



Curso Multimedia Home Platform 1.1.2

System Information III. Filtrado de Secciones

Accediendo a su contenido con Filtros

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 Section Filtering ? Recordemos conceptos de Protocolos de Bajo nivel

- Principalmente lo vamos a usar para acceder a datos de streams MPEG-2 a los que no tenemos acceso de ninguna otra forma.
- Los Streams MPEG permiten transportar, además de Audio y Video, Sistemas de Ficheros, información de CA, datos para permitir la descryptación, subtítulos, referencias de tiempo...PSI...
- Como ya sabemos, la información de un **TS(tream)** viaja multiplexada en su más bajo nivel en paquetes de **188 bytes**. Los Streams de Video y Audio utilizan por encima un esquema de transmisión denominado **PES: Packetized Elementary Stream**, y de igual manera la información de **PSI** utiliza lo que se llama **Private Sections**.
- Si leemos el apartado 4.2 del documento ETSI TR 101 202 V1.2.1, **Implementation guidelines for Data Broadcasting**, leemos lo siguiente:

¿ Qué es Section Filtering ? Recordemos conceptos de Protocolos de Bajo nivel

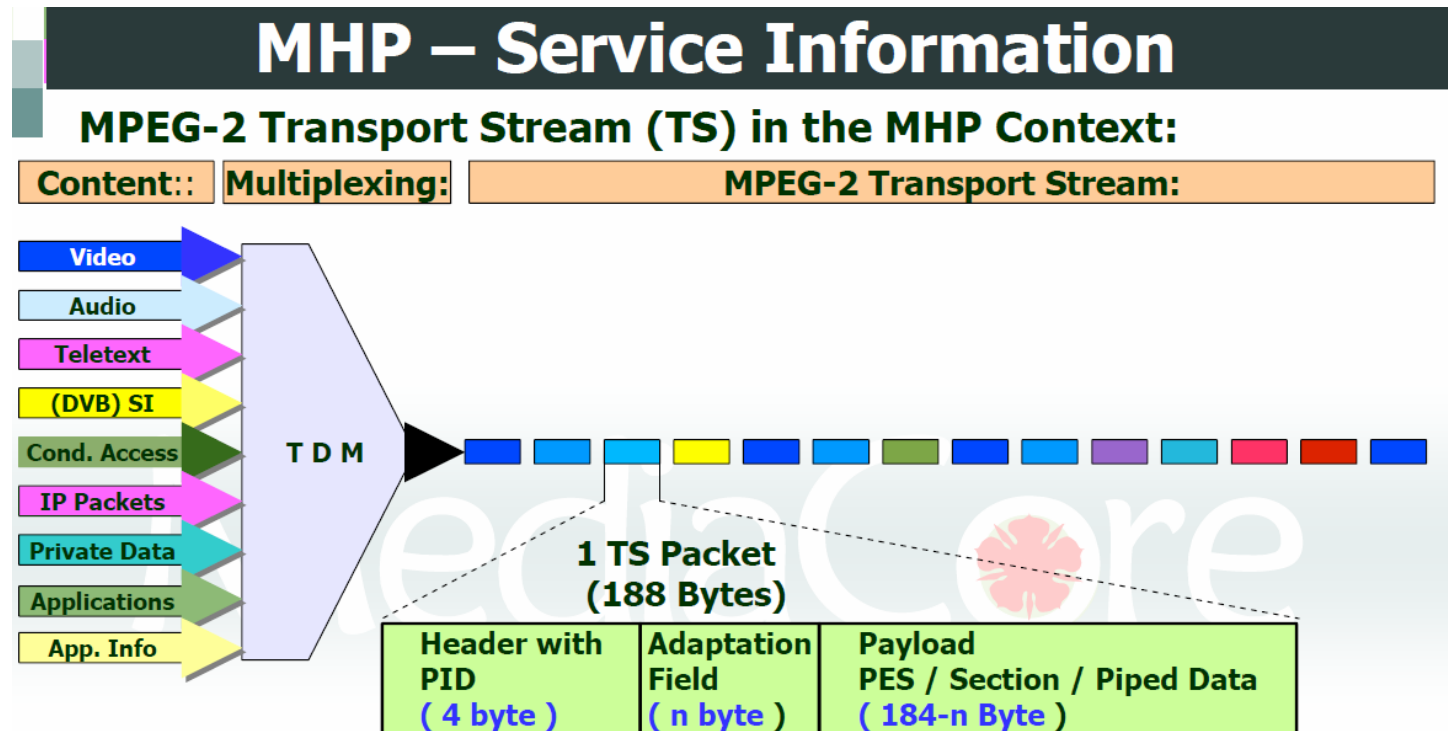
4.2 , ETSI TR 101 202 V1.2.1

–Generally data of any kind of protocols are transmitted in packetized form ("datagrams"). These datagrams may have different length. If the data are not packetized or the packetization method is irrelevant or hidden to the DVB transmission chain the most appropriate way of transmission is the **Data Pipe (see EN 301 192 [1], clause 4)**.

