ID<br/>    reserved<br/>    CA_PID<br/>    for (i = 0; i &lt; N; i++) {<br/>        private_data_byte<br/>    }<br/>}</pre> | <p>8<br/>8<br/>16<br/>3<br/>13<br/>8</p> | <p><b>uimsbf</b><br/><b>uimsbf</b><br/><b>uimsbf</b><br/><b>bslbf</b><br/><b>uimsbf</b><br/><b>uimsbf</b></p> |

## CA

- CA\_descriptor. Significado CA\_PID
  - Cuando los **CA Descriptor** están en la **CAT** el CA\_PID indica en qué ES se encuentran los EMMs y el CA se puede usar a nivel GLOBAL del TS
  - Cuando los **CA Descriptor** residen en la **PMT** indican que el ES al que están asociados está encriptado y en qué ES del Service se encuentran los ECMs, y el CA se puede usar a nivel de Service.
  - Los CA también pueden existir según iso131818, en la TSDT: Transport Stream Descriptor Table!!!!

## Bouquets

- ¿ Qué es un Bouquet ?
  - Es una agrupación lógica de Servicios. Se usa por ejemplo para poder identificar un grupo de Servicios propiedad de un mismo emisor, pero que son emitidos por las limitaciones técnicas del multiplex, a través de diferentes Transport Stream.
- La Bouquet Association Table, BAT, Lista los servicios incluidos en un bouquet.
- BAT Format (table\_id= 0x4A)

EN 300 468

Table 4: Bouquet association section

| Syntax                          | Number of bits | Identifier |
|---------------------------------|----------------|------------|
| bouquet_association_section() { |                |            |
| table_id                        | 8              | uimsbf     |
| section_syntax_indicator        | 1              | bslbf      |
| reserved_future_use             | 1              | bslbf      |
| reserved                        | 2              | bslbf      |
| section_length                  | 12             | uimsbf     |
| bouquet_id                      | 16             | uimsbf     |
| reserved                        | 2              | bslbf      |
| version_number                  | 5              | uimsbf     |
| current_next_indicator          | 1              | bslbf      |
| section_number                  | 8              | uimsbf     |
| last_section_number             | 8              | uimsbf     |
| reserved_future_use             | 4              | bslbf      |
| bouquet_descriptors_length      | 12             | uimsbf     |
| for(i=0;i<N;i++){               |                |            |
| descriptor()                    |                |            |
| }                               |                |            |
| reserved_future_use             | 4              | bslbf      |
| transport_stream_loop_length    | 12             | uimsbf     |
| for(i=0;i<N;i++){               |                |            |
| transport_stream_id             | 16             | uimsbf     |
| original_network_id             | 16             | uimsbf     |
| reserved_future_use             | 4              | bslbf      |
| transport_descriptors_length    | 12             | uimsbf     |
| for(j=0;j<N;j++){               |                |            |
| descriptor()                    |                |            |
| }                               |                |            |
| }                               |                |            |
| CRC_32                          | 32             | rpchof     |
| }                               |                |            |

## SDT

- Service Description Table (SDT). Es obligatoria. Ofrece información respecto a los Services existentes en el TS
- También ofrece información orientada al usuario como el nombre de cada servicio y una descripción, su estado (running/not running/starting in a few seconds), control parental...en fin, información orientada al usuario.
- Puede ofrecer información respecto a Servicios del TS actual o respecto a Servicios que se encuentran dentro de otros TS. Al igual que ocurría con el NIT, tenemos dos códigos distintos para el table\_id
  - SDT-actual: 0x42
  - SDT-other: 0x46

Table 5: Service description section

| Syntax                          | Number of bits | Identifier |
|---------------------------------|----------------|------------|
| service_description_section() { |                |            |
| table_id                        | 8              | uimsbf     |
| section_syntax_indicator        | 1              | bslbf      |
| reserved_future_use             | 1              | bslbf      |
| reserved                        | 2              | bslbf      |
| section_length                  | 12             | uimsbf     |
| transport_stream_id             | 16             | uimsbf     |
| reserved                        | 2              | bslbf      |
| version_number                  | 5              | uimsbf     |
| current_next_indicator          | 1              | bslbf      |
| section_number                  | 8              | uimsbf     |
| last_section_number             | 8              | uimsbf     |
| original_network_id             | 16             | uimsbf     |
| reserved_future_use             | 8              | bslbf      |
| for (i=0;i<N;i++){              |                |            |
| service_id                      | 16             | uimsbf     |
| reserved_future_use             | 6              | bslbf      |
| EIT_schedule_flag               | 1              | bslbf      |
| EIT_present_following_flag      | 1              | bslbf      |
| running_status                  | 3              | uimsbf     |
| free_CA_mode                    | 1              | bslbf      |
| descriptors_loop_length         | 12             | uimsbf     |
| for (j=0;j<N;j++){              |                |            |
| descriptor()                    |                |            |
| }                               |                |            |
| }                               |                |            |
| CRC_32                          | 32             | rpchof     |
| }                               |                |            |

