MHP/OCAP iTV Applications in a Nutshell

Cédric Monnier

Senior Product Manager
NDS
www.nds.com/middleware.html

TS-4255

Understanding Java™ Technology in Set Top Box

Experimenting With Java Technology on TV

Learn how to develop an interactive TV application for MHP/OCAP Digital Set Top Boxes, with best practises and some good advices

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices

Once Upon a Time

- Spring 1997: creation of the DVB-TAM group
  - Objectives: define an open API allowing for application interoperability
  - Main actors/contributors: Microsoft, Sun, C+T/Canal+, Philips, Panasonic, Sony, Nokia, BBC, HP, Intel, OpenTV, and later, Liberate
- 1998: DVB decides to go for a Java-centric solution

A Brief Analysis of What Was Required

1. An application needs to be downloaded into the STB
   → Need to define an application download protocol

3. A download protocol is not enough: we need to define where to locate the application in the modules that carry it, and access to its attributes (e.g., boot class, application name, possibly icon…)
   → Need to define an application signaling protocol

4. Once it is downloaded, we need to get it started
   - How to start it? Upon service selection? Auto start? Can it survive zapping? How to control its execution?

   → Need to define an application lifecycle model

A Brief Analysis of What Was Required

1. While it is running, and even before it started its execution, has the application the right to access the resources that are made available to a downloaded application? Is it a trusted application?
   → Need to define a security model

3. What are the resources a downloaded application need to access in order to provide its service?
   → Need to identify these resources and define APIs to access it

4. For scarce resources, how to manage them when several apps are executing at the same time?
   → Need to provide minimum mechanisms in order to be able to manage scarce resources

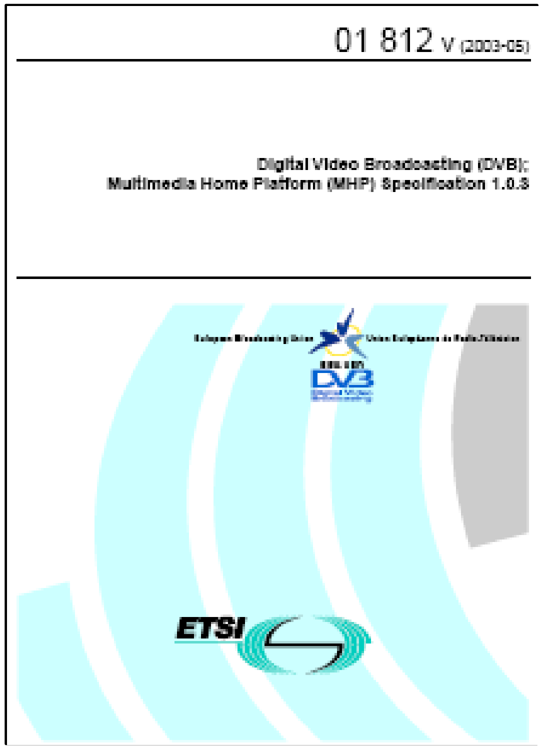
A Brief Analysis of What Was Required

1. For images, fonts, audio clips, what decoder can a downloaded application expect to be present in the STB ?
   → Need to define a minimum set of content formats that need to be supported by the STB

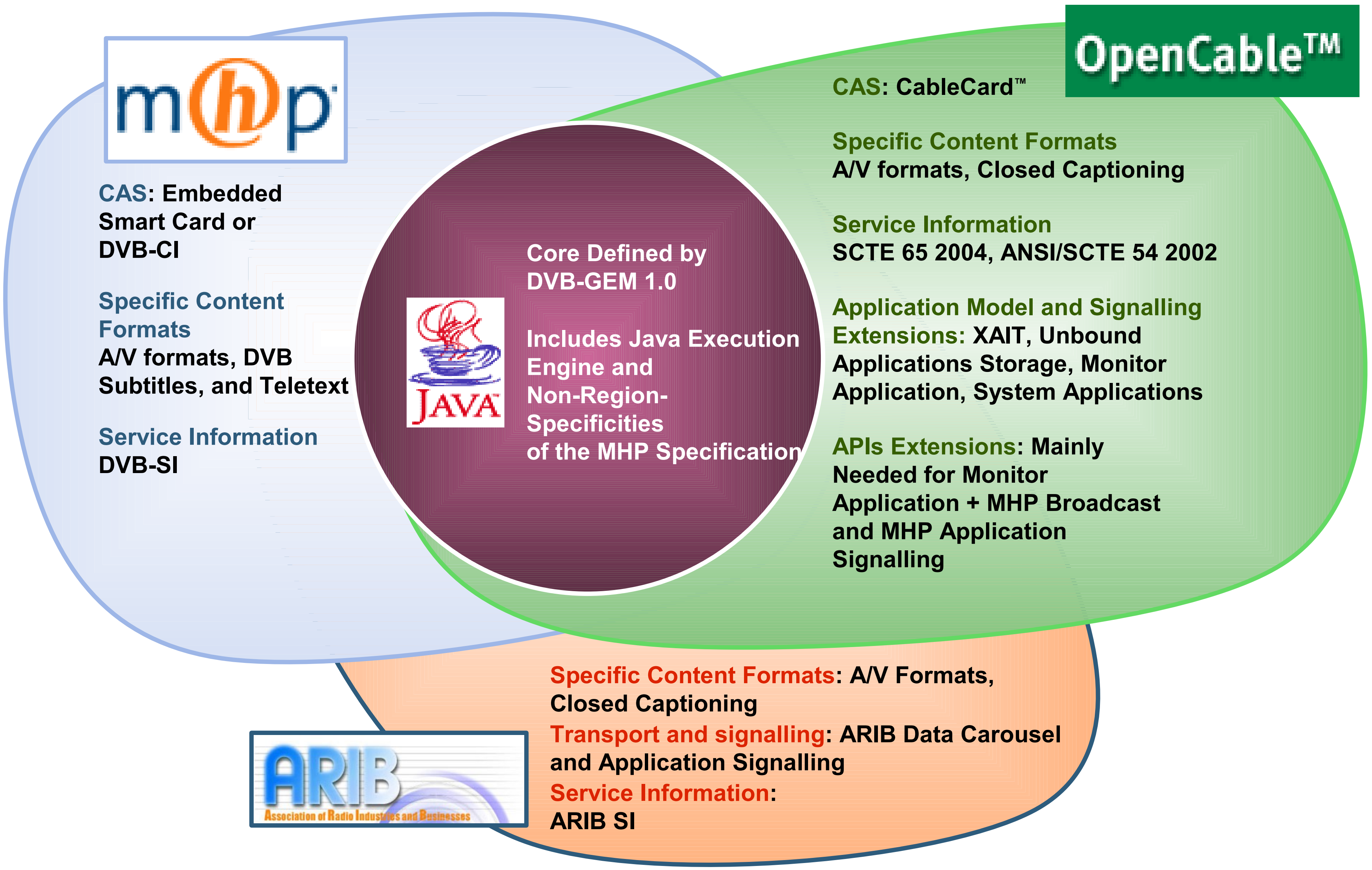
3. For use of the return channel, what does an application require?
   → Need to define the protocol suite that needs to be supported on the return channel, and how to secure it

What We Got

- January 2002: Release of MHP 1.0.2 specification
  - A subset of PersonalJava™ technology with Java TV™ API
  - HAVi (widget set)
  - Low-level MPEG section access
  - DSMCC access
  - SI low-level access API (DVB dependant)
  - Conditional access API (‘CI-ready’)
  - Application discovery and launching API
  - Tuner control API
- January 2003: Release of GEM 1.0.0 specification
  - Objective: Build on momentum of MHP spec by enabling it on other markets


01 812 V (2003-05)

Digital Video Broadcasting (DVB);
Multimedia Home Platform (MHP) Specification 1.0.3



CAS: Embedded Smart Card or DVB-CI

Specific Content Formats
A/V formats, DVB Subtitles, and Teletext

Service Information
DVB-SI

Core Defined by DVB-GEM 1.0

Includes Java Execution Engine and Non-Region-Specificities of the MHP Specification

CAS: CableCard™

Specific Content Formats
A/V formats, Closed Captioning

Service Information
SCTE 65 2004, ANSI/SCTE 54 2002

Application Model and Signalling Extensions: XAIT, Unbound Applications Storage, Monitor Application, System Applications

