Java Technology for Interactive TV: Developing and Deploying Effective OCAP Applications

Anne Dirkse

Training Engineer
Vidiom Systems, Inc.
www.vidiom.com

TS-0697

Presentation Goals

What You Will Learn in this Presentation

Discuss the fundamentals of building OCAP Applications and the issues surrounding application development and deployment

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

Application Capabilities

OCAP User Interfaces

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

Application Capabilities

OCAP User Interfaces

What Is OCAP?

The OpenCable Application Platform

- Java™ technology-based middleware platform for Digital TV
- Developed by CableLabs
- Allows Java technology-based applications to be delivered with television content
- Allows applications like the Electronic Program Guide (EPG) to be written (once) in Java platform
- Allows cable operators to distribute applications nationwide, to diverse platforms

New Terms

The vocabulary to get you going

- MSO
  - Multiple Systems Operator; a Cable Operator
- CableLabs
  - Non-profit consortium working on behalf of MSOs
- EPG
  - Electronic Program Guide
- OCAP Implementation/ Middleware/ Stack
  - Synonyms; The OCAP layer on the Host Device

Related Specifications

GEM and MHP

- The Multimedia Home Platform (MHP)
  - MHP is a worldwide Java technology-based middleware platform for digital TV; based on Java TV™ API
  - Forms the foundation for OCAP
- Globally Executable MHP (GEM)
  - An effort to keep MHP-based middleware specifications compatible with one another as the technology evolves
  - OCAP is a GEM-based specification
  - MHP and Java technology based Blu-ray Disc (BD-J) are also GEM-based

The OCAP 1.0 Platform

OCAP Foundations and Application Types

- Based on Java 1.1.8 platform/PersonalJava™ Platform
  - Additional APIs from a multitude of sources
- Two types of applications
  - Bound
    - Applications like play-along game shows or live sports stats
    - Tied to broadcast content; app killed at channel change
  - Unbound
    - Applications like the EPG
    - Life cycle unaffected by channel change

Services

Service Definition and OCAP Extensions

- An integrated collection of audio, video, and program data that is presented together
- Object representation defined in Java TV API
- What we typically think of as a TV Channel
- Bound applications are killed when new Services are selected
- OCAP defines applications that can exist outside of a broadcast service
    - Unbound apps are associated with an **`org.ocap.service.AbstractService`**

DEMO

A Bound Application

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

Application Capabilities

OCAP User Interfaces

Constrained Devices

- OCAP devices are constrained devices
- Memory, processor and storage requirements are only loosely defined in the OpenCable specs
- Applications should use caution with
  - Processor intensive code
    - Animations
    - Transparency
  - Operations with large memory requirements
    - Large zip files
    - Large amounts of application data

The Monitor Application

- OCAP defines a privileged application called the Monitor Application
  - Provided by the MSO
  - Has a special set of permissions
- The Monitor Application has dynamic control over
  - The applications that are allowed to run
  - The priority of applications
  - The permissions applications are granted
  - The resources applications can access

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

Application Capabilities

OCAP User Interfaces

Application Life-cycle Control

The Application Manager and the Xlet Interface

- OCAP Applications are managed applications
- Application Manager in the OCAP stack controls the lifecycle of applications
- OCAP Applications must implement the `javax.tv.xlet.Xlet` interface

A Simple Xlet

```
class HelloOCAP implements javax.tx.xlet.Xlet {

public void initXlet(XletContext context)
            throws XletStateChangeException {
}

public void startXlet()
            throws XletStateChangeException {
}

public void pauseXlet() {
}

public void destroyXlet(boolean forced)
            throws XletStateChangeException {
}
}
```

Life-cycle Etiquette

- Don’t initialize your application in the constructor; that's what the `initXlet` method is for
  - But especially don’t reserve scarce resources in a constructor
- Return from life-cycle methods in a timely manner
- Paused applications should release scarce resources

Application Permissions

- Applications may be signed or unsigned
  - Signed applications have greater default permissions
- Signed applications can request extra permissions in a Permission Request File (PRF)
  - XML file in the same directory as the Xlet
- PRF Permissions include network and file system access

Application Permissions

- Applications can be denied requested permissions by the Monitor Application
- Applications should ensure that they have been granted the permissions that they requested
- Applications should respond appropriately if these permissions are denied

DEMO

A Permission Request File

Scarce Resources

Managing access to scarce resources

- OCAP uses the DAVIC resource framework
- Scarce resources include
  - The tuner
  - MPEG section filters
  - An exclusively-reserved keycode
- Framework defines several interfaces
  - **`ResourceClient`**: the entity that wants the resource
  - **`ResourceProxy`**: a proxy to the resource
  - **`ResourceStatusListener`**: listener for resource availability