## SDT

- Mediante el **service\_descriptor** nos indica que **Tipo de Servicio** es cada uno.
- Notad:
  - 0x01: Digital Tv
  - 0x02: Digital Radio
  - 0x10: DVB MHP Service
  - 0x0C: Data Broadcast Service

| Syntax                       | Number of bits | Identifier |
|------------------------------|----------------|------------|
| service_descriptor(){        |                |            |
| descriptor_tag               | 8              | uimsbf     |
| descriptor_length            | 8              | uimsbf     |
| service_type                 | 8              | uimsbf     |
| service_provider_name_length | 8              | uimsbf     |
| for (i=0;i<N;I++){           |                |            |
| char                         | 8              | uimsbf     |
| }                            |                |            |
| service_name_length          | 8              | uimsbf     |
| for (i=0;i<N;I++){           |                |            |
| Char                         | 8              | uimsbf     |
| }                            |                |            |
| }                            |                |            |

| service_type | Description   |
|--------------|---|
| 0x00         | reserved for future use                                     |
| 0x01         | digital television service (see note 1)                     |
| 0x02         | digital radio sound service (see note 2)                    |
| 0x03         | Teletext service  |
| 0x04         | NVOD reference service (see note 1)                         |
| 0x05         | NVOD time-shifted service (see note 1)                      |
| 0x06         | mosaic service  |
| 0x07         | reserved for future use                                     |
| 0x08         | reserved for future use                                     |
| 0x09         | reserved for future use                                     |
| 0x0A         | advanced codec digital radio sound service                  |
| 0x0B         | advanced codec mosaic service                               |
| 0x0C         | data broadcast service                                      |
| 0x0D         | reserved for Common Interface Usage (CENELEC EN 50221 [38]) |
| 0x0E         | RCS Map (see EN 301 790 [7])                                |
| 0x0F         | RCS FLS (see EN 301 790 [7])                                |
| 0x10         | DVB MHP service   |
| 0x11         | MPEG-2 HD digital television service                        |
| 0x12 to 0x15 | reserved for future use                                     |
| 0x16         | advanced codec SD digital television service                |
| 0x17         | advanced codec SD NVOD time-shifted service                 |
| 0x18         | advanced codec SD NVOD reference service                    |
| 0x19         | advanced codec HD digital television service                |
| 0x1A         | advanced codec HD NVOD time-shifted service                 |
| 0x1B         | advanced codec HD NVOD reference service                    |
| 0x1C to 0x7F | reserved for future use                                     |
| 0x80 to 0xFE | user defined  |
| 0xFF         | reserved for future use                                     |

NOTE 1: MPEG-2 SD material should use this type.

NOTE 2: MPEG-1 Layer 2 audio material should use this type.

## EIT

- La Event Information Table (**EIT**) proporciona los Programas (Shows) y sus horarios que van a aparecer en los determinados canales (Servicios)
- Existen dos tipos de **EIT**:
  - **EIT present/following**: cada TS debe ofrecer para todos los canales que contiene la información de los programas que están en este momento en el aire así como los siguientes a estos.
    - Al igual que ocurre con NIT y con SDT, se puede ofrecer información respecto a otros TS distintos del sintonizado, por lo que tenemos 2 table\_id distintos:
      - 0x4E: **EIT present/following-actual**
      - 0x4F: **EIT present/following-other**
  - **EIT Schedule**: opcionalmente cada TS puede ofrecer para todos los canales que contiene la información de los programas futuros. Igualmente tenemos dos códigos de tablas:
    - 0x50-0x5F: **EIT Schedule-actual**
    - 0x60-0x6F: **EIT Schedule-other**

## EIT

- EIT Format (table\_id= 0x4E/0x4F/0x50-0x5F/0x60-0x6F)

**Table 7: Event information section**

| Syntax                        | Number of bits | Identifier |
|-------------------------------|----------------|------------|
| event_information_section() { |                |            |
| table_id                      | 8              | uimsbf     |
| section_syntax_indicator      | 1              | bslbf      |
| reserved_future_use           | 1              | bslbf      |
| reserved                      | 2              | bslbf      |
| section_length                | 12             | uimsbf     |
| service_id                    | 16             | uimsbf     |
| reserved                      | 2              | bslbf      |
| version_number                | 5              | uimsbf     |
| current_next_indicator        | 1              | bslbf      |
| section_number                | 8              | uimsbf     |
| last_section_number           | 8              | uimsbf     |
| transport_stream_id           | 16             | uimsbf     |
| original_network_id           | 16             | uimsbf     |
| segment_last_section_number   | 8              | uimsbf     |
| last_table_id                 | 8              | uimsbf     |
| for(i=0;i<N;i++) {            |                |            |
| event_id                      | 16             | uimsbf     |
| start_time                    | 40             | bslbf      |
| duration                      | 24             | uimsbf     |
| running_status                | 3              | uimsbf     |
| free_CA_mode                  | 1              | bslbf      |
| descriptors_loop_length       | 12             | uimsbf     |
| for(i=0;i<N;i++) {            |                |            |
| descriptor()                  |                |            |
| }                             |                |            |
| }                             |                |            |
| CRC_32                        | 32             | rpchof     |
| }                             |                |            |

EN 300468

## Otras

- **TSDT**: Transport Stream Descriptor Table, PID = 0x0002, table\_id = 0x03
- **TDT**: Time and Date Table. Lleva el UTC (Universal Clock Time). PID= 0x0014, table\_id = 0x70
- **TOT**: Time Offset Table. Lleva la UTC y el offset local. PID= 0x0014, Table\_id = 0x73
- **RST**: Running Status Table: Estado de un evento. PID= 0x0013, table\_id=0x71