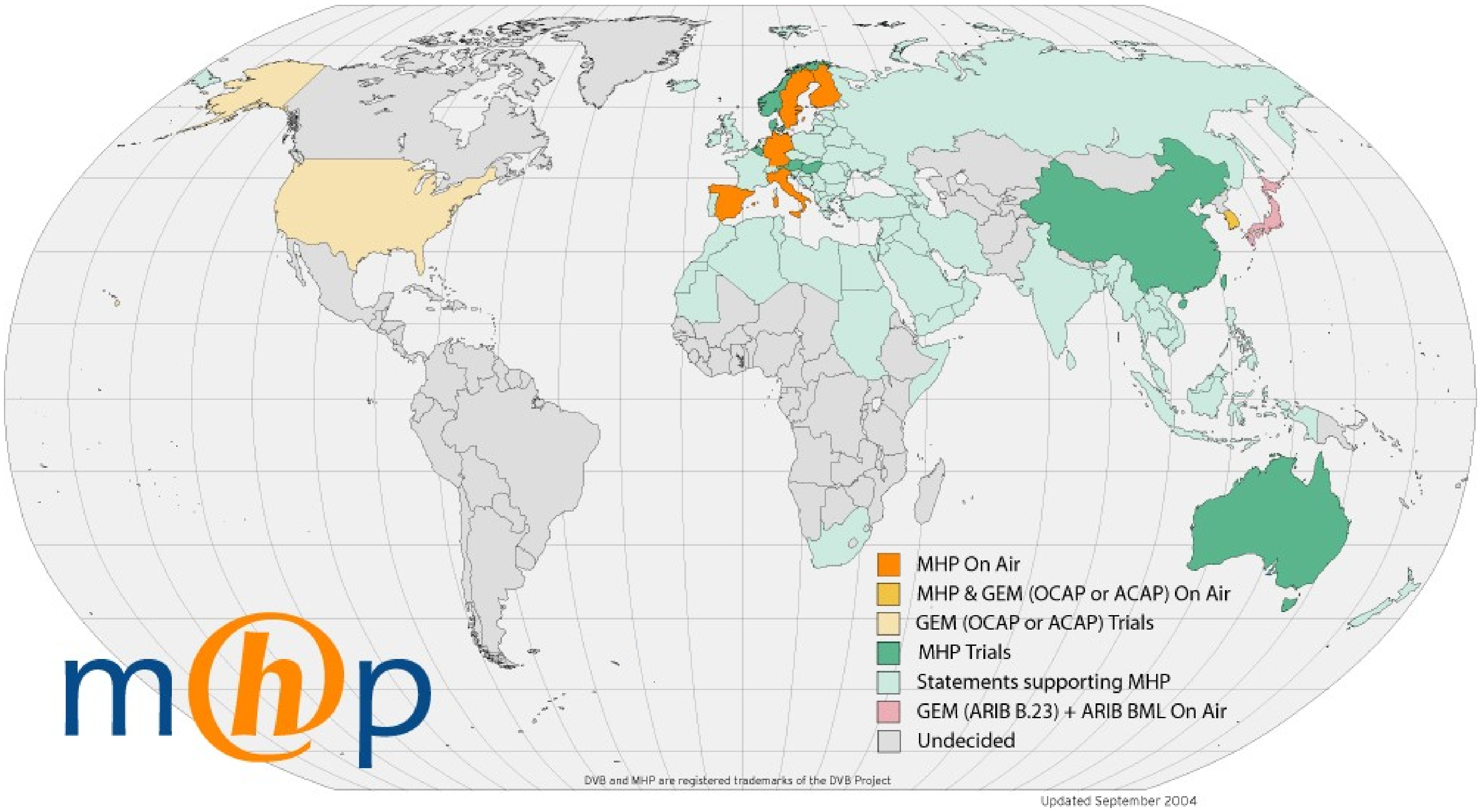
APIs Extensions: Mainly Needed for Monitor Application + MHP Broadcast and MHP Application Signalling

Specific Content Formats: A/V Formats, Closed Captioning

Transport and signalling: ARIB Data Carousel and Application Signalling

Service Information:
ARIB SI

Adoption Map


🟧 MHP On Air
🟨 MHP & GEM (OCAP or ACAP) On Air
🟨 GEM (OCAP or ACAP) Trials
🟩 MHP Trials
🟩 Statements supporting MHP
🟥 GEM (ARIB B.23) + ARIB BML On Air
⬜ Undecided
mhp
DVB and MHP are registered trademarks of the DVB Project
Updated September 2004

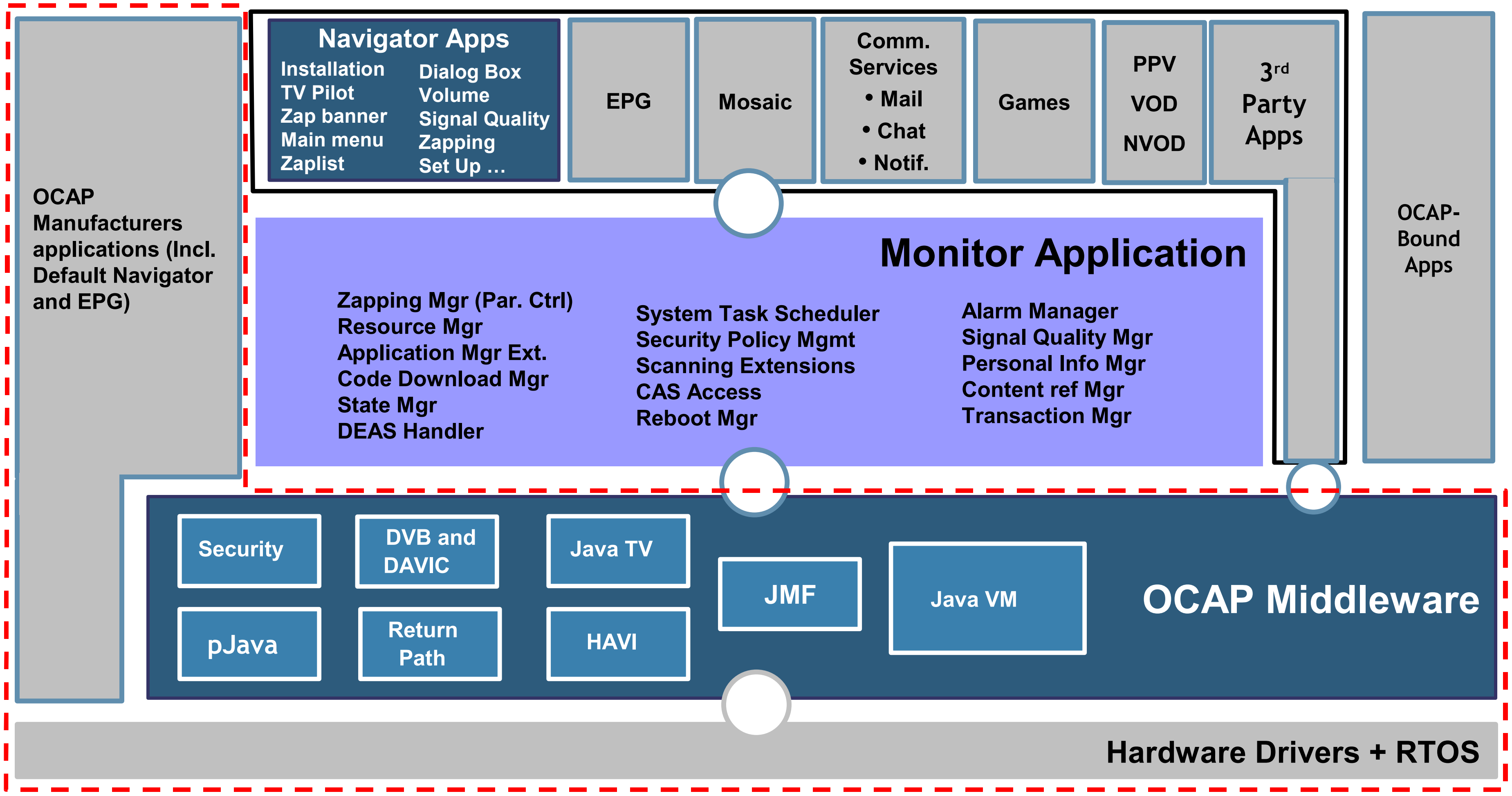
DEMO

Some Interactive Applications

Applications Types

- Service-bound applications
  - Can run only on the channel that they are signaled to run on; Any zapping kill them
- Unbound applications (only OCAP)
  - Not tied to a specific channel and which can be started whatever is the current channel
- Monitor application (only OCAP)
  - An unbound application which is able to supersede resident one (built in the firmware)
  - Provided by the network operator
- System applications (only OCAP)
  - Specific features like Emergency Alert System

OCAP Architecture



MHP Roadmap

- MHP 1.0.3 widely deployed
  - More than 4.2M boxes in Italy
- With MHP 1.1.2, new features have been added
  - HD
  - Support for “drop call” tele-voting
  - Multiple tuners
  - Introduction of broadband networks (IP)
- MHP-PDR specification ready
  - Common part with OCAP DVR specification

OCAP Roadmap

- OCAP 1.0 based on MHP 1.0 via GEM 1.0, right now
- OCAP-specific extensions
  - DVR extension most well mature
  - Home networking extension specification published
- Deployments start in named US cities in 2006
  - Comcast: Philadelphia; Denver; Union, NJ; Boston
  - Time-Warner Cable: New York City; Milwaukee; Green Bay; Lincoln, NE; Waco, TX
  - FCC mention a joint group on technical issues with integrating OCAP in multi-function devices