Working With Scarce Resources

- Applications that utilize scarce resources should be built with the realization that the resource they request may or may not be granted to them
  - The request may be denied outright
  - May lose the scarce resource to another application
- Should have a contingency plan to operate in a modified mode or exit if a required resource is unavailable

DEMO

The Resource Contention Framework

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Capabilities

Application Delivery

OCAP User Interfaces

Application Signaling

Telling OCAP how and when to launch apps

- OCAP devices are told when and how to launch apps by tables carried in the Transport Stream
  - Application Information Table (AIT)
    - Signals bound applications
  - eXtended Application Information Table (XAIT)
    - Signals unbound applications
- Describes the application’s name, identifiers, base directory, main class, etc.
- Additional ways to register manufacturer and MSO applications

Delivery of OCAP Applications

- OCAP Applications can be delivered to the Host Device by a variety of means
- Can be installed by the Host Manufacturer
- Can be delivered over the network
  - Over HTTP
  - In an In-band carousel
  - Downloaded applications may be stored on the host device
- Can be delivered by an MSO-specific means and installed by the Monitor Application

Object Carousels

- OCAP Applications may be delivered in object carousels
  - A read-only, broadcast file system
  - May involve significant latencies in file delivery
- MHP defines a `DSMCCObject` in `org.dvb.dsmcc` that extends `java.io.File`
  - Allows applications to asynchronously load files from a carousel
  - `AsynchronousLoadingEventListener` notified when the file is fully loaded

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

Application Capabilities

OCAP User Interfaces

Service APIs

- OCAP applications use the `javax.tv.service` APIs to find, analyze and select `Services`
- The `ServiceContext` associates a `Service` with the implementation
  - Every application is associated with a `ServiceContext`
- `Services` contain `ServiceComponents`
  - represent the various Elementary Streams
    - Things like audio, video and data
    - Applications can select `ServiceComponents` using the `ServiceContext` (i.e., to change the audio presenting)

Media in OCAP

- OCAP’s media APIs are based on Java Media Framework 1.0 API
- DAVIC extends the Java Media Framework APIs to include a `MediaLocator` that ties into broadcast `Locators`

Network Access

- OCAP devices have a guaranteed return path for two-way communication
  - This is not an open connection to the internet
    - MSO controls access to outside resources
- OCAP Applications use the java.net APIs to access network resources
- No need to configure the Return Channel interface as in MHP
- Support for secure sockets provided by Java Secure Socket Extension (JSSE) software

Session Agenda

What is OCAP?