–**On the layer of MPEG-2 Transport Stream** data are transmitted within packets with a fixed length of **188 bytes (184 bytes payload)**, therefore datagrams of higher layers must be fragmented at the transmission side and be re-assembled at the reception. For fragmentation of the datagrams there are three possible ways (see also figure 4.1):

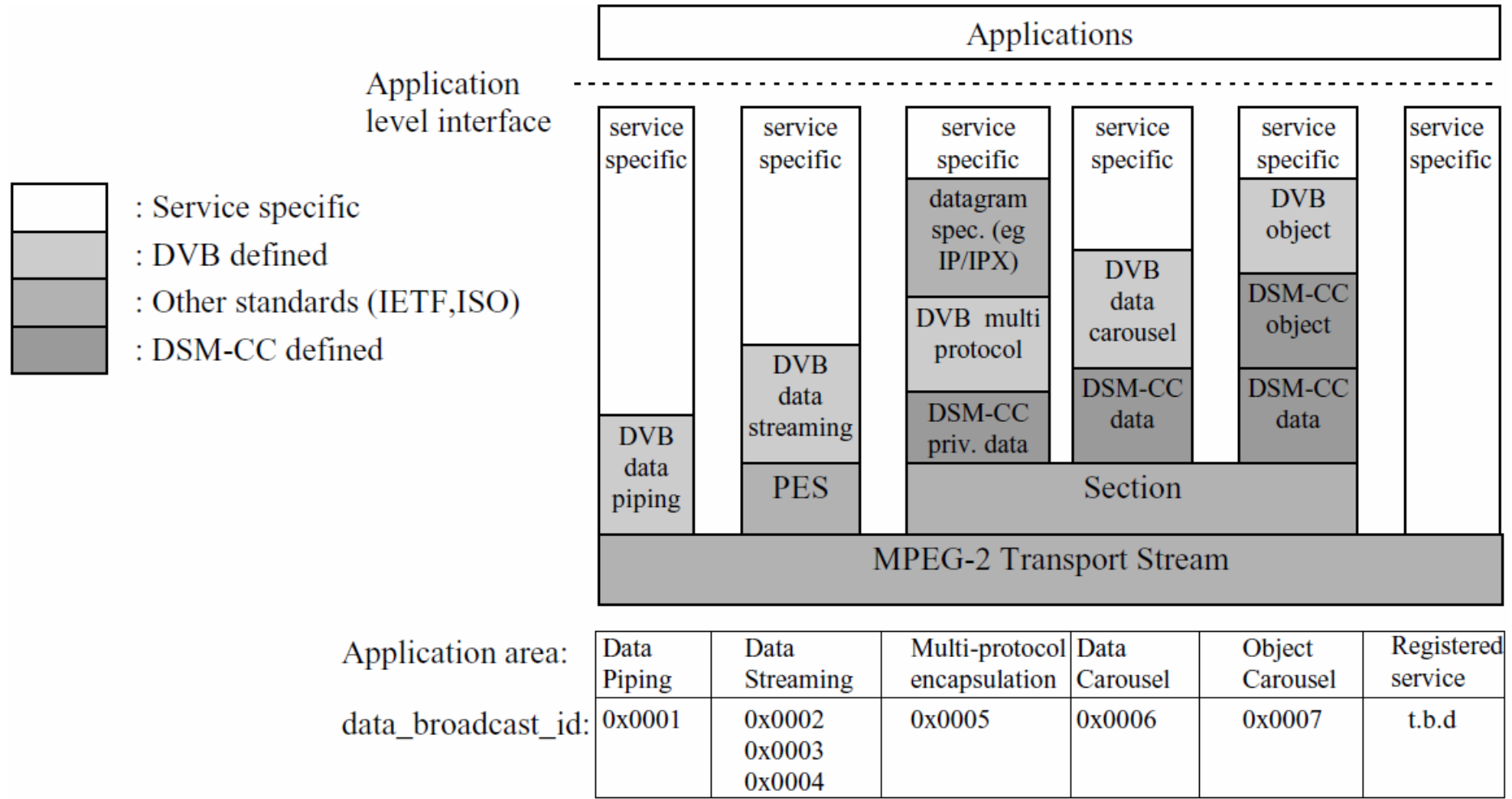
- **Private mechanisms based on the Data Pipe.**
- **MPEG-2 Packetized Elementary Streams (PES).**
- **MPEG-2 Sections.**

MPEG-2 Transport Stream



- **MPEG-2 packets can contain :**
Video, Audio, Teletext, Data streaming (13818-1)
DSM-CC Sections (data carousel, object carousel, SI-tables, etc) (13818-6)
DVB Data Piping

Protocolos y Formatos (TR 101 202)