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

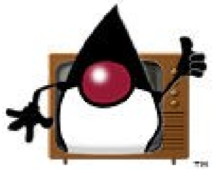
Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

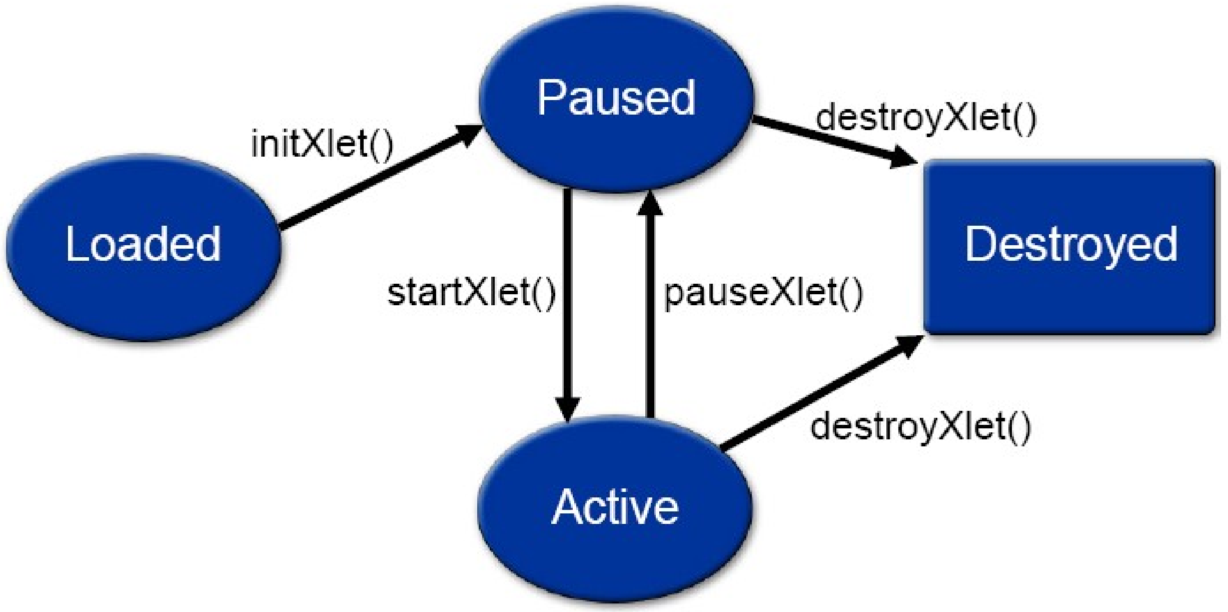
Some Advices



Application Manager

- The main application which manages all resident and downloaded applications
  - Application detection
  - Loading
  - Instantiation
  - Lifecycle management
  - Management of applications within the system
- An MHP/OCAP application is called an “`Xlet`”
  - `javax.tv.xlet` package
- An application is defined by its status

Xlet Lifecycle



- The glue is done via callbacks

Xlet Interface

```
public interface Xlet {
   public void initXlet(XletContext ctx)
                throws XletStateChangeException;
   public void startXlet()
                throws XletStateChangeException;
   public void pauseXlet();
   public void destroyXlet(boolean unconditional)
                throws XletStateChangeException;
}
```

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices

Resource Definition

- A resource is a precious device since it is shared
  - Remote Control (RC) key events
  - Graphic area on the TV screen
  - Tuner
  - Return Channel
- Because several applications can coexist in an STB
  - Arbitration is required for resource accesses
- An application requiring a resource must:
  - Reserve it before using it
  - Release it when it is no longer required
- Reservation can be:
  - Exclusive: nobody else can access the resource for as long as it is reserved (modem)
  - Shared: another Xlet can use it in parallel (RC key events)

DAVIC Resource Mechanism

- Use it to:
  - Manage exclusive or not reservations
  - Be notified of the status of reserved resources
- Package org.davic.resources
  - ResourceClient: implemented by the resource user
  - ResourceProxy: implemented by the resource itself
  - ResourceServer: implemented by the middleware
  - ResourceStatusListener: implemented by the one who wants to be notified about resource status changes
- The means for reserving a resource is not defined in the interface ResourceClient

ResourceClient Interface

```
public interface ResourceClient {
    public abstract boolean requestRelease(
                                    ResourceProxy proxy,
                                    Object requestData);
    public abstract void release( ResourceProxy proxy);
    public abstract void notifyRelease( ResourceProxy pxy);
}
```

Resource Lifecycle

- Creation during the Xlet init phase Xlet::initXlet()
  - Instantiate a ResourceProxy for each resource to use
  - Set the resource parameters
  - Implement the interface ResourceClient to be able to manage a forced release
- Reservation during Xlet activation Xlet::startXlet ()
  - Call a particular method of the ResourceServer to request reservation
- Release at the latest on destruction Xlet::destroyXlet() or before any call to Xlet::pauseXlet()
  - Call a particular method of the ResourceServer to request release

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

Playing With Remote Control

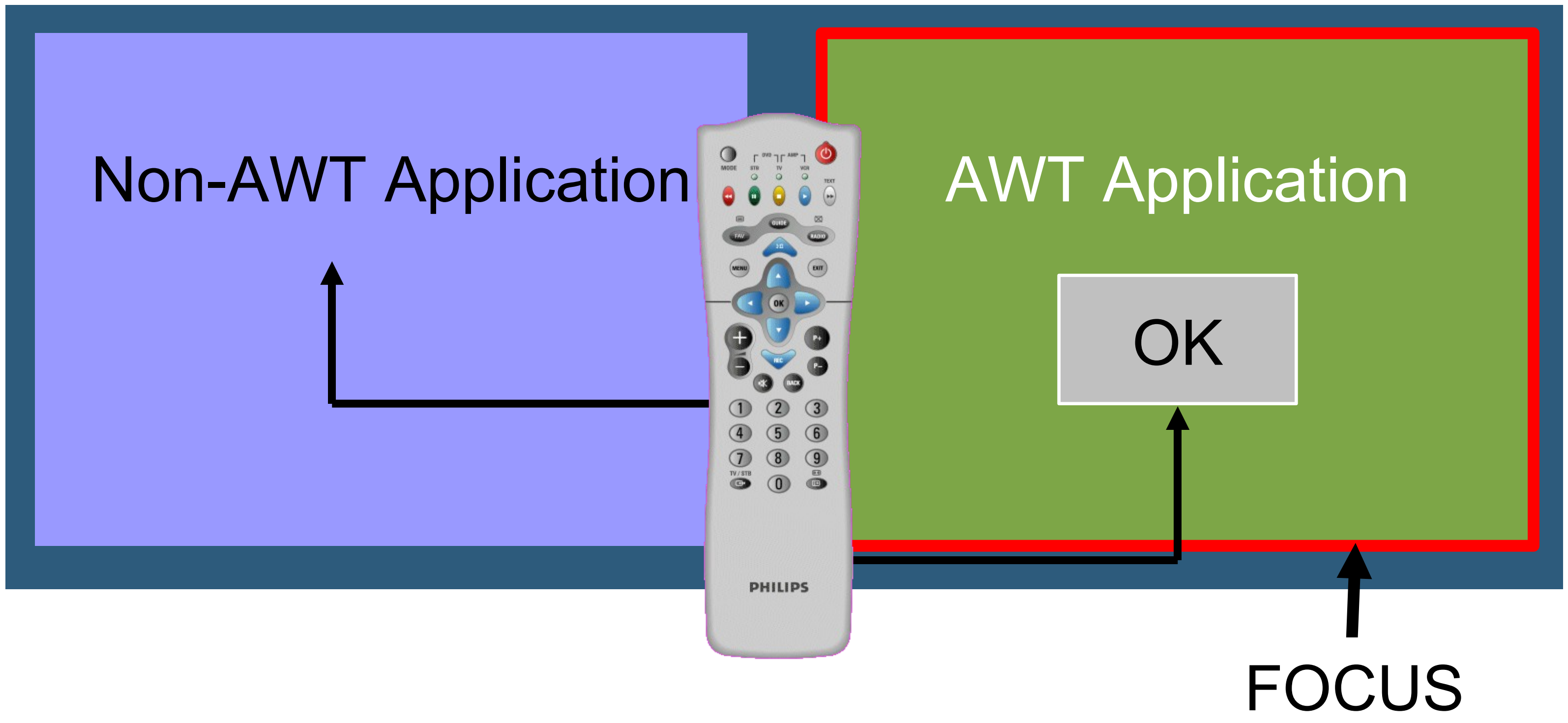
Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices

Events Loop



Org.dvb.event Provides an API for Receiving Events Before They Enter the AWT Event Loop

Event Listeners

- An AWT-based application:
  - Must implement a listener for each event requested via **`addXXListener(XXListener)`**
  - Can request exclusive reservation
  - Receives the events
    - When the graphical component is visible and receives the focus
- A non-AWT application:
  - MUST implement the interface **`UserEventListener`**
  - MUST reserve the events before receiving them
  - Receives the events
    - Without needing the focus

