The Java TV 1.0 API: Technical Overview

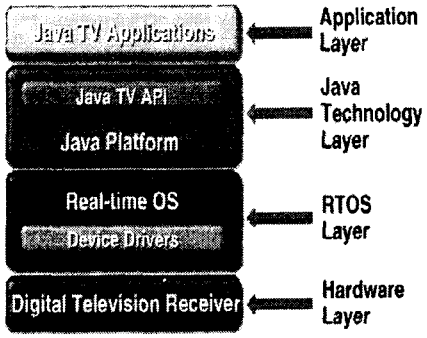
James Van Loo
Television Execution Environment Lead
Sun Microsystems

Key Topics