## TSDT

- Con respecto a la TSDT: Es opcional (ver gráfica de tablas inicial) pero puede incluir descriptores de los tipos siguientes

Table 2-39 – Program and program element descriptors

| descriptor_tag | Identification                          | descriptor_tag | Identification                                |
|----------------|---|----------------|---|
| 0              | Reserved                                | 16             | Smoothing_buffer_descriptor                   |
| 1              | Reserved                                | 17             | STD_descriptor                                |
| 2              | video_stream_descriptor                 | 18             | IBP_descriptor                                |
| 3              | audio_stream_descriptor                 | 19-26          | Defined in ISO/IEC 13818-6                    |
| 4              | hierarchy_descriptor                    | 27             | MPEG-4_video_descriptor                       |
| 5              | registration_descriptor                 | 28             | MPEG-4_audio_descriptor                       |
| 6              | data_stream_alignment_descriptor        | 29             | IOD_descriptor                                |
| 7              | target_background_grid_descriptor       | 30             | SL_descriptor                                 |
| 8              | Video_window_descriptor                 | 31             | FMC_descriptor                                |
| 9              | CA_descriptor                           | 32             | External_ES_ID_descriptor                     |
| 10             | ISO_639_language_descriptor             | 33             | MuxCode_descriptor                            |
| 11             | System_clock_descriptor                 | 34             | FmxBufferSize_descriptor                      |
| 12             | Multiplex_buffer_utilization_descriptor | 35             | MultiplexBuffer_descriptor                    |
| 13             | Copyright_descriptor                    | 36-63          | ITU-T Rec. H.222.0   ISO/IEC 13818-1 Reserved |
| 14             | Maximum_bitrate_descriptor              | 64-255         | User Private                                  |
| 15             | Private_data_indicator_descriptor       |                |   |

ISO 13818-1



- Códigos de PID de los ES para localizar Tablas Principales y códigos de table\_id (descriptors)

**Table 1: PID allocation for SI**

| Table                         | PID value        |
|-------------------------------|------------------|
| PAT                           | 0x0000           |
| CAT                           | 0x0001           |
| TSDT                          | 0x0002           |
| reserved                      | 0x0003 to 0x000F |
| NIT, ST                       | 0x0010           |
| SDT, BAT, ST                  | 0x0011           |
| EIT, ST CIT (TS 102 323 [14]) | 0x0012           |
| RST, ST                       | 0x0013           |
| TDT, TOT, ST                  | 0x0014           |
| network synchronization       | 0x0015           |
| RNT (TS 102 323 [14])         | 0x0016           |
| reserved for future use       | 0x0017 to 0x001B |
| inband signalling             | 0x001C           |
| measurement                   | 0x001D           |
| DIT                           | 0x001E           |
| SIT                           | 0x001F           |

- Códigos de PID de los ES para localizar Tablas Principales y códigos de table\_id (descriptors) (y 2)

Table 2: Allocation of table\_id values

| Value        | Description  |
|--------------|--|
| 0x00         | program_association_section  |
| 0x01         | conditional_access_section   |
| 0x02         | program_map_section  |
| 0x03         | transport_stream_description_section                                   |
| 0x04 to 0x3F | reserved   |
| 0x40         | network_information_section - actual_network                           |
| 0x41         | network_information_section - other_network                            |
| 0x42         | service_description_section - actual_transport_stream                  |
| 0x43 to 0x45 | reserved for future use  |
| 0x46         | service_description_section - other_transport_stream                   |
| 0x47 to 0x49 | reserved for future use  |
| 0x4A         | bouquet_association_section  |
| 0x4B to 0x4D | reserved for future use  |
| 0x4E         | event_information_section - actual_transport_stream, present/following |
| 0x4F         | event_information_section - other_transport_stream, present/following  |
| 0x50 to 0x5F | event_information_section - actual_transport_stream, schedule          |
| 0x60 to 0x6F | event_information_section - other_transport_stream, schedule           |
| 0x70         | time_date_section  |
| 0x71         | running_status_section   |
| 0x72         | stuffing_section   |
| 0x73         | time_offset_section  |
| 0x74         | application information section (TS 102 812 [16])                      |
| 0x75         | container section (TS 102 323 [14])                                    |
| 0x76         | related content section (TS 102 323 [14])                              |
| 0x77         | content identifier section (TS 102 323 [14])                           |
| 0x78         | MPE-FEC section (EN 301 192 [4])                                       |
| 0x79         | resolution notification section (TS 102 323 [14])                      |
| 0x79 to 0x7D | reserved for future use  |
| 0x7E         | discontinuity_information_section                                      |
| 0x7F         | selection_information_section  |
| 0x80 to 0xFE | user defined   |
| 0xFF         | reserved   |