¿ Qué es Section Filtering ? Recordemos conceptos de Protocolos de Bajo nivel

4.2 , ETSI TR 101 202 V1.2.1

- **MPEG-2 PES** provides a mechanism to transmit datagrams of variable size with a maximum length of **64 kbytes**. Additionally it provides the facility to **synchronize different data streams accurately** (as used in MPEG for synchronization of Video and Audio), therefore it was chosen by DVB for the transmission of **synchronous and synchronized but also asynchronous data streams** (see EN 301 192 [1], clauses 5 and 6).
- **MPEG-2 Sections** can be used to transmit datagrams of variable size with a maximum length of **4 kbytes**. The transmission is **asynchronous**. **MPEG-2 Sections are built in a way that MPEG-2 demultiplexers available on the market** can filter out single sections in hardware which may reduce the required software processing power of the receiver. This is the main reason why the **MPEG-2 Sections have been chosen as the mechanism for the transmission of encapsulated protocols and data carousels**.

Un apunte: data_broadcast_id (recordemos)

- El **data_broadcast_descriptor** tiene el formato apuntado (EN 300 468) siendo su tag = 0x64, y sirve para ofrecer información acerca del tipo de Protocolo empleado que viaja en un ES.
- El **data_broadcast_descriptor** puede aparecer en la SDT y EIT mientras que una versión reducida lo hace en la **PMT: data_broadcast_id_descriptor (tag = 0x66)**. En el primer caso “apunta” a un stream (component_tag) y en el segundo lo califica (loop).
- El campo **data_broadcast_id** nos dice el modelo seguido para transmitir datos en el ES indicado. Por ejemplo, valdrá **0x0005** para indicar que este descriptor de data nos lleva a un Stream de tipo **Multiprotocol Encapsulation**. Ved tipos existentes en la siguiente slide.
- Interesante el tema del data_broadcast_descriptor y el data_broadcast_id. Ved EN 300 192, EN 300468 y TR 101 202.

Syntax	Number of bits	Identifier
data_broadcast_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
data_broadcast_id	16	uimsbf
component_tag	8	uimsbf
selector_length	8	uimsbf
for (i=0; i<selector_length; i++){		
selector_byte	8	uimsbf
}		
ISO_639_language_code	24	bslbf
text_length	8	uimsbf
for (i=0; i<text_length; i++){		
text_char	8	uimsbf
}		
}		

Syntax	Number of bits	Identifier	
data_broadcast_id_descriptor(){			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	TR 101 162
for(i=0; i < N;i++){			
id_selector_byte	8	uimsbf	
}			
}			

Un apunte: valores de **data_broadcast_id** reservados por DVB y propietarios.

Data broadcast specification	Data_broadcast_id
Reserved for future use	0x0000
Data pipe	0x0001
Asynchronous data stream	0x0002
Synchronous data stream	0x0003
Synchronised data stream	0x0004
Multi protocol encapsulation	0x0005
Data Carousel	0x0006
Object Carousel	0x0007
DVB ATM streams	0x0008
Higher Protocols based on asynchronous data streams	0x0009
Reserved for future use by DVB	0x000A-0x00FF
Reserved for registration	0x0100-0xFFFFE
Reserved for future use	0xFFFF

Data_broadcast_id	Data broadcast specification name
0x0100	Eutelsat Data Piping
0x0101	Eutelsat Data Streaming
0x0102	SAGEM IP encapsulation in MPEG-2 PES packets
0x0103	BARCO Data Broadcasting
0x0104	CyberCity Multiprotocol Encapsulation (New Media Communications Ltd.
0x0105	CyberSat Multiprotocol Encapsulation (New Media Communications Ltd.
0x0106	The Digital Network
0x0107	OpenTV Data Carousel
0x0108	Panasonic
0x0109	MSG MediaServices GmbH
0x0110	Televizja Polsat
0x0111	UK DTG
0x0112	SkyMedia
0xB BBB	Bertelsmann Broadband Group
0xB BBB1	BBG Data Caroussel
0xB BBB2	BBG Object Caroussel

Introducción

- **Las Sections** (o también llamadas Private Sections, ved capítulo DVB-SI) son utilizadas tanto para transmitir información de **System Information** como de **DSMCC data**, lo importante es que para extraer la información de las mismas tenemos la ayuda del Hardware!!!
- Recordemos:
 - 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

- Para buscar las sections se podrá filtrar por PID o table_id o bytes...dentro de un Stream
- Es posible que cuando accedemos a DVB SI sean chips los que estén haciendo el trabajo de recuperar las secciones requeridas y no software cuyo trabajo afectaría muchísimo al performance del sistema. Normalmente los STBs modernos disponen de **al menos 8 filtros** hardware, pero pueden tener muchos más.
- La diferencia de tener o no Filtros HW es definitiva con respecto al performance del deco y por tanto de nuestras aplicaciones. Si habéis tenido la oportunidad de usar más de un deco es posible que hayáis notado que en algunos se tarda mucho en cambiar de canal.