Events Management

- `EventManager`: general event manager
- `UserEventRepository`: a set of events
- `UserEventListener`: listener for a non-AWT application
- `UserEvent`: remote control event itself; it specifies:
  - A unique event code
    - For use when reserving with the eventManager
    - Defined in `java.awt.event.KeyEvent` and `org.havi.ui.event.HRcEvent`
  - A type representing the event, KEY_PRESSED, KEY_RELEASED, or KEY_TYPE

UserEventRepository Object

```
public class UserEventRepository {
    public UserEventRepository (String name);
    public void addKey (int keycode);
    public void removeKey (int keycode);
    public void addAllNumericKeys();
    public void addAllColorKeys();
    public void addAllArrowKeys();
    public void removeAllNumericKeys();
    public void removeAllColourKeys();
    public void removeAllArrowKeys();
}
```

DEMO

How to Catch Remote Control Keys

Remote Key Management

```
// in initXlet(), I order the keys to receive
myBag = new UserEventRepository("any name");
myBag.addAllArrowKeys();
myBag.addAllColourKeys();
myEM= EventManager.getInstance();

// in startXlet(), I ask for receiving the key events
myEM.addUserEventListener(A LISTENER, myBag);

// pauseXlet(), I stop listening to keys
myEM.removeUserEventListener(A LISTENER);

// in destroyXlet(), I get back resources to the system
myBag.removeAllArrowKeys();
myBag.removeAllColourKeys();
myEM.removeUserEventListener(A LISTENER);
```

Remote Key Management

```
// implementation of the listener

public void userEventReceived(UserEvent arg0) {
// I filter on the type, avoiding double events handling
    if (arg0.getType()== KeyEvent.KEY_PRESSED)
     {
     System.out.println("key caught is: "+arg0.getCode());
     if ( arg0.getCode() == KeyEvent.VK_LEFT)
                 …
     if ( arg0.getCode() == KeyEvent.VK_RIGHT)
                 …
     }
}
```

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

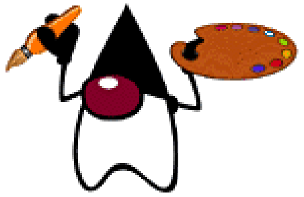
Playing With Remote Control

Using Graphic Devices

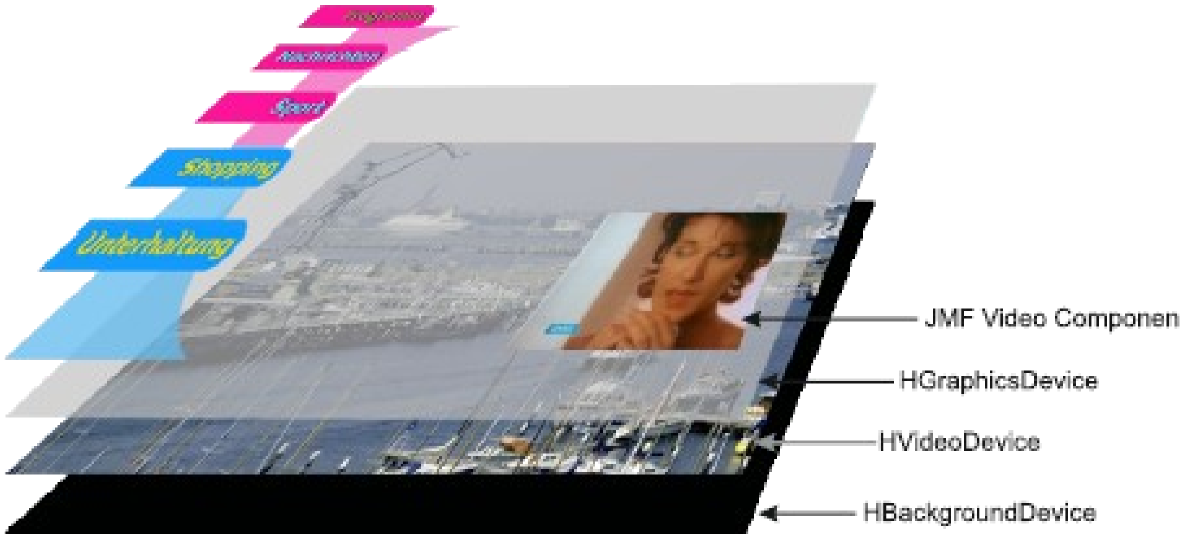
Discovering TV Channels

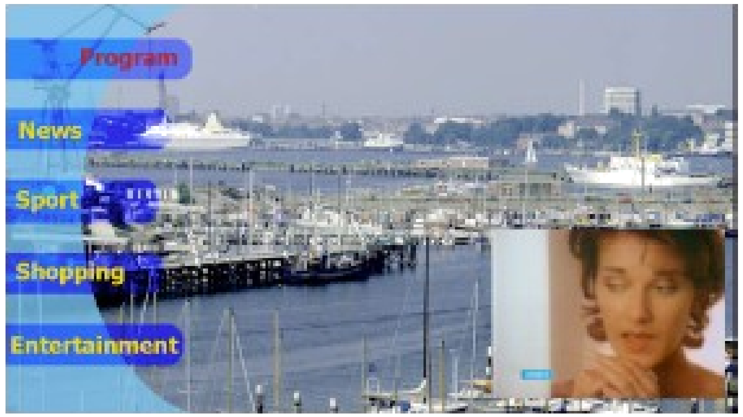
Playing With Media

Some Advices

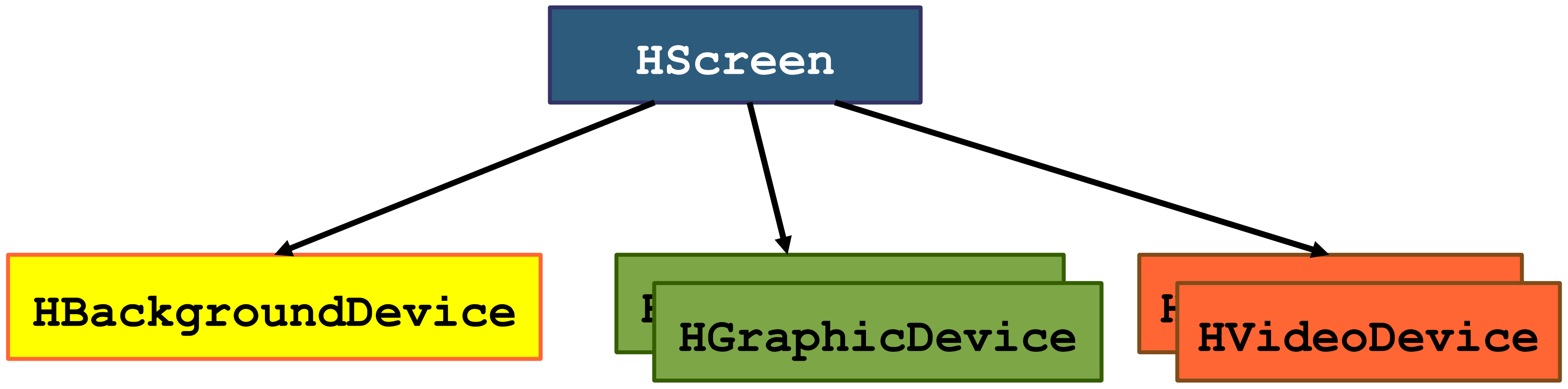


Graphic Layers





Layers Access



- The GUIs are based on the HAVi model
  - Two packages: org.havi.ui and org.havi.ui.event
  - Built on the framework Java AWT (java.awt.*)
  - The AWT components are not included but do have equivalents in HAVi

HScreen Class

```
public class HScreen {
    public static HScreen[] getHScreens()
    public static HScreen getDefaultHScreen()
    public HVideoDevice[] getHVideoDevices()
    public HGraphicsDevice[] getHGraphicsDevices()
    public HVideoDevice getDefaultHVideoDevice()
    public HGraphicsDevice getDefaultHGraphicsDevice()
    public HBackgroundDevice getDefaultHBackgroundDevice()
        …
 }
```

Using a Device

- Each device has to be configured before use
  - `HScreenConfiguration` derived for each device
  - `HScreenConfigTemplate` derived for each device
- To make sure that devices are available and accept the desired properties, it is essential to reserve them
- Set preferences in the Template, with priorities (`REQUIRED,PREFERRED, UNNECESSARY`)
- Ask for the best configuration which fits required properties
- Do not forget to release devices at the end of the application (pause or destroy)

Managing Layers: Configuration

```
// I prepare my objects, in initXlet()
HScreen screen = HScreen.getDefaultHScreen();
HGraphicsDevice device =
                    screen.getDefaultHGraphicsDevice();

// I create template and set preference(s)
HGraphicsConfigTemplate template;
template = new HGraphicsConfigTemplate();

// I ask for image scaling support property
template.setPreference(template.IMAGE_SCALING_SUPPORT,
                template.PREFERRED);

// I request a device corresponding to my preference
HGraphicsConfiguration configuration =
                  device.getBestConfiguration(template);
```

