



Curso Multimedia Home Platform 1.1.2

Return Channel Connection API

Usando ISPs para acceder a la red

Curso Multimedia Home Platform 1.1.2

Copyright 2008 © Enrique Pérez Gil

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

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

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

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

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

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

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

Introducción

- Resumidamente lo que vamos a ver en este capítulo es **como gestionar una conexión a la red usando un MODEM.**
- EL uso de modems para acceder a Internet en MHP STBs ha sido la primera opción al alcance de estos dispositivos, y aunque ya muchos incorporan Ethernet para su conexión directa a routers disponiendo por tanto de una conexión permanente, la opción del modem está relativamente extendida (según qué países) y es un legado a tener en cuenta (p.e. en Italia).
- Lo que vamos a ver es la **gestión de establecimiento de la conexión y su liberación, NO** las características del protocolo que usemos después para acceder a la información: HTTP, HTTPS...

Introducción

- Para situarnos, lo que obtendremos una vez hemos establecido la conexión es una IP en una red, y dependiendo del profile de MHP que estemos usando tendremos algo más o no:
 - En MHP 1.0.x sólo se soporta HTTP 1.0 y DNS
 - En MHP 1.1.x se soporta HTTPS. (En detalle en el capítulo Protocolos)
- Como decíamos al principio, si tienes una conexión permanente olvídate de este capítulo, sin embargo si necesitas gestionar una conexión a través de un modem o cualquier otro dispositivo similar entonces si debes conocer como se hace.
- Por cierto el paquete donde residen las clases de este API es: **org.dvb.net.rc**
- A pesar de que este API se usa principalmente para conectarnos a una red a través de un dispositivo como un modem, el API también abarca la gestión de conexiones en entornos de conexión permanente.

Empecemos.

Conexión

- Para solicitar una conexión usamos la clase: **org.dvb.net.rc.RCInterfaceManager**
- El objeto es un singleton que obtenemos así: `RCInterfaceManager.getInstance()`; una vez que lo tenemos disponemos de **4 métodos** para solicitar conexión:

```
public RCInterface getInterface(InetAddress addr)
```

- Usado para conexiones tipo Modem pasando la `InetAddress` que vamos a usar.

```
public RCInterface getInterface(Socket s)
```

```
public RCInterface getInterface(URLConnection u)
```

- Usados para “always-connected”

- El primero es el que necesita de gestión. El tipo de objeto que devuelve este método es uno que hereda de **RCInterface**: **org.dvb.net.rc.ConnectionRCInterface**
- Los dos últimos asumen que la conexión ya está establecida, de forma que lo que devuelven es el Channel Interface que se está usando.

Conexión

- Para saber si un RCInterface es de tipo conexión u “always-connected” basta con ver si es una instancia de **ConnectionRCInterface** o no.
- Habíamos dicho 4 métodos. Existe un cuarto que me ofrece **todos los Channel Interfaces existentes**, de manera que se pueda seleccionar el que interesa:

```
public RCInterface[] getInterfaces()
```

- El API de RCInterface, clase base de ConnectionRCInterface, es muy escueto:

- public int **getType():**

```
public final static int org.dvb.net.rc.RCInterface.TYPE_PSTN = 1;  
public final static int org.dvb.net.rc.RCInterface.TYPE_ISDN = 2;  
public final static int org.dvb.net.rc.RCInterface.TYPE_DECT = 3;  
public final static int org.dvb.net.rc.RCInterface.TYPE_CATV = 4;  
public final static int org.dvb.net.rc.RCInterface.TYPE_LMDS = 5;  
public final static int org.dvb.net.rc.RCInterface.TYPE_MATV = 6;  
public final static int org.dvb.net.rc.RCInterface.TYPE_RCS = 7;  
public final static int org.dvb.net.rc.RCInterface.TYPE_UNKNOWN = 8;  
public final static int org.dvb.net.rc.RCInterface.TYPE_OTHER = 9;
```

- public int **getDataRate():**

- data rate in KBaud o -1, si no se conoce. (ved el Java API)

Conexión. org.dvb.net.rc.ConnectionRCInterface

- RCInterfaceManager implementa **org.davic.resources.ResourcesServer**, es decir, gestiona **recursos caros**, que son los que nos proporciona. Habremos de tratar los mensajes de *Scarce Resources* en nuestros Channel Interfaces.
- ¿ qué ofrece **ConnectionRCInterface** ? ¿ cómo defino mis parámetros de conexión ? ¿ cómo solicito la conexión ? ¿ cómo sé que estoy conectado ?
¿ cómo me desconecto ?