Introducción

- MHP especifica que como mínimo deben de haber disponibles **para las aplicaciones** al menos **2 Filtros por Servicio** en decodificación. Si nuestro deco es capaz de sintonizar a la vez dos canales entonces deberían de haber 4 Filtros disponibles, 2 por servicio.
- Si sólo hay una aplicación haciendo uso de Filtros nada le impide usar todos los disponibles. Sin embargo las aplicaciones de un mismo Service habrán de compartir los asignados a su Service. **Ojo: estos filtros no son los que usará el middle para ,por ejemplo, obtener la PSI, son de dedicación exclusiva para las apps.**
- Como ya habréis imaginado/recordado un Section Filter **es un recurso Caro!!!** Y habremos de gestionarlo como tal, recordemos:

`org.davic.mpeg.sections.SectionFilterGroup.attach()`

¿ Qué es una Private Section ?

Table 2-30 – Private section

Syntax	No. of bits	Mnemonic
private_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
private_section_length	12	uimsbf
if (section_syntax_indicator == '0') {		
for (i = 0; i < N; i++) {		
private_data_byte	8	bslbf
}		
}		
else {		
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (i = 0; i < private_section_length-9; i++) {		
private_data_byte	8	bslbf
}		
CRC_32	32	rpchof
}		
}		

iso 13818-1

Private Section. Campos

- **table_id**: indica a qué Table pertenece la información.
 - Existe un mapeo entre el table_id de la descripción de la tabla Lógica de alto nivel y el de la Private Section
 - Puede tener varios valores dependiendo de la información. Hay que conocer estos valores. Por ejemplo:

Table 2-26 iso 13818-1

Table 2 iso 300 468

Table 9.3 DSMCC iso 13818-6

Table 2-26 – table_id assignment values

Value	description
0x00	program_association_section
0x01	conditional_access_section (CA_section)
0x02	TS_program_map_section
0x03	TS_description_section
0x04	ISO_IEC_14496_scene_description_section
0x05	ISO_IEC_14496_object_descriptor_section
0x06-0x37	ITU-T Rec. H.222.0 ISO/IEC 13818-1 reserved
0x38-0x3F	Defined in ISO/IEC 13818-6
0x40-0xFE	User private
0xFF	forbidden

Private Section. Campos

- Table 2 iso 300 468

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

Private Section. Campos

- **Table 9.3 DSMCC iso 13818-6**

Table 9-3 DSM-CC table_id assignments

table_id	DSMCC Section Type
0x00 - 0x37	ITU-T Rec. H.222.0 ISO/IEC 13818-1 defined
0x38 - 0x39	ISO/IEC 13818-6 reserved
0x3A	DSM-CC Sections containing multi-protocol encapsulated data
0x3B	DSM-CC Sections containing U-N Messages, except Download Data Messages
0x3C	DSM-CC Sections containing Download Data Messages
0x3D	DSM-CC Sections containing Stream Descriptors
0x3E	DSM-CC Sections containing private data
0x3F	ISO/IEC 13818-6 reserved
0x40 - 0xFE	User private
0xFF	forbidden

Private Section. Campos

- **section_syntax_indicator:** Si '1' el formato es el de después del “else” , si no, los private_data_bytes siguen al campo private_section_length field.
- **private_section_length:** tamaño del private_data_bytes justo después de este campo. El tamaño no puede superar **4093**
- **table_id_extension:** en ocasiones se usa en conjunción con el table_id. Hay que ver caso por caso lo que puede contener (en caso de section_syntax_indicator='0')
- **current_next_indicator:** cuando vale '1' nos dice que la Private_Section es “Buena” y el campo Version_Number es por tanto la versión correcta. Cuando vale '0' nos indica que esta Private_Section NO es aplicable y que lo será la próxima que llegue con el mismo **section_number** y **table_id**
- **version_number:** número de versión de la private_section. Se incrementa en 1 mod 32 cuando hay cambios en la información que lleva esta Private_Section.
- **section_number:** Número de Section correspondiente a la Información que se está transmitiendo. Empieza en 0.
- **last_section_number:** Número de la última Section de la Private Table que se está enviando usando estas private_sections.

Private Section. Filtrado

- Nos apoyaremos en los valores de **table_id** y **table_id_extension** para filtrar, y también **en los primeros bytes de las private_data_bytes** cuando sea en esos lugares donde resida la información que identifica lo que necesitamos.
 - Recordemos que dentro de estos Private_Sections se encuentran nuevos esquemas de información!!!
- Veamos un ejemplo: **ETSI EN 300 468 V1.8.1, p 20**
 - “...Any sections of the NIT which describe the actual network (that is, the network of which the TS containing the NIT is a part) shall have the table_id 0x40 with the same table_id_extension (network_id)....”
 - Se indica que la **table_extension_id** cuando se describe la Table NIT contendrá la network_id

El API

- Basado en el modelo de reserva de recursos.
- El modelo de petición de información es asíncrono.
- ¿ donde se encuentra ?
org.davic.mpeg.sections