Managing Layers: Usage

```
// before any modification, I request device reservation
// to do in the startXlet()

if (device.reserveDevice( a ResourceClient interface) )
{
    // then I apply my configuration to the device
    try { device.setConfiguration(configuration)  }
    catch (Exception e) {…………}
}

// In destroyXlet(), you must release the resource
device.releaseDevice()
```

DEMO

How to Display a Picture in the Background

Displaying a Picture in the Background

```
// I get the entry point to the layers
HScreen screen= HScreen.getDefaultHScreen();

// I get the background layer
device = screen.getDefaultHBackgroundDevice();

// I create a configuration
HBackgroundConfigTemplate template = new
                      HBackgroundConfigTemplate();
template.setPreference(template.IMAGE_SCALING_SUPPORT,
                template.PREFERRED);
Config = device.getBestConfiguration(template);

// I create my picture
picture = new HBackgroundImage("background.jpg");
```

Displaying a Picture in the Background

```
// in startXlet, I request the background layer
if (device.reserveDevice(a ResourceClient interface))
{
    try
     {  device.setBackgroundConfiguration(config);}
    catch (Exception ex)
     {
        System.out.println("Can't initialize!!");
       device.releaseDevice();
     }
    try {  config.displayImage(picture);}
    catch (java.io.IOException ioe)
    {
        System.out.println("Can't display image - IO
                                                exception");
     }
}
```

0,0

0.0,0.0

Graphic Coordinate Models

- For display in the three graphics layers, MHP/OCAP defines three coordinate models(!)
  - **Screen resolution**
    - Based on the pixel size of the available space
    - Absolute positions: top left corner {0,0} and bottom right corner {640,480} in NTSC
  - **AWT coordinates**
    - Pixel resolution, positions are relative to the root window
  - **Normalized screen space**
    - Independent of any pixel notion
    - Abstract positions: top left corner {0.0, 0.0} and bottom right corner {1.0, 1.0}

320,240

0.5,0.5

640,480

1.0,1.0

HScene Class, Entry-Point to Graphics

- There is no window manager, so it's not possible to use a **`java.awt.Frame`**
- So HAVI defines a class **`HScene`** which contains all the GUI components (**`org.havi.ui`**)
- Applications request an **`HScene`** via a factory **`org.havi.ui.HSceneFactory.GetBestScene()`**
- An HScene is configured in the same way as other screen devices (mechanism of **`HSceneTemplate`**)
- Once the **`HScene`** has been retrieved, it can be manipulated like any other AWT container class
- On destroy, it is essential to call a **`dispose`** method

HSceneTemplate

```
public class HSceneTemplate extends Object {
    // priorities
    public static final int REQUIRED;
     public static final int PREFERRED ;
    public static final int UNNECESSARY ;
    // preferences
    public static final Dimension LARGEST_DIMENSION ;
    public static final int GRAPHICS_CONFIGURATION;
    public static final int SCENE_PIXEL_RESOLUTION;
    public static final int SCENE_PIXEL_RECTANGLE;
    public static final int SCENE_SCREEN_DIMENSION;
    // methods to set preferences
    public void setPreference( int preference,
                                 Object object,
                                 int priority);
}
```

Screen Configuration and Reservation

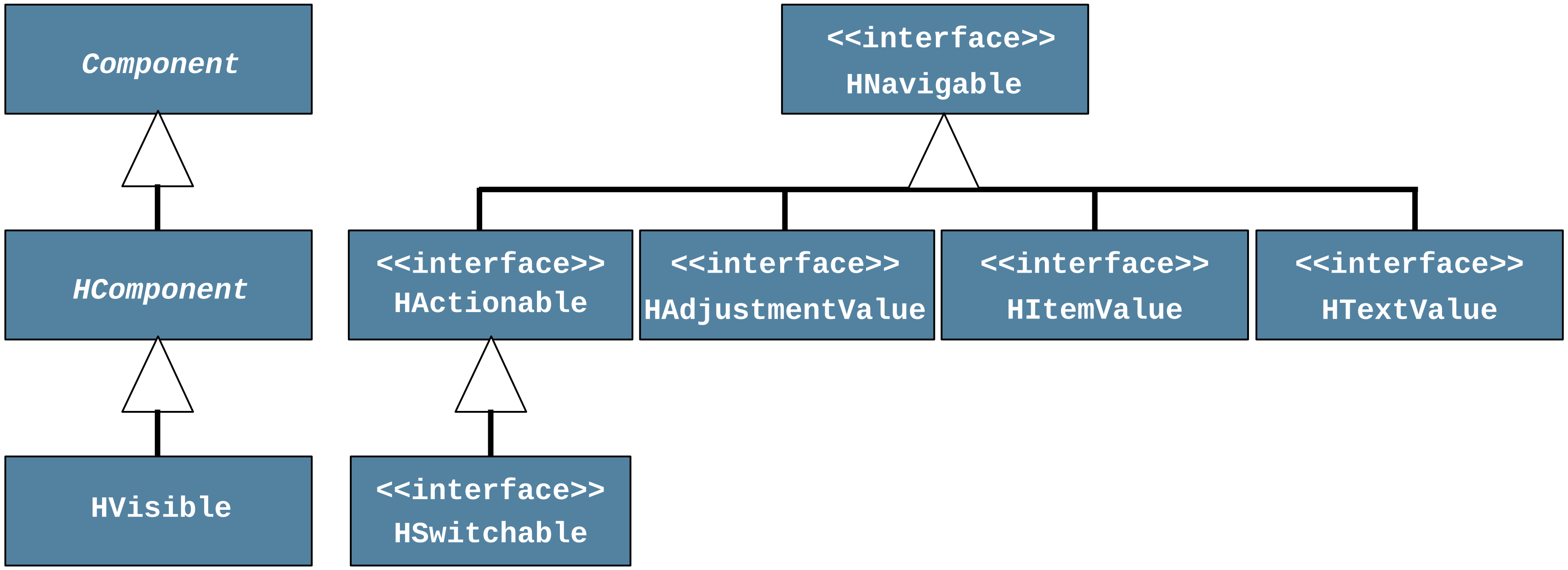
```
HSceneTemplate hst = new HSceneTemplate();

// use of normalized coordinates
hst.setPreference(HSceneTemplate.SCENE_SCREEN_DIMENSION,
                  new org.havi.ui.HScreenDimension(1,1),
                  HSceneTemplate.REQUIRED);
hst.setPreference(HSceneTemplate.SCENE_SCREEN_LOCATION,
                  new org.havi.ui.HScreenPoint(0,0),
                  HSceneTemplate.REQUIRED);

// retrieval of HScene compatible with our preferences
scene = factory.getBestScene(hst);
```

HAVi Widgets

- HAVi widgets are constructed by association of properties (interfaces) that define the « feel »
- The « look » is defined by default and can have private **`Hlook`** interfaces loaded on top



Focus Management

- The property « navigable » lets a component receive the focus
- Navigation is done using the keys arrows in the order of addition of components in the container
- You can modify the navigation direction for each component of type **`HNavigable`**
  - **`Composant.SetMove(KeyEvent, otherComponant)`**
  - **`W.SetMove(KeyEvent.VK_UP, otherWidget)`**
  - **`SetFocusTraversal(Wup, Wdown, Wleft, Wright)`**
  - Setting a direction to NULL blocks navigation in that direction
- When an **`Hscene`** is displayed, it remains only to request focus management
  - **`requestFocus()`**

Font Management

- By default, the font Tiresias is resident in the STB
  - Available in 24, 26, 31, and 36 point
  - Manipulate using `java.awt.Font`
  - A method `setFont` exists for HAVi components
  - `setFont(new Font("Tiresias", Font.PLAIN, 16))`
- Use downloaded fonts if necessary
  - Special format: PFR in `org.dvb.ui.FontFactory`
  - Create a font index file (XML) `dvb.fontindex`
  - Instantiate a `FontFactory`
  - Create a font:

  -
    ```
    Font f = factory.createFont("myFont",
                                Font.PLAIN, 16))
    ```

Graphics and Color Management

- For 2D drawings, you can use the possibilities of **`awt.Graphics`**
  - DVB defines classes that are better adapted to STBs
  - **`DVBGraphics`** provides alpha blending support
- For colors, you can use the class **`awt.Color`**
  - **`org.dvb.ui.DVBColor`** an extension of **`java.awt.Color`** defines transparency levels
  - Alpha component in the range [0, 255] (alpha=0: full transparency)
  - MHP needs at least three levels: 0, 178, and 255
  - MHP defines color palettes on eight bits
- Image formats allowed are GIF, JPG, MPEG-I, and PNG

Building GUI

- During initXlet
  - Configure the required `HScene` using an `HSceneTemplate`
  - Retrieve the `HScene` from the `HSceneFactory`
- During startXlet
  - Create widgets `(HComponents)`
  - Add components to an `HContainer.add()` and/or add this to the `HScene`
  - Display the `HScene` using `setVisible(true)`
  - Request the focus `requestFocus()`
- During destroyXlet
  - Hide the `Hscene`
  - Remove the added components `remove()`
  - Flush all created images `image.flush()`
  - Flush the `HScene.Dispose()`

DEMO

How to Display Widgets on Screen

How to Display Widgets on Screen