Vamos por pasos

Conexión. `org.dvb.net.rc.ConnectionRCInterface`

- **Paso 1**: Voy a gestionar un recurso caro. Necesito implementar el interface **`org.davic.resources.ResourceClient`**.
- **Paso 2**: Voy a preparar mis parámetros de conexión: para ello uso la clase **`org.dvb.net.rc.ConnectionParameters`**, la cual me ofrece 2 constructores para configurarla que se definen por sí mismos:
 - `public ConnectionParameters(String tfnumber, String username, String password)`
 - `public ConnectionParameters(String tfnumber, String username, String password, InetAddress[] dns)`

En el primer caso se asume que los DNS los proporciona el proveedor

Conexión. org.dvb.net.rc.ConnectionRCInterface

- **Paso 3**: Voy a solicitar el ConnectionRCInterface. Una opción puede ser:

```
RCInterface[] ifaces = org.dvb.net.rc.RCInterfaceManager.getInstance().getInterfaces();
if (ifaces!=null){
    for (int i=0;i<ifaces.length;i++)
        if (ifaces[i] instanceof ConnectionRCInterface) return (ConnectionRCInterface)ifaces[i];
}
return null;
```

- **Paso 4**: Voy a reservar el recurso (es caro). Suponiendo que crci es mi ConnectionRCInterface:

```
crci.reserve (mi_instancia_de_ResourceClient, requestData (generalmente null) );
```

mi_instancia_de_ResourceClient recibirá las notificaciones para “compartir” el recurso.

Conexión. org.dvb.net.rc.ConnectionRCInterface

- **Paso 5**: Ahora que lo tengo reservado establezco mis parámetros de conexión (si es que no quiero usar los que disponga por defecto) usando el método:

```
public void setTarget(ConnectionParameters target)
```

```
crci.setTarget(myconnectionParams);
```

OJO: si cambio mis parámetros en el objeto myconnectionParams se tendrán en cuenta la próxima vez que se use el objeto.

- public ConnectionParameters **getCurrentTarget()**
 - Me devuelve los parámetros actuales, bien sean los que haya establecido o bien los que hubiera por defecto.
- public void **setDefaultTarget()**
 - Establece como parámetros de conexión los de por defecto.

Conexión. org.dvb.net.rc.ConnectionRCInterface

- **Paso 6**: antes de conectar, puesto que **necesitamos** saber en qué momento se realiza la conexión, o se cae (en el siguiente método vemos porqué)...nos vamos a suscribir al siguiente Listener:

```
public void addConnectionListener( ConnectionListener l )  
public interface ConnectionListener extends java.util.EventListener{  
    public void connectionChanged(ConnectionRCEvent e);  
}
```

Los eventos que podemos recibir son:

- ConnectionDroppedEvent
- ConnectionEstablishedEvent
- ConnectionFailedEvent
- ConnectionTerminatedEvent
- NoDialToneEvent
- TargetBusyEvent

Conexión. org.dvb.net.rc.ConnectionRCInterface

- **Paso 7**: Conectemos: `public void connect();` Si la conexión ha ido bien podremos usar java.net normalmente. Si se hubiera necesitado marcar un TF se le habría preguntado al usuario.

```
crci.connect();
```

- **OJO: este método es asíncrono!!** El éxito o no de la conexión lo sabremos por el Listener establecido antes, de ahí la necesidad de suscribirse. Sin embargo si en la llamada al método hay algún problema conocido (no reservaste, no hay RCInterfaces disponibles)...lanzará una excepción.

- **Paso 8**: Hemos terminado nuestras comunicaciones:
 - Primero desconectamos: `disconnect();` Método **asíncrono!**
 - Nos desuscribimos del listener. `removeConnectionListener(this);`
 - Liberamos el recurso (es caro): `release()`

Conexión. org.dvb.net.rc.ConnectionRCInterface

- Otros métodos de ConnectionRCInterface
 - public ResourceClient **getClient()**.
Nos da el ResourceClient proporcionado cuando se reservó.
 - public int **getConnectedTime()**
Segundos que lleva el interface conectado.
 - public float **getSetupTimeEstimate()**
Segundos estimados que puede tardar en conectar.
 - public boolean **isConnected()**
Si está conectado (envía y recibe paquetes)

org.dvb.net.rc.RCInterfaceManager

- RCInterfaceManager ofrece un Listener para ser notificados de la reserva/liberación de RCInterfaces en el sistema.

```
public void addResourceStatusEventListener (org.davic.resources.ResourceStatusListener listener)
public void removeResourceStatusEventListener(org.davic.resources.ResourceStatusListener listener)
public interface ResourceStatusListener extends java.util.EventListener{
    public void statusChanged(ResourceStatusEvent event);
}
```

- **Los dos eventos** que se reciben son los siguientes, en cuyo método **public Object getSource()** se ofrece el **RCInterface que ha sido reservado o liberado**

```
org.dvb.net.rc.RCInterfaceReleasedEvent
org.dvb.net.rc.RCInterfaceReservedEvent
```

Ejercicios Bloque RC-1

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