El API. org.davic.mpeg.sections.Section

- Representa el elemento fundamental del API. El dato que deseamos obtener.
- **API:** Los métodos ofrecen lo visto en su formato.
 - public byte[] **getData()**
 - Toda la información de la Section
 - public byte[] **getData(int index, int length)**
 - Parte de los datos de la Section. desde = index (**el primer byte empieza en 1**), número de bytes = length
 - public byte **getByteAt(int index)**
 - **Un byte concreto. El primero empieza en 1**
 - public int **table_id()**
 - public boolean **section_syntax_indicator()**

El API. org.davic.mpeg.sections.Section

- **API:** Los métodos ofrecen lo visto en su formato
 - public boolean **private_indicator()**
 - public int **section_length()**
 - public int **table_id_extension()**
 - public short **version_number()**
 - public boolean **current_next_indicator()**
 - public int **section_number()**
 - public int **last_section_number()**
 - public boolean **getFullStatus()**
 - Indica si el objeto contiene datos válidos
 - public void **setEmpty()**
 - Sirve para indicar que los datos que contiene no son válidos. Se usa con RingSectionFilters para indicar que se puede reutilizar

El API. org.davic.mpeg.sections.SectionFilter

- Nos permitirá definir los criterios de búsqueda. Ver **Davic 1.4.1 part 9** (Annex E) y en menor medida **Davic 1.4.1 part 10** (refs/davic)
- Esta clase **no** se puede instanciar, es la clase Base de 3 tipos distintos de Filtros, que son los que instanciamos, y que sirven para distinguir la cantidad de información a recuperar:
 - **SimpleSectionFilter**: Obtiene una sola Section
 - **RingSectionFilter**: Permite obtener varias Sections, tantas como le indiquemos.
 - **TableSectionFilter**: Define un buffer lo suficientemente grande como para almacenar toda la información relativa a la SI Table que nos interese.
- Estos tres filtros sólo tienen un método propio al margen de un constructor vacío:
 - Section[] **getSections()**

Paso 1

- Creamos el Filtro que nos interesa, SimpleSectionFilter, RingSectionFilter o TableSectionFilter, usando la Factory **SectionFilterGroup**. Más adelante veremos esta Clase en detalle.

Paso 2

- En el Filtro me suscribo mediante el registro de un Listener, para ser notificado de los sucesos en el proceso de filtrado.

- public void **addSectionFilterListener**(SectionFilterListener listener)
 - public void **removeSectionFilterListener**(SectionFilterListener listener)
- ```
public interface SectionFilterListener extends java.util.EventListener{
 public void sectionFilterUpdate (SectionFilterEvent event)
}
```

Los eventos que se pueden recibir son del tipo:

- **SectionFilterEvent**: Tipo Base de todos en el que encontramos **getAppData**
- **TimeOutEvent**
- **IncompleteFilteringEvent**: Ver después TableSectionFilter
- **VersionChangeDetectedEvent**: Ver después TableSectionFilter
- **SectionAvailableEvent**: Nueva Section!!!
- **EndOfFilteringEvent**: Ver después TableSectionFilter

Veamos a continuación los tipos de Filtros....

## Paso 3. Establezco el Filtro

- SectionFilter ofrece una serie de métodos para filtrar, que se pueden clasificar en 2 grupos: filtros sin máscaras y con máscaras.
- **Sin máscaras:**
  - (1) public void **startFiltering**(Object appData, int pid)
    - Queremos obtener las Section de un determinado Stream(pid).
    - El appData sirve para lo mismo que en DVB SI: identificar la petición realizada.
  - (2) public void **startFiltering**(Object appData, int pid, int table\_id)
    - Ídem al anterior + de un **table\_id** determinado.
- **Con Máscaras:** antes de ver los métodos estudiemos en detalle el mecanismo de filtros en la zona de datos usando máscaras.

## ¿ Qué es una Private Section ? (iso 13818-1)

Table 2-30 – Private section

| Syntax                                           | No. of bits | Mnemonic      |
|--------------------------------------------------|-------------|---------------|
| private_section() {                              |             |               |
| <b>table_id</b>                                  | <b>8</b>    | <b>uimsbf</b> |
| <b>section_syntax_indicator</b>                  | <b>1</b>    | <b>bslbf</b>  |
| <b>private_indicator</b>                         | <b>1</b>    | <b>bslbf</b>  |
| <b>reserved</b>                                  | <b>2</b>    | <b>bslbf</b>  |
| <b>private_section_length</b>                    | <b>12</b>   | <b>uimsbf</b> |
| if (section_syntax_indicator == '0') {           |             |               |
| for (i = 0; i < N; i++) {                        |             |               |
| <b>private_data_byte</b>                         | <b>8</b>    | <b>bslbf</b>  |
| }                                                |             |               |
| }                                                |             |               |
| else {                                           |             |               |
| <b>table_id_extension</b>                        | <b>16</b>   | <b>uimsbf</b> |
| <b>reserved</b>                                  | <b>2</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 < private_section_length-9; i++) { |             |               |
| <b>private_data_byte</b>                         | <b>8</b>    | <b>bslbf</b>  |
| }                                                |             |               |
| <b>CRC_32</b>                                    | <b>32</b>   | <b>rpchof</b> |
| }                                                |             |               |
| }                                                |             |               |

## Máscaras

- En la figura de abajo (davic 1.4.1 p9) observamos el formato de las secciones MPEG2. En total hay 8 bytes (0-7) con la información de Cabecera/Header.
- Asumiendo la primera fila como el “número de byte” (0,1,2,3..), vemos que a partir del byte número 3 inclusive (el cual junto con el 4 nos da el valor de la table\_id\_extension) es cuando comienza la zona en la que tiene sentido poder filtrar, pues la anterior, table\_id, ya está contemplada en el Filtro.