- Las tablas principales de la emisión (MPEG/ DVB) tienen reservados unos PID concretos, de manera que los STBs van a “tiro hecho” a leer estos ES (resumen de lo anterior)

| Obligatorias, MPEG | Obligatorias, DVB                | Opcionales, DVB  | PID Reservado |
|--------------------|----------------------------------|--|---------------|
| PAT                |                                  |  | 0x0000        |
| PMT                |                                  |  |               |
| CAT                |                                  |  | 0x0001        |
|                    | NIT-actual (del TS seleccionado) | NIT-other ( en otros TS existentes)                            | 0x0010        |
|                    | SDT-actual (del TS seleccionado) | SDT-other (en otros TS existentes)                             | 0x0011        |
|                    | EIT-present/following (actual)   | EIT-schedule (actual y other)<br>EIT-present/following (other) | 0x0012        |
|                    | TDT                              |  | 0x0014        |
|                    |                                  | TOT  | 0x0014        |
|                    |                                  | BAT  | 0x0011        |

- ¿ Cómo acceder a esta información ? Hay tres modos: **MHP APIs, Java TV APIs y Section Filtering**
- MHP APIs funcionan con DVB Standards mientras que los JavaTV son más genéricos y valen también para ATSC (buena parte de EEUU); pero lo más importante es que ambos ofrecen lo mismo.
- El mecanismo de Section Filtering ofrece un API de acceso genérico a todo tipo de información; se usa para acceder a los bytes de los descriptores que no están representados como componentes de negocio.
- **Un detalle:** para acceder a la información de Aplicaciones no se usan estos APIs sino los descritos en el capítulo “Finding & Launching Applications”.

## DVB SI API

- ¿ donde ? : **org.dvb.si**
  - Nota: Al igual que ocurre con los ficheros, recordemos que estamos en un sistema de Broadcasting y que la información de la SI no está siempre on-line. Hay tablas de información (p.e. PAT) que incluso se cachearán en el deco, pero en cualquier caso los modelos de actuación tendrán una componente asíncrona importante.
- Elemento Principal: La BBDD
  - org.dvb.si.SIDatabase** extends **SIInformation**

## SI Classes

- Veamos las clases que nos ofrecerán la información de la SI. Como vemos representan muchos de los objetos de información existentes en las Tablas de la SI, así por ejemplo, existe un **SIService** que se extrae de la SDT y un **PMTService** que se extrae de la PMT: aunque sólo exista un Service tenemos dos entidades distintas que nos ofrecen información sobre él
  - **SINetwork**: Representa una sub-table de la NIT describiendo una network
  - **SIBouquet**: Representa una sub-tabla de la Bouquet Association Table (BAT) describiendo un bouquet particular.
  - **SITransportStream**: Interface base para ofrecer la información de transport streams.
    - Los métodos de **SIDatabase** y de **SINetwork** que ofrecen **SITransportStreams** buscan en la **NIT** y devuelven el tipo **SITransportStreamNIT**
    - Los métodos de **SIBouquet** que ofrecen **SITransportStreams** buscan en la **BAT** y devuelven el tipo **SITransportStreamBAT**
  - **SITransportStreamBAT**: hereda de **SITransportStream** añadiendo `public int getBouquetID()`
  - **SITransportStreamNIT**: hereda de **SITransportStream** añadiendo `public int getNetworkID()`

## SI Classes

- **SIService**: Representa un Service emitido por un TS. Lo que se le pida a este Interface se extraerá de la **SDT**.
- **PMTService**: Representa un Service emitido por un TS, pero cuya información se ha leído de la **PMT**
- **SITransportStreamDescription**: Representa la TSDT (Transport Stream Description Table)
- **PMTElementaryStream**: Este interface representa un ES de un Servicio. Se extrae su información de la PMT.
- **SIEvent**: Representa un evento particular dentro de un Servicio
- **SITime**: Representa la Time and Date Table (**TDT**) y la opcional Time Offset Table (**TOT**)

Ahora obtengámoslas!!

## Obteniendo la SIDatabase

- Las SIDatabase representan la información de SI de cada sistema de conexión de nuestro STB: cable, terrestre, satélite...
- Tenemos una SIDatabase por Tuner!!! (Sintonizador). Como casi siempre: un singleton.

```
public static org.dvb.si.SIDatabase[] getSIDatabase()
```

- Mediante la clase **org.dvb.net.tuning.DvbNetworkInterfaceSIUtil** podemos saber qué Tuner, es decir, tipo `org.davic.net.tuning.NetworkInterface`, está usando nuestra SIDatabase.

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

- Ya tenemos la SIDatabase. A partir de ella obtenemos el resto de la información

## Obteniendo la SINetwork (from NIT)

- SIDatabase ofrece los siguientes métodos para obtener SINetwork
  - public **org.dvb.si.SIRequest retrieveActualSINetwork**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
    - Devuelve **la Network actual**, es decir, aquella a la que pertenece el TS actualmente sintonizado por el SIDatabase
  - public **org.dvb.si.SIRequest retrieveSINetworks**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int networkId, short[] someDescriptorTags)
    - Para **networkId=-1** devuelve las networks que aparezcan en el **NIT-actual** o **NIT-other** emitidas por el TS actual.

## Obteniendo la SInetwork (from NIT)

- Veamos los parámetros (la mayoría se usarán en muchas llamadas)
  - **retrieveMode** = org.dvb.si.SIInformation.[  
FROM\_CACHE\_ONLY,  
FROM\_CACHE\_OR\_STREAM,  
FROM\_STREAM\_ONLY ]. Gestión de caché del STB
  - **org.dvb.si.SIRetrievalListener**: el listener al que se le proporcionará la información.
  - **appData**: lo podemos usar para identificar nuestras llamadas, por ejemplo, pues se incluye en el callback al listener.
  - **someDescriptorTags**: si estamos interesados en que recupere tablas marcadas con algunos descriptores específicos o **new short[]{-1}** si en todos.