```
// I get my Hscene: preparation of the properties
hst = new HSceneTemplate();
hst.setPreference(HSceneTemplate.SCENE_SCREEN_DIMENSION,
                new org.havi.ui.HScreenDimension(1,1),
                HSceneTemplate.REQUIRED);
hst.setPreference(HSceneTemplate.SCENE_SCREEN_LOCATION,
                new org.havi.ui.HScreenPoint(0,0),
                HSceneTemplate.REQUIRED);

// I create my picture to display
picture = Toolkit.getDefaultToolkit().
                         createImage("DukeThumbsUp.jpg");
```

How to Display Widgets on Screen

```
// I retrieve of HScene compatible with our preferences
myScene = HSceneFactory.getInstance().getBestScene(hst);

// now I add my components on screen
button1 = new HTextButton("Btn 1", 20, 40, 80,30);
myScene.add(button1);
button2 = new HTextButton("Btn 2", 20, 80, 80,30);
myScene.add(button2);
icon = new HStaticIcon(picture3, 150,150, 200,200);
icon.setResizeMode(HVisible.RESIZE_ARBITRARY);
myScene.add(icon);

// I draw everything on screen
myScene.setVisible(true);
myScene.requestFocus();
```

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

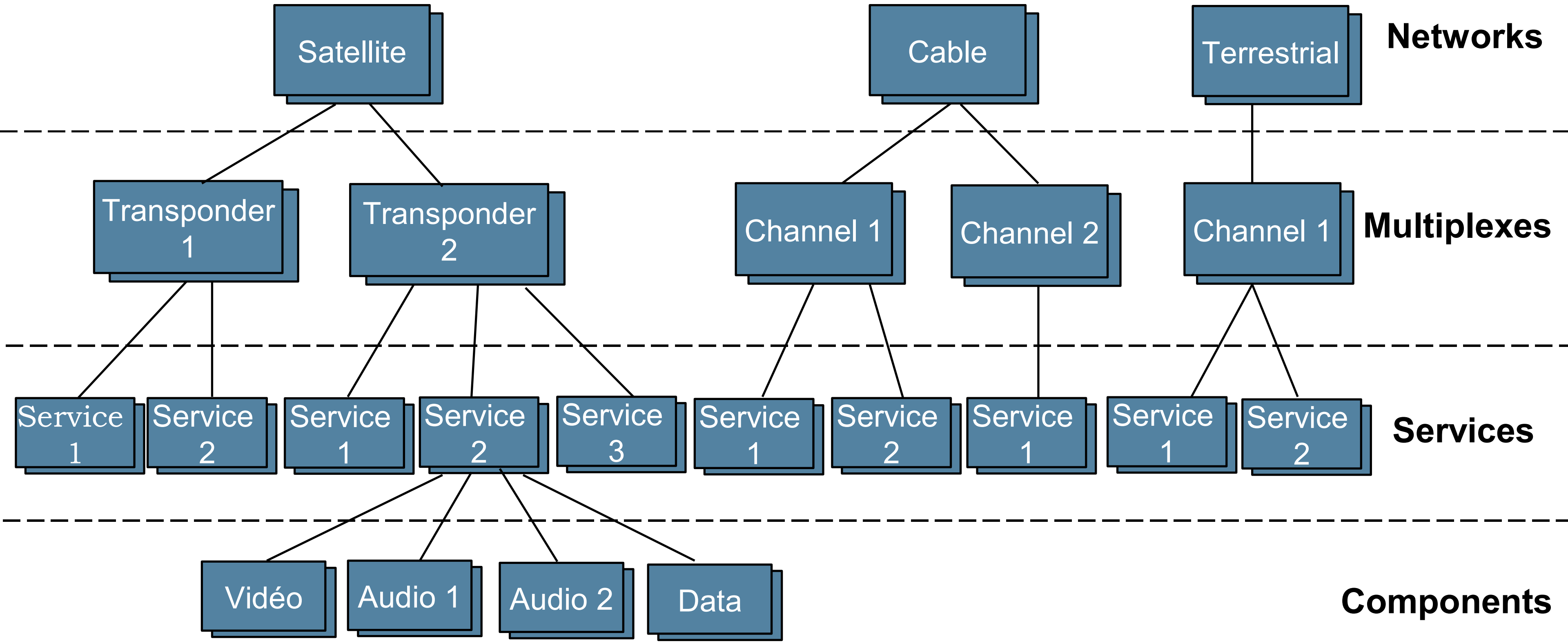
Playing With Media

Some Advices

TV Channels, Notion of Service

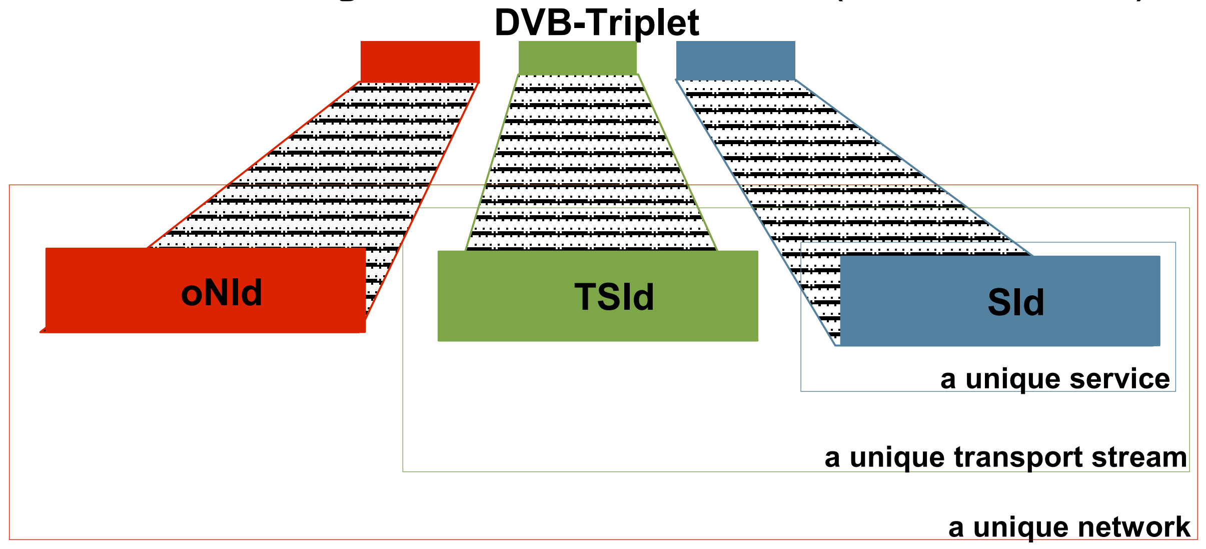
- There are two ways in which the video from a channel can be displayed on the screen:
  - Using a Java Media Framework API-based player to get the audio/video stream
  - Using the Service selection functions
- Zapping amounts to selecting a new service
  - Audio/video display
  - Destruction of any current application if it is service bound
  - Execution of the application signaled as autostart on the new service
- The package `javax.tv` provides Service management
  - Exploration of the service database
  - Selection of services

DVB Network Structure



Service Locator in MHP

- A service is identified by its DVB triplet
- Embedded in a
  `locator dvb://<onID>.<tsID>.<sID>`
  - For Java Media Framework technology: **`javax.tv.locator`** (Locator)
  - For Java TV: **`org.davic.net.dvb`** (DVBLocator)



Getting Services List

- During STB scanning, the Service Information data are stored in FLASH memory
- The class `javax.tv.service.SIManager` enables handling of previously stored information

```
public abstract class SIManager
{
    public abstract Service getService(Locator locator)
                               throws InvalidLocatorException,
                               SecurityException;
    public abstract ServiceList filterServices
                                  (ServiceFilter servicefilter);
}
```

Service Object