### E.8.2 MPEG2 section format (informative)

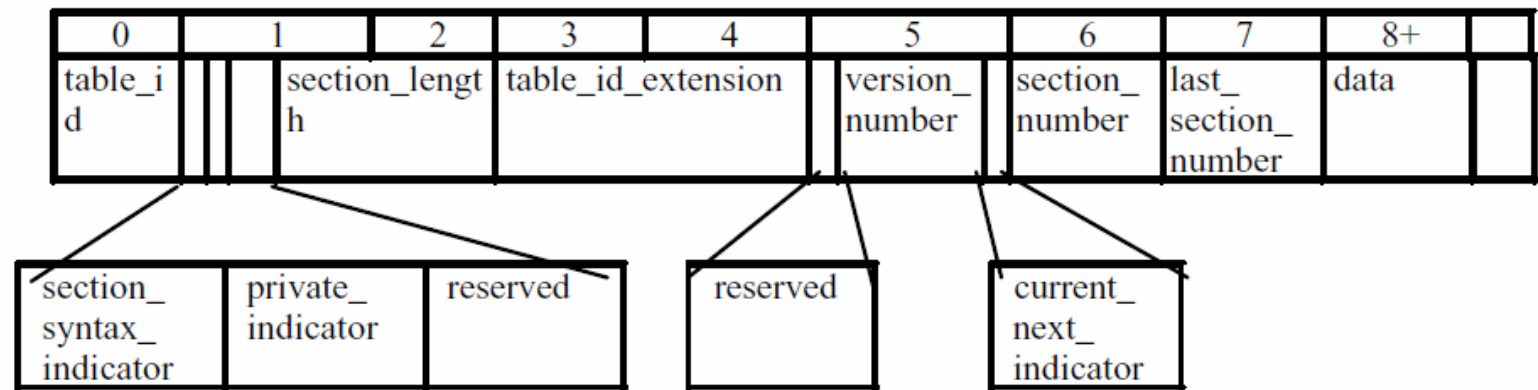


Figure E-4 Format of long header of MPEG-2 section

davic 1.4.1 p9

## Máscaras

- Así pues cuando nos refiramos al filtrado por máscaras, **este comienza en el byte número 3** (table\_id\_extension cuando el formato es largo, y private\_data\_bytes cuando es corto).
- De esta forma si tenemos la posibilidad de definir un filtro de **8 bytes**, y este comienza a aplicarse en el **byte número 3**, entonces seremos capaces de llegar **hasta el byte número 10**, es decir, entramos **3 bytes en zona de datos en el caso de formato largo! (8,9,10) y 8 bytes en el corto.**