## Inciso: modelo de petición asíncrono de información

- Como observamos en el método, al margen de su funcionalidad en sí, devuelve un objeto del tipo **SIRequest**, y además se le pasa un objeto del tipo **SIRetrievalListener**
- **SIRequest** se devuelve de inmediato y nos permite por un lado cancelar la petición realizada y por otro saber si la información existe en caché y se va a recuperar de ella o no.

```
public boolean isAvailableInCache()
```

```
public boolean cancelRequest(): true si se acepta la cancelación.
```

- El listener **SIRetrievalListener** recibirá un evento de tipo **SIRetrievalEvent** con la información solicitada (o no) y los datos pasados en la solicitud si es que se pasó alguno, de manera que podamos relacionar petición-respuesta.

```
public interface SIRetrievalListener extends java.util.EventListener {  
    public void postRetrievalEvent(SIRetrievalEvent event);  
}
```

## Inciso: modelo de petición asíncrono de información

- **SIRetrievalEvent** es la **clase base** de los eventos y ofrece los dos métodos siguientes:
  - public Object **getAppData()**
    - Devuelve el appData que le pasamos en la petición
  - public Object **getSource()**
    - Ofrece el SIRequest devuelto inmediatamente al hacer la petición
- En realidad los **SIRetrievalEvent** que se pueden recibir son los siguientes tipos:
  - **SISuccessfulRetrieveEvent**: método para obtener la info: public **SIiterator getResult()**
  - **SIRequestCancelledEvent**: cuando se solicitó cancelación y esta se ha producido.
  - **SINotInCacheEvent**: si se pidió de caché y no estaba
  - **SIObjectNotInTableEvent**: objeto solicitado no existe, (no hay objeto para el descriptor)
  - **SITableUpdatedEvent**: la información se ha actualizado y ya no es coherente con la existente en el momento de la petición. Se debería de volver a solicitar la información desde el principio.
  - **SITableNotFoundEvent**

## SINetwork API

- Veamos qué ofrece. **Parte común**, la del Interface del que hereda y que es común a todos los vistos: **SIInformation**. De todos modos los métodos de los descriptores serán sobre-escritos para aportar las especificidades de cada Interface (ved la doc del API Java)
  - public SIDatabase **getSIDatabase()**;
    - La SIDatabase a la que pertenece.
  - public org.davic.mpeg.TransportStream **getDataSource()**;
    - Transport Stream en el que estamos
  - public java.util.Date **getUpdateTime()**;
    - Momento en que la información proporcionada en el objeto se actualizó por última vez
  - public short[] **getDescriptorTags()**;
    - Si la sub-tabla NIT en la que se basa esta Network se compone de varias secciones, devuelve los descriptores de estas en el mismo orden en que son broadcasted. Método que sobre-escibe uno de SIInformation. **Es Síncrono**.

## SINetwork API

- Veamos qué ofrece. **Parte común**

- public SIRequest **retrieveDescriptors**(short retrieveMode, Object appData, SIRetrievalListener listener);
  - Idem a getDescriptorTags() pero asíncrono. Sobre-escibe SIInformation.
- public SIRequest **retrieveDescriptors**(short retrieveMode, Object appData, SIRetrievalListener listener, **short[]** someDescriptorTags)
  - Idem al anterior salvo que se filtrarán los descriptores devueltos: sólo se obtendrán aquellos que contengan su tag incluido en la lista pasada. New short[]{-1} si queremos todos (equivalente al anterior). Sobre-escibe uno de SIInformation

## SINetwork API

- Veamos qué ofrece. **Parte propia**
  - public int **getNetworkID()**
    - El Id de la Network
  - public String **getName();**
  - public String **getShortNetworkName();**
    - El Nombre de la Network, y en formato Corto. (mirad JavaDoc para detalle multi-idioma)
  - public SIRequest **retrieveSITransportStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - Devuelve un SIIterator con los Transport Streams que envía la Network. Los objetos devueltos serán del tipo **SITransportStreamNIT** y se obtienen de la NIT.

## Ejercicios Bloque DVBSI-1

## Obteniendo SITransportStream

- Disponemos de los siguientes métodos para obtener SITransportStream
  - public org.dvb.si.SIRequest **retrieveActualSITransportStream**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve un **SITransportStreamNIT**, es decir, aquel al que actualmente está sintonizado el SIDatabase a través del Network Interface
  - public SIRequest **retrieveSITransportStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - **SINetwork**. Devuelve un **SITransportStreamNIT**, el correspondiente al que lleva la NetWork
  - public SIRequest **retrieveSIBouquetTransportStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIBouquet**. Devuelve los **SITransportStreamBAT** referenciados en la BAT. La información es la que aparece en esta.
- Veamos los parámetros: ídem a SINetwork

## SITransportStream API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public int **getNetworkID()**
    - El Id de la Network
  - public int **getOriginalNetworkID()**
    - El Id de la Network Original
  - public int **getTransportStreamID();**
    - El Id del TS

## SITransportStream API

- **Parte propia:**
  - public org.davic.net.dvb.DvbLocator **getDvbLocator()**
    - El Locator del Transport Stream
  - public SIRequest **retrieveSIServices**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - Devuelve un SIIterator con los SIService que ofrece este TS.