```
public interface Service
{
   public abstract String getName();
   public abstract boolean hasMultipleInstances();
   public abstract ServiceType getServiceType();
   public abstract Locator getLocator();
   public abstract boolean equals(Object obj);
   public abstract int hashCode();
}
```

Displaying List of Services

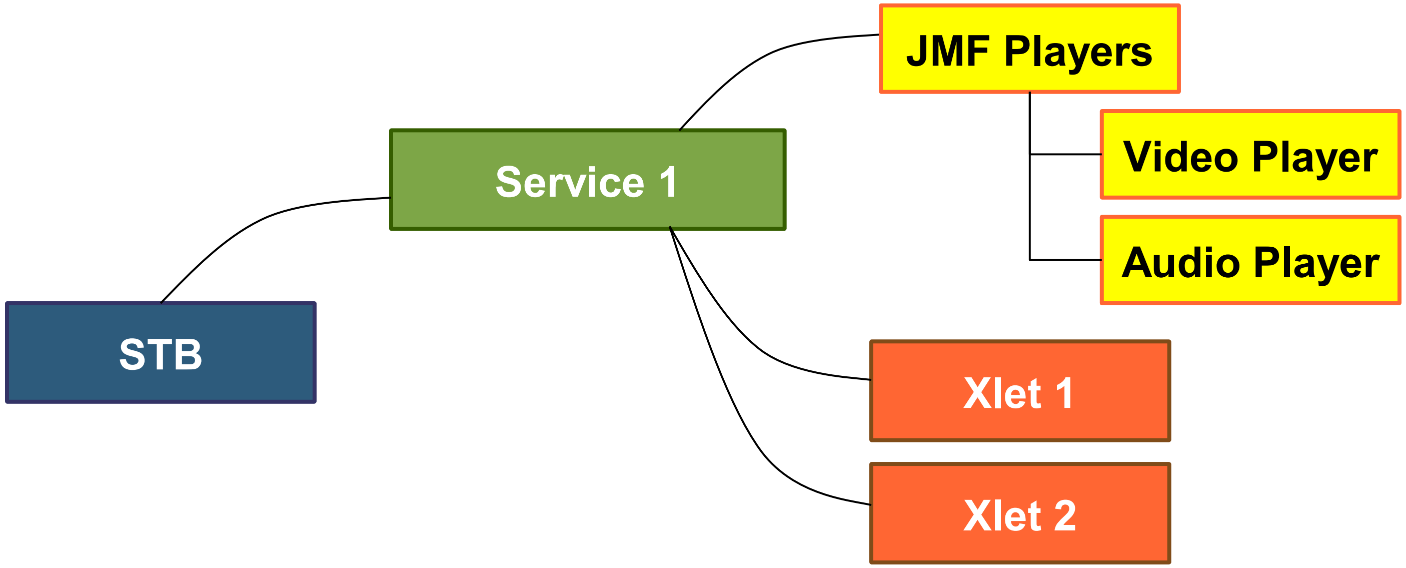
```
ServiceList serviceList=null
SIManager siManager=null;

// I retrieve the ServiceContext
ServiceContext serviceContext =
ServiceContextFactory.getInstance().createServiceContext()
;

// I retrieve the SI database, and filter
siManager = SIManager.createInstance();
serviceList = siManager.filterServices(null);

// I display the list of services
for(int i=0; i< serviceList.size(); ++i)
{
   System.out.println(""+i+": "
             +serviceList.getService(i).getName()); }
```

Accessing to Players



- `javax.tv.service.selection.ServiceContext` is an object covering the set of resources relative to the presentation of a service
- `ServiceContextFactory` allows to get and create ServiceContexts

ServiceContext

```
public interface ServiceContext {
     public void select(Locator[] components) throws
                                  InvalidLocatorException,
                               InvalidServiceComponentException,
                               java.lang.SecurityException
     public void select(Service selection) throws
                               java.lang.SecurityException;
     public void stop() throws java.lang.SecurityException;
     public Service getService();
     public
   ServiceContentHandler[]getServiceContentHandlers()
                              throws java.lang.SecurityException;
}
```

DEMO

Displaying List of Services and Zapping on a Specific One

Displaying List of Services and Zapping

```
// I retrieve the ServiceContext
ServiceContext serviceContext =
ServiceContextFactory.getInstance().getServiceContexts()[0];

// I retrieve the SI database, and filter
siManager = SIManager.createInstance();
serviceList = siManager.filterServices(null);

// I display the list of services
for(int i=0; i< serviceList.size(); ++i)
 {System.out.println(""+i+": "+
serviceList.getService(i).getName()); }

// I select the first service
serviceContext.select(serviceList.getService(0));
```

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

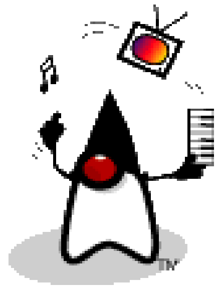
Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices



Java Media Framework-Based Player

- Retrieve the Player from the current service
  - `ServiceContext::getServiceContentHandlers()` returns the set of MediaHandler managing the Service components
  - If the service shows A/V, the first element of the table provides access to the current player
- Lifecycle
  - During **`init`**: retrieve the `Player` from the `ServiceContext`
  - During **`start`**: retrieve the necessary controls, declare the listeners
  - During **`pause`**: restore the initial context
  - During **`destroy`**: restore the player `deallocate()`

Java Media Framework-Based Controllers

- A number of controllers exist in Java Media Framework technology and are complemented by the iTV specific ones
  - Because of the broadcast nature of the media used in an STB, the controllers acting on a time basis are inoperative
  - 11 controllers have been specified with different objectives
  - `javax.tv.media.MediaSelectControl`
  - Choose the I/Os presented in a service; useful for multi angle/language
  - `org.dvb.media.BackgroundVideoPresentationControl`
  - Position the Player on the screen with its size
  - `org.dvb.media.VideoTransformation`
  - Sets video size (resizing)
  - `org.davic.media.LanguageControl`
  - Change the language for subtitles/audio track
  - `javax.media.GainControl`
  - Volume control

DEMO

Resizing and Moving the Video

Resizing and Moving the Video

```
Player
p=(Player)serviceContext.getServiceContentHandlers()[0];

// I retrieve controller to change size and position
BackgroundVideoPresentationControl
b=(BackgroundVideoPresentationControl) player.getControl(
     "org.dvb.media.BackgroundVideoPresentationControl");

// I prepare new parameters to set
VideoTransformation video= new VideoTransformation(
             new Rectangle(640, 480),0.5F,0.5F,
             new HScreenPoint((float)0F,(float)0F));