Application Considerations

OCAP Applications

Application Delivery

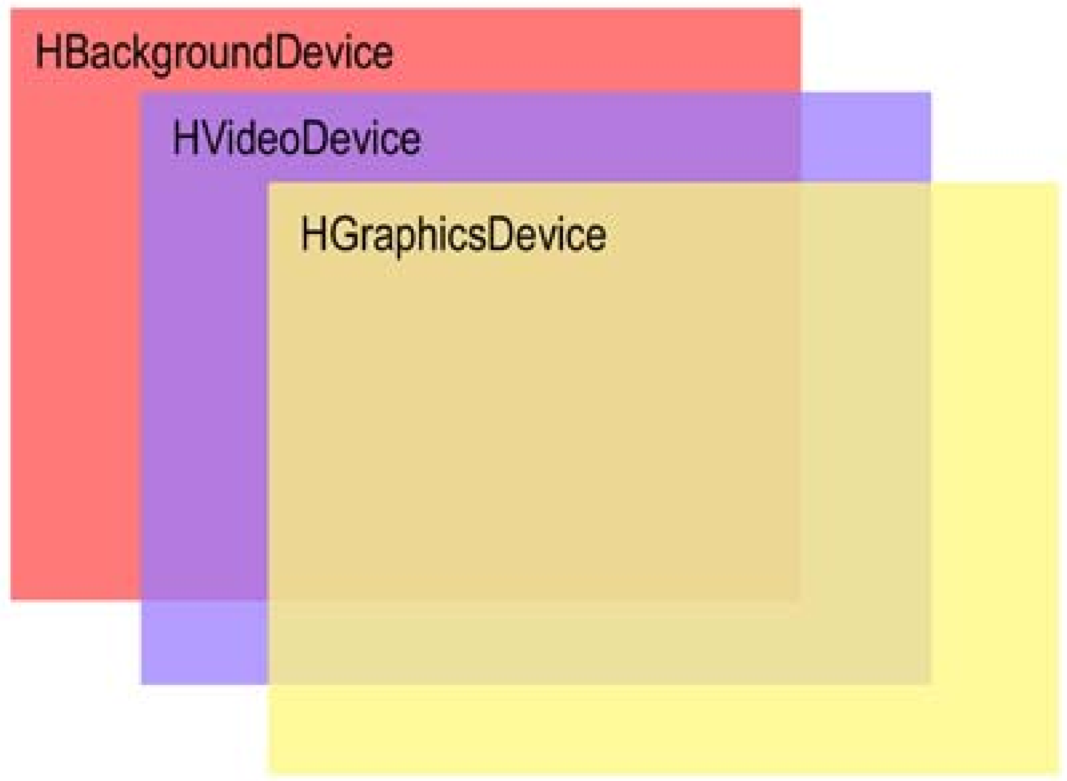
Application Capabilities

OCAP User Interfaces

TV Displays

- TV Displays != PC Displays
- OCAP requires support for different resolutions and aspect ratios
- Not every row of pixels is guaranteed to display
  - Thin lines in graphics or fonts could be omitted on some displays
- The full screen area may not display
  - Application developers should keep critical graphics and Components in the ‘Safe Zone’

Graphics Planes



Images

- OCAP supports three image types:
  - GIF (no support for animated GIFs), JPEG and PNG
- Applications should consider
  - Size
  - Re-usability
  - Transparency
- Images should be loaded using a `MediaTracker`
- **AWT `Toolkit`'s `getImage` returns a cached image if possible; new method `createImage` does not**

Colors

- `java.awt.Color` is implemented in OCAP
  - Color as of 1.1.8 does not support transparency
  - Extended by `javax.tv.graphics.AlphaColor` **to add transparency support**
  - `AlphaColor` extended by `org.dvb.ui.DVBColor`
- **Applications should choose colors with RGB values of 240 or less**
  - **Highly saturated colors can burn into the screen**
  - **Can be difficult to read**

Transparency

- OCAP supports transparency between planes
- HAVi components have transparent backgrounds by default
  - Allows for video to show through behind the graphical components
- Host devices are only required to support 0, 30 and 100% transparency
- Application developers should consider this when looking at factors like readability

HAVi Components

- The HAVi Level 2 User Interface specification defines a set of lightweight components that are used in OCAP devices
- HAVi Components provide support for
  - Transparency
  - A pluggable look and feel
  - Transfer of focus related to the arrow keys on a remote control
- HAVi defines the `HScene`, the root container of an OCAP application

Events from the Remote Control

The AWT and HAVi Event Frameworks

- OCAP requires a connected remote control
- A keyboard is optional in OCAP
- `KeyEvent` represents events from the keyboard
- `KeyEvent` is extended in HAVi and OCAP to define remote control keycodes
- `org.ocap.ui.event.OCRcEvent` includes standard 1.1.8 keycodes plus 55 HAVi keycodes and 27 OCAP-specific keycodes

The DVB Event APIs

Generating KeyPress events in non-focused apps

- To receive remote control events via AWT a component must be focused
- Non-focused applications may want to respond to a keypress
  - i.e., the EPG responding to the 'Guide' button
- The DVB Event Framework allows for this
- Applications place events or keycodes they are interested in into a **`UserEventRepository`**
  - Applications register the repository with the **`EventManager`**

DEMO

HAVi Components/DVB Events

Summary

- OCAP allows Java technology-based Applications to run on Digital Set Top Boxes and TVs
- OCAP shares many APIs with other GEM-based platforms
- The Monitor Application controls application resources and priorities
- OCAP Application developers should consider the memory, processor and display requirements of the application
  - As well as its means of delivery

For More Information

Additional OCAP Sessions

- TS-0011—OCAP: Summary of Technical Features and APIs
- TS-5931—OCAP Roadmap and Future Interactive Services on Cable TV
- BOF-5724—TV Technology Q&A

For More Information

OCAP URLs

- http://www.opencable.com
  - OCAP and related specifications
- http://www.vidiom.com/support/forums.html
  - Vidiom support forum; OCAP questions and answers
- http://forum.java.sun.com/forum.jspa?forumID=36
  - Sun JavaTV Developer Forum

Q&A

Anne Dirkse, Vidiom Systems, Inc.

Java Technology for Interactive TV: Developing and Deploying Effective OCAP Applications

Anne Dirkse

Training Engineer
Vidiom Systems, Inc.
www.vidiom.com

TS-0697