## Ejercicios Bloque DVBSI-2

## Obteniendo SIBouquet

- Servicios de un mismo “propietario” o relacionados por algún criterio que son transmitidos en diferentes redes/transportStreams
- Disponemos de los siguientes métodos para obtener SIBouquet
  - public org.dvb.si.SIRequest **retrieveSIBouquets**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int bouquetId, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve los SIBouquet encontrados en la **BAT** (con bouquetId=-1)
- Veamos los parámetros: ídem a SINetwork

## SIBouquet API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public int **getBouquetID()**;
    - El Id del Bouquet
  - public String **getName()**, public String **getShortBouquetName()**
    - El nombre, y en formato corto (ETR 211 without emphasis marks)
  - public org.davic.net.dvb.DvbLocator[] **getSIServiceLocators()**
    - Locators que identifican a los Services del Bouquet

## SIBouquet API

- **Parte propia:**
  - public SIRequest **retrieveSIBouquetTransportStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - Devuelve un SIIterator con los SITransportStreamBAT referenciados en la BAT

## Ejercicios Bloque DVBSI-3

## Obteniendo SITransportStreamDescription. TSDT

- Disponemos de los siguientes métodos para obtener SITransportStreamDescription
  - public org.dvb.si.SIRequest **retrieveSITransportStreamDescription**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve un objeto **SITransportStreamDescription** correspondiente a la TSDT del TS Actual
- Veamos los parámetros: ídem a SINetwork

## SITransportStreamDescription API

- **Parte común:** ídem a SINetwork
- **Parte propia:** Ninguna.

## Ejercicios Bloque DVBSI-4

## Obteniendo SIService

- Disponemos de los siguientes métodos para obtener SIService
  - public org.dvb.si.SIRequest **retrieveActualSIServices**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve los SIService actuales, los correspondientes al TS Actual sintonizado.
  - public org.dvb.si.SIRequest **retrieveSIServices**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int originalNetworkId, int transportStreamId, int serviceId, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve los SIService pero filtrados para una networkID o para un TS (-1 si no queremos estos filtros), o queremos que sea un Service concreto (-1 ídem). Si no hay filtros obtiene todos los SIService de los que disponga información.
  - public org.dvb.si.SIRequest **retrieveSIService**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve el SIService correspondiente al locator

## Obteniendo SIService

- Disponemos de los siguientes métodos para obtener SIService
  - public SIRequest **retrieveSIServices**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - **SITransportStream**. Devuelve un SIIterator con los SIService que ofrece este TS.
  - public SIRequest **retrieveSIService**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIEvent**. Devuelve un SIService que representa al Service al cual pertenece el Event
- Veamos los parámetros: ídem a SINetwork

## SIService API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public org.davic.net.dvb.DvbLocator **getDvbLocator()**;
    - Locator del Service
  - public int **getOriginalNetworkID()**;
    - Original Network ID
  - public int **getTransportStreamID()**;
    - Transport Stream ID
  - public int **getServiceID()**;
    - Service ID

## SIService API

- **Parte propia:**
  - public short **getServiceType();**
    - Devuelve el tipo del Service. Ver SIServiceType en siguiente slide.
  - public String **getName()**, public String **getShortServiceName();**
    - Nombre del Service y corto
  - public String **getProviderName()**, **getShortProviderName();**
    - Nombre del Proveedor del Servicio y Corto
  - public boolean **getEITScheduleFlag();**
    - Si true entonces existe información de **Schedule** (Programación)
  - public boolean **getEITPresentFollowingFlag();**
    - Si true entonces existe información de **present y/o Following Event**

## SIService API

- **Parte propia:**

- SIServiceType. Interesante el tipo que aparece como tipo de Canal!!!

```
public interface SIServiceType{
    public final static short UNKNOWN = -1;
    public final static short DIGITAL_TELEVISION = 0x01;
    public final static short DIGITAL_RADIO_SOUND = 0x02;
    public final static short TELETEXT = 0x03;
    public final static short NVOD_REFERENCE = 0x04;
    public final static short NVOD_TIME_SHIFTED = 0x05;
    public final static short MOSAIC = 0x06;
    public final static short PAL = 0x07;
    public final static short SECAM = 0x08;
    public final static short D_D2_MAC = 0x09;
    public final static short FM_RADIO = 0x0A;
    public final static short NTSC = 0x0B;
    public final static short DATA_BROADCAST = 0x0C; //MPE
    public final static short MHP_APPLICATION = 0x10;
}
```



## Un recordatorio respecto a los tipos de Services:

- Un apunte respecto a Canales de Aplicaciones: **0x10: DVB MHP Service**. Strong 5510 MHP 1.1.2



## SIService API

- **Parte propia:**

- public byte **getRunningStatus()**;

- Devuelve el **RunningStatus** del Service:

```
public interface SIRunningStatus {  
    public static final byte UNDEFINED          = 0;  
    public static final byte NOT_RUNNING        = 1;  
    public static final byte STARTS_IN_A_FEW_SECONDS = 2;  
    public static final byte PAUSING            = 3;  
    public static final byte RUNNING           = 4;  
}
```