// I apply parameters to the controls
b.setVideoTransformation(video);
```

Agenda

What Is MHP/OCAP?

Interactive TV Application

Resource Management

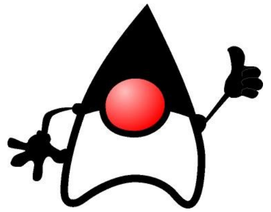
Playing With Remote Control

Using Graphic Devices

Discovering TV Channels

Playing With Media

Some Advices



Some Advice: Lifecycle

- Like at dinner: do not be a boor
  - MHP defines a set of Java-based APIs
  - MHP defines an application model
- All methods enable the middleware to:
  - Detect and find an application
  - To load, start, stop, pause, end an application
  - Have mechanisms for handling many applications
- This life cycle must be followed

Advice 1: Respect your host…

…follow its rules

Some Advice: Platform Constraints

- Like at dinner: do not ask for more than what you have on your plate
  - MHP defines a set of minimum available features
    - MPEG decoder, graphic components, java interpreter
  - MHP defines a set of behaviors
    - Screen size, RC keys mapping, color mapping, characters set...
  - MHP defines a minimum of constraints
    - Image formats, font sizes
    - Level transparency, min memory, permanent storage

Advice 2: Respect your host's generosity

Some Advice: Resources

- Like at dinner: do not take all the food!
  - The STB offers resources for the application
    - Remote control events, video player, modem line, graphic layer, tuner....
    - Most of these are scarce resources
  - Because several applications can coexist in an STB...
    - Arbitration is required for resource accesses
  - Because at any moment, the STB could take the lead...
    - An application must be aware of system needs
  - An application requiring a resource must follow rules

Advice 3: Do not be selfish…

…share resources

Some Advice: Exceptions

- Since you do not know the characteristics of the STB running your application, error management is crucial
  - Errors could occurs because:
    - You are doing prohibited things
    - The system is not able to do what you expected it to
  - Always use try/catch mechanisms, if not...
    - You could alter the other applications
    - You could make the system lose memory
    - You could hang the system

Advice 4: Catch all exceptions, be safe for others

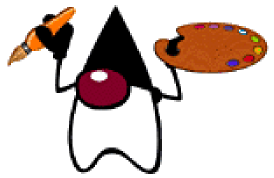
Some Advice: Memory

- STB does not have a lot of memory; it is shared with the middleware and all other applications
- Help the memory manager to save space
  - Force object references to null when no longer necessary
  - Break cross-references before losing instances
  - Avoid explicitly calling garbage collector
  - Free all loaded pictures, opened files...

Advice 5: Control your memory usage

Some Advices: Graphics

- GUI layer based on TV-centric AWT components
- Background layer permits "wallpaper"
  - Few widgets since we are doing TV and not "PC
    - No windows manager, no need of sliders, complex widgets
    - End user is passive with only a remote control
    - Drive your look-and-feel
    - Focus management
    - Check default behaviors: widgets properties must be completely set
    - Inform the user that your application is running



Advice 6: Design light and sexy for TV

Summary

- MHP/OCAP are Java technology-based middleware for TV
- Java Media Framework API, HAVi, and other APIs help to develop quickly
- But, coders must assume that a STB is a very constricted device
- Applications must adapt to rules and mechanisms to run correctly
- Including HD, Home Networking, these standards are the future of our devices life
- In any cases never forget that it is …TV!

For More Information

Web Links

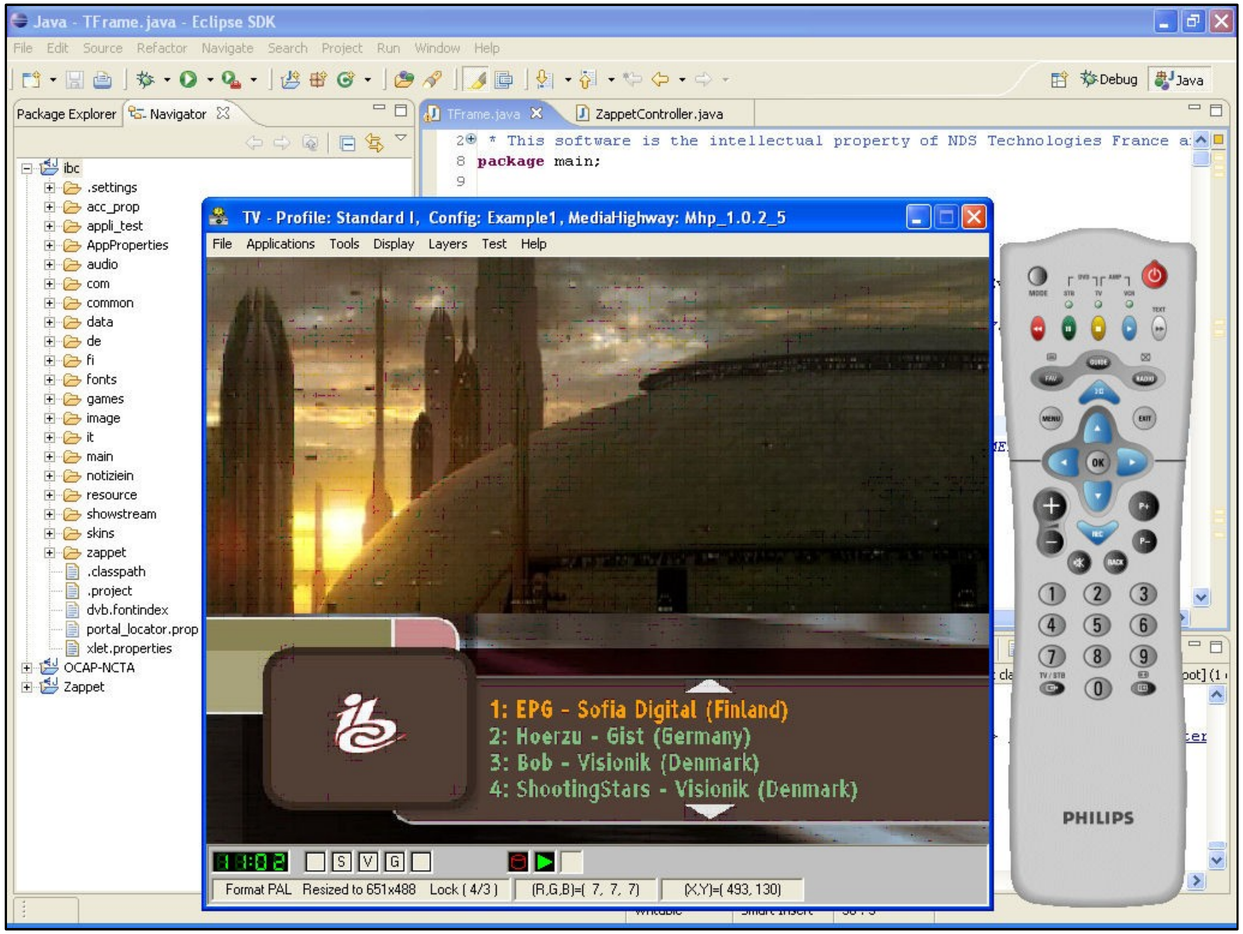
- www.dvb.org
- www.mhp.org/documentation
- www.opencable.com/specifications

Forums

- The official MHP forum
  - http://www.mhp.org/forum/forum.xml
- Java TV Technology Developer forum
  - http://forum.java.sun.com/forum.jsp?forum=36

For Really More, Try It for Free!

www.nds.com/middleware

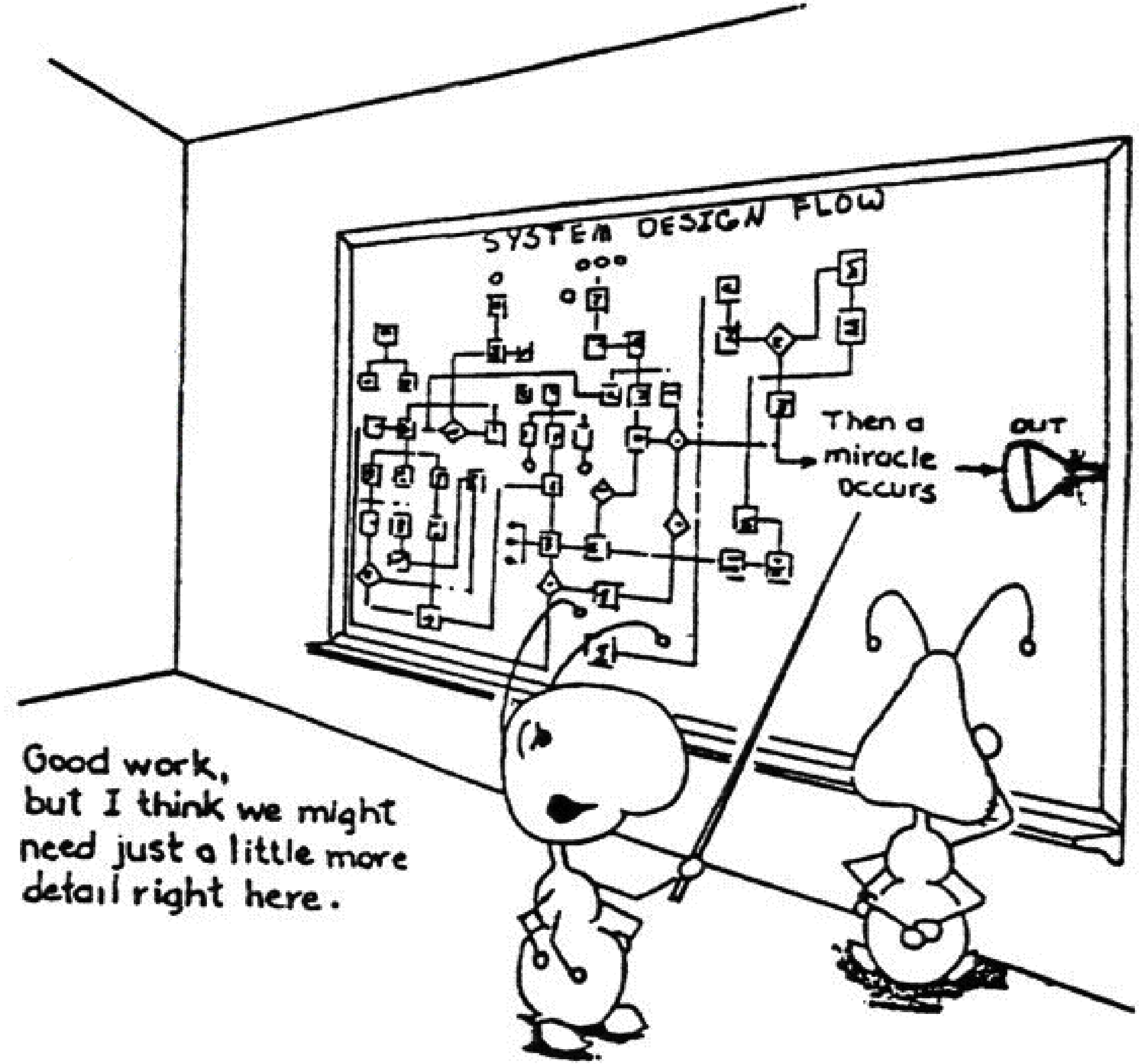


```
 * This software is the intellectual property of NDS Technologies France a
package main;
```

```
1: EPG - Sofia Digital (Finland)
2: Hoerzu - Gist (Germany)
3: Bob - Visionik (Denmark)
4: ShootingStars - Visionik (Denmark)
```

```
11:02
Format PAL  Resized to 651x488  Lock ( 4/3 )  (R,G,B)=( 7, 7, 7)  (X,Y)=( 493, 130)
```

Q&A


SYSTEM DESIGN FLOW
Then a miracle occurs
OUT
Good work, but I think we might need just a little more detail right here.

MHP/OCAP iTV Applications in a Nutshell

Cédric Monnier

Senior Product Manager
NDS
www.nds.com/middleware.html

TS-4255