### E.8.2 MPEG2 section format (informative)

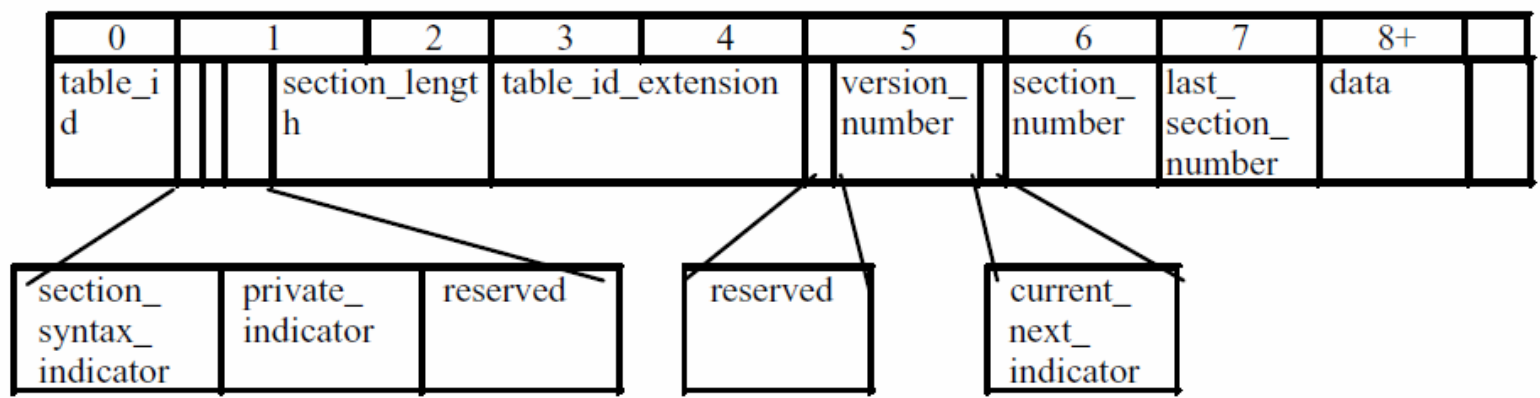


Figure E-4 Format of long header of MPEG-2 section

## Paso 3. Establezco el Filtro

- Con Máscaras:
  - (3) public void **startFiltering**(Object appData, int pid, int table\_id, byte **posFilterDef**[], byte **posFilterMask**[] )
    - Ídem al anterior + filtrando también en la zona de datos (davic 1.4.1 part 9)
    - **posFilterDef**: consiste en un array de 8 bytes de longitud en el que vamos a describir los bits que han de coincidir.
    - **posFilterMask**: consiste en un array de 8 bytes de longitud en el que allí donde hay un bit a **1** se indica que el bit homólogo en **posFilterDef** ha de tenerse en cuenta para el filtro; y cuando el valor es **0** se deberá ignorar.
    - Para todos los de máscaras: siempre se empieza en el byte número 3.
  - (4) public void **startFiltering**( Object appData, int pid, int table\_id, **int offset**, byte posFilterDef[], byte posFilterMask[] )
    - Ídem al anterior + podemos decidir desde qué byte comenzar a filtrar = **offset**
    - **El valor del offset debe ser :  $3 \leq \text{offset} < 31$ . Siendo 3 el valor por defecto que corresponde al número de byte en el que se comienza siempre (el cuarto byte)**

## Paso 3. Establezco el Filtro

- Con Máscaras:
  - (5) public void **startFiltering**( Object appData, int pid, int table\_id, byte posFilterDef[], byte posFilterMask[], **byte negFilterDef[], byte negFilterMask[]**)
    - Ídem al 3 + filtrando también en negativo (davic 1.4.1 part 9). El filtro final es = Pos and (not Neg)
    - **negFilterDef**: consiste en un array de 8 bytes de longitud en el que vamos a describir los bits que **NO han de coincidir**.
    - **negFilterMask**: consiste en un array de 8 bytes de longitud en el que allí donde hay un bit a **1** se indica que el bit homólogo en **negFilterDef** ha de tenerse en cuenta para el filtro; y cuando el valor es **0** se deberá ignorar.
  - (6) public void **startFiltering**( Object appData, int pid, int table\_id, int **offset**, byte posFilterDef[], byte posFilterMask[], byte **negFilterDef[], byte negFilterMask[]**)
    - Ídem al 4 + filtrando también en negativo (davic 1.4.1 part 9)

## Paso 4. Attach de TS. Arranca el Filtrado!!!

- ¿ Cuando comienza el Filtrado ? A pesar de que los métodos anteriores empiezan con startX este no se produce mientras que el filtro no esté “**Attached**” a un TS en el que poder filtrar. Más adelante vemos el detalle.
- Si cuando se llama a startX se encuentra “Attached” entonces comienza de forma inmediata, si no, simplemente almacena los parámetros hasta que esa asociación se produce.
- ¿ Hasta cuando dura el filtrado ? Para establecer este dato disponemos de dos opciones complementarias
  - public void **setTimeout**(long milliseconds)
    - Cuando se produce el timeout se deja de filtrar y se le manda un evento del tipo **TimeoutEvent** al listener.
    - Si se pasa como parámetro 0 indica que no hay timeout establecido.
  - public void **stopFiltering**()
    - Detiene el proceso de filtrado en curso

## El API. org.davic.mpeg.sections.SimpleSectionFilter

- Recibe una sola Section y para de filtrar.
- Cuando se recibe el event **SectionAvailableEvent** se deberá llamar al método siguiente para recuperar la información.

```
public Section getSection()
```

No tiene más métodos.

## El API. `org.davic.mpeg.sections.RingSectionFilter`

- Este Filtro dispone internamente de un buffer cíclico de Sections que se van rellenando con la información de las Section encontradas en el Stream, de tal manera que el filtro se detiene cuando se encuentra una Section **no Empty**. Lanzará un evento **EndOfFilteringEvent**
- No existe una manera de decirle cuantas Sections queremos en el Buffer.
- El método **Section[] getSections()** nos devuelve siempre el buffer entero, de manera que somos nosotros los responsables de recorrer el array, tomar la información que nos interesa (una buena opción es `Section.clone()`) y establecer a Empty las Sections del buffer si queremos que siga filtrando.
- Cada vez que el Filtro encuentra una Section lanzará un evento del tipo: **SectionAvailableEvent**