- public boolean **getFreeCAMode()**;

- Si false ninguno de los componentes de este Service están **encriptados**

- public SIRequest **retrievePresentSIEvent**(short retrieveMode, Object appData, SIRetrievalListener listener, short[]someDescriptorTags)

- Devuelve el **SIEvent** actual definido en la **EIT-present/following**

## SIService API

- **Parte propia:**

- public SIService **retrieveFollowingSIEvent**(short retrieveMode, Object appData, SIServiceListener listener, short[] someDescriptorTags)
  - Devuelve el **SIEvent** siguiente definido en la EIT-present/following
- public SIService **retrieveScheduledSIEvents**(short retrieveMode, Object appData, SIServiceListener listener, short[] someDescriptorTags, java.util.Date startTime, java.util.Date endTime)
  - Devuelve los **SIEvent** Programados en el Service incluidos en la EIT-Schedule que se encuentren en un lapso determinado de tiempo (en UTC) ( $\leq$ SIEvent.getStartTime  $<$ end)
- public SIService **retrievePMTService**(short retrieveMode, Object appData, SIServiceListener listener, short[] someDescriptorTags)
  - Devuelve el **PMTService** correspondiente a este service. Recordemos que este se lee de la **SDT**

## Ejercicios Bloque DVBSI-5

## Obteniendo PMTService

- Disponemos de los siguientes métodos para obtener PMTService
  - public org.dvb.si.SIRequest **retrievePMTServices**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int serviceId, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve los PMTService actuales, los correspondientes al TS Actual sintonizado. Se puede especificar un service (-1 TODOS)
  - public org.dvb.si.SIRequest **retrievePMTService**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve el PMTService correspondiente al Locator
  - public SIRequest **retrievePMTService**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
    - **SIService**. Devuelve un PMTService que representa al Service
- Veamos los parámetros: ídem a SINetwork

## PMTService API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public org.davic.net.dvb.DvbLocator **getDvbLocator()**;
    - Devuelve el Locator del Service
  - public int **getOriginalNetworkID()**;
    - Original Network ID
  - public int **getTransportStreamID()**;
    - Transport Stream ID
  - public int **getServiceID()**
    - Service ID

## PMTService API

- **Parte propia:**
  - public int **getPcrPid()**;
    - PID donde reside el PCR
  - public SIRequest **retrievePMTElementaryStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] somePMTDescriptorTags)
    - Devuelve las instancias de PMTElementaryStream que representan los ES que componen este Service

## Ejercicios Bloque DVBSI-6

## Obteniendo PMTElementaryStream

- Disponemos de los siguientes métodos para obtener **PMTElementaryStream**
  - public org.dvb.si.SIRequest **retrievePMTElementaryStreams**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, int serviceId, int componentTag, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve los PMTElementaryStream de un Servicio del TS Actual sintonizado. Se puede (componentTag= -1 TODOS)
  - public org.dvb.si.SIRequest **retrievePMTElementaryStreams**(short retrieveMode, java.lang.Object appData, org.dvb.si.SIRetrievalListener listener, org.davic.net.dvb.DvbLocator dvbLocator, short[] someDescriptorTags)
    - **SIDatabase**. Devuelve el PMTElementaryStream correspondiente al Locator
  - public SIRequest **retrievePMTElementaryStreams**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] somePMTDescriptorTags)
    - **PMTService**. Devuelve los PMTElementaryStream del Servicio
- Veamos los parámetros: ídem a SINetwork

## PMTElementaryStream API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public org.davic.net.dvb.DvbLocator **getDvbLocator()**;
    - Locator del Stream
  - public int **getOriginalNetworkID()**;
    - Original Network ID
  - public int **getTransportStreamID()**;
    - Transport Stream ID
  - public int **getServiceID()**;
    - Service ID

## PMTElementaryStream API

- **Parte propia:**

- public int **getComponentTag()**;
  - Component Tag o -2 si no tiene.

- public byte **getStreamType()**;
  - Stream Type. Ver PMTStreamType

```
public interface PMTStreamType {  
    public static final byte MPEG1_VIDEO = 1;  
    public static final byte MPEG2_VIDEO = 2;  
    public static final byte MPEG1_AUDIO = 3;  
    public static final byte MPEG2_AUDIO = 4;  
}
```

- public short **getElementaryPID()**;
  - PID

## Ejercicios Bloque DVBSI-7

## Obteniendo SIEvent

- Disponemos de los siguientes métodos para obtener SIEvent
  - public SIREquest **retrievePresentSIEvent**(short retrieveMode, Object appData, SIREtrievalListener listener, short[] someDescriptorTags)
    - **SIService**. Devuelve el **SIEvent** actual definido en la EIT-present/following
  - public SIREquest **retrieveFollowingSIEvent**(short retrieveMode, Object appData, SIREtrievalListener listener, short[] someDescriptorTags)
    - **SIService**. Devuelve el **SIEvent** siguiente definido en la EIT-present/following
  - public SIREquest **retrieveScheduledSIEvents**(short retrieveMode, Object appData, SIREtrievalListener listener, short[] someDescriptorTags, java.util.Date startTime, java.util.Date endTime)
    - **SIService**. Devuelve los **SIEvent** Programados en el Service incluidos en la EIT-Schedule que se encuentren en un lapso determinado de tiempo (en UTC) (<=SIEvent.getStartTime <end)
- Veamos los parámetros: ídem a SINetwork