## El API. `org.davic.mpeg.sections.TableSectionFilter`

- Permite filtrar todas las Sections de una Table con mínima intervención.
- Cuando comienza a Filtrar y recibe la primera Section determina el número de Sections que necesita, reserva todo el espacio necesario y re-setea el Filtro para que comience de nuevo.
- Cada vez que llega una Section se recibe el evento **SectionAvailableEvent**.
- El método **Section[] getSections()** ofrece el buffer con todas las Sections disponibles con **null** allí donde aún no se ha recibido. Se encuentran en orden de `section_number`.
- Todas las Section tendrán la misma versión. Si se encuentra una Section cuya versión es distinta de aquella que estamos filtrando se lanzará un evento (sólo esta vez aunque lleguen más, por acción de filtrado) **VersionChangeDetectedEvent**, y se seguirá filtrando con la primera (vamos, que si queréis re-filtrar para conseguir la otra versión habréis de parar el filtro...)
- Cuando ha conseguido filtrar todas se manda un evento del tipo: **EndOfFilteringEvent**.
- Si los parámetros del filtro no han sido correctamente establecidos de manera que por ejemplo se bloquee, se lanzará un **IncompleteFilteringEvent**.

## El API. Procesado de Sections

- Cada vez que se recibe una notificación en el Listener de que se ha recibido una Section, conviene procesarla en un Thread separado para no bloquear la notificación de eventos.

## Proceso de Attachment

- Mediante: **SectionFilterGroup**
  - **SectionFilterGroup** implements `org.davic.resources.ResourceProxy`, y `org.davic.resources.ResourceServer`
- Básicamente lo que permite **SectionFilterGroup** es realizar operaciones atómicas para un grupo de Filtros físicos, de manera que se minimiza el riesgo de “deadlocks”
- También funciona como Factory de los Filtros, de hecho, no se pueden crear sin usar esta Factory! (constructores privados)
- Por supuesto ofrece el API necesario para la gestión de recursos caros!

Veamos su API en detalle

## El API. org.davic.mpeg.sections.SectionFilterGroup

- El API. Constructores
  - public **SectionFilterGroup**(int numberOfFilters)
    - Creamos un SectionFilterGroup indicando el número de Filtros físicos que necesitamos reservar.
    - OJO: son recursos caros, de manera que cuantos más solicitemos más probable será que se nos **DENIEGUEN. Hemos de usar en cada momento exclusivamente los que necesitamos**
    - La reserva de varios recursos a la vez no es muy “generosa” entre las aplicaciones, por eso es importante no solicitar más de lo necesario.
    - Este enfoque permite por otro lado que una aplicación reciba el total de lo que necesita para operar **o nada**.
  - public **SectionFilterGroup**(int numberOfFilters, boolean highPriority)
    - Ídem al anterior pero con la posibilidad de indicar una prioridad de manera que en caso de conflictos en la asignación de los recursos “caros” facilitemos este trabajo al middle.
    - En el caso de que dos aplicaciones tengan la misma prioridad, aquella que haya establecido el parámetro a false=baja será la que será “despojada” de su recurso caro.
    - Por defecto la prioridad siempre es alta.

## El API. org.davic.mpeg.sections.SectionFilterGroup

- El API. Factorías
  - public **SimpleSectionFilter** newSimpleSectionFilter()  
Creamos un SimpleSectionFilter
  - public **SimpleSectionFilter** newSimpleSectionFilter(int size)
    - Creamos un SimpleSectionFilter que incluirá en la Section como mucho el número de bytes indicado. (no recomendado)
  - public **RingSectionFilter** newRingSectionFilter(int ringSize)  
Creamos un RingSectionFilter indicando el número de slots del buffer.
  - public **RingSectionFilter** newRingSectionFilter(int ringSize, int size)  
Ídem al anterior indicando el tamaño máximo de los bytes / section.
  - public **TableSectionFilter** newTableSectionFilter(), newTableSectionFilter(int size)  
Creamos un TableSectionFilter, y con versión size

## El API. org.davic.mpeg.sections.SectionFilterGroup

- El API. Recursos Caros

- public void **attach**(org.davic.mpeg.TransportStream stream, org.davic.resources.ResourceClient client, Object requestData)

- Asociamos un TS al SectionFilterGroup indicando quien es el usuario del **Recurso Caro**. Así mismo pasamos la información requestData que se puede utilizar para “Dialogar” entre aplicaciones contendientes.

- public void **detach**()

- Desasociamos el TS y liberamos todos los recursos caros. Todos los filtros en ejecución se paran.

- El API. Informativos

- public org.davic.mpeg.TransportStream getSource()

- public org.davic.resources.ResourceClient getClient()

## El API. org.davic.mpeg.sections.SectionFilterGroup

- El API. Notificación de Estado de Reserva de Recursos Caros:
  - Estos métodos **NO** son para nosotros, son de gestión del middle en cuanto al ResourceManagement: es el API de ResourceServer
    - public void **addResourceStatusEventListener** (org.davic.resources.ResourceStatusListener listener)
    - public void **removeResourceStatusEventListener** (org.davic.resources.ResourceStatusListener listener)

## Ejercicios Bloque SFSI-1

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