## SIEvent API

- **Parte común:** ídem a SINetwork
- **Parte propia:**
  - public org.davic.net.dvb.DvbLocator **getDvbLocator()**;
    - Event Locator
  - public int **getOriginalNetworkID()**;
    - Original Network ID
  - public int **getTransportStreamID()**;
    - Transport Stream ID
  - public int **getServiceID()**;
    - Service ID
  - public int **getEventID()**;
    - Event ID

## SIEvent API

- **Parte propia:**

- public java.util.Date **getStartTime()**;
  - UTC Start
- public long **getDuration()**;
  - Seconds Duration
- public byte **getRunningStatus()**;
  - Running Status
- public boolean **getFreeCAMode()**;
  - false significa que ninguno de de los componentes de este event están encriptados
- public String **getName()**;
  - Event name

## SIEvent API

- **Parte propia:**

- public String **getShortEventName()**;
  - Short Event Name
- public String **getShortDescription()**;
  - Descripción corta
- public byte[] **getLevel1ContentNibbles()**;
  - Nibbles de nivel 1 (ved api)
- public byte[] **getContentNibbles()**;
  - Nibbles del evento (ved api)
- public SIRequest **retrieveSIService**(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)
  - SIService en el que se retransmite el Evento

## Ejercicios Bloque DVBSI-8

## Acceso a los Descriptors

- Hasta ahora hemos visualizado la información de alto nivel tipificada en clases de **org.dvb.si** pero podría interesarnos acceder a un determinado contenido de un Descriptor. **Mecanismo:** mediante los métodos de SIInformation:
  - `public short[] getDescriptorTags();`
    - Devuelve los descriptors de esta SIInformation en el mismo orden en que son broadcasted. Método que sobre-escribe cada uno de SIInformation. Es Síncrono.
  - `public SIRequest retrieveDescriptors(short retrieveMode, Object appData, SIRetrievalListener listener)`
    - Idem a `getDescriptorTags()` pero asíncrono.
  - `public SIRequest retrieveDescriptors(short retrieveMode, Object appData, SIRetrievalListener listener, short[] someDescriptorTags)`
    - Idem al anterior salvo que se filtrarán los descriptors devueltos: sólo se obtendrán aquellos que contengan su tag incluido en la lista pasada. `New short[]{-1}` si queremos todos (equivalente al anterior).

## Acceso a los Descriptors

- Los métodos retrieveX devolverán objetos del tipo:

```
public class Descriptor{  
    public short getTag()  
    public short getContentLength()  
    public byte getByteAt(int index)  
    public byte[] getContent()  
}
```

## Ejercicios Bloque DVBSI-9

## DVB SI Listeners

- Es posible “escuchar” eventos de cambios en las diferentes tablas del sistema; de esta manera podemos saber, por ejemplo, que hay nuevo ángulo de vídeo. Los métodos se encuentran en **SI Database** y son bastante Obvios:
  - public void **addBouquetMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int bouquetId)
    - Eventos en la configuración del Bouquet
  - public void **addEventPresentFollowingMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId)
    - Eventos de cambios en la configuración de la EIT de events following y present
  - public void **addEventScheduleMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId, java.util.Date startTime, java.util.Date endTime)
    - Eventos de cambios en la configuración de la EIT Scheduled

## DVB SI Listeners

- Los métodos se encuentran en **SIDatabase** y son bastante Obvios:
  - public void **addNetworkMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int networkId)
    - Events de cambio en la Network
  - public void **addPMTServiceMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int originalNetworkId, int transportStreamId, int serviceId)
    - Events de cambio en la PMTService
  - public void **addServiceMonitoringListener**(org.dvb.si.SIMonitoringListener listener, int originalNetworkId, int transportStreamId)
    - Events de cambio en un Service

## DVB SI Listeners

- Los objetos que se recibirán serán del tipo **SIMonitoringEvent** el cual contiene información variada, de manera que cubre todos los tipos de monitorización.
- Para saber a qué tipo de monitorización corresponde se dispone del método en el SIMonitoringEvent **getSIInformationType()**

```
public class SIMonitoringEvent extends java.util.EventObject {  
    public Object getSource()  
    public byte getSIInformationType() (ver SIMonitoringType en siguiente slide)  
    public int getNetworkID()  
    public int getBouquetID()  
    public int getOriginalNetworkID()  
    public int getTransportStreamID()  
    public int getServiceID()  
    public java.util.Date getStartTime()  
    public java.util.Date getEndTime()  
}
```

## DVB SI Listeners

```
public interface SIMonitoringType{  
    public final static byte NETWORK = 1;  
    public final static byte BOUQUET = 2;  
    public final static byte SERVICE = 3;  
    public final static byte PMT_SERVICE = 4;  
    public final static byte PRESENT_FOLLOWING_EVENT = 5;  
    public final static byte SCHEDULED_EVENT = 6;  
}
```

|                         |   |
|-------------------------|---|
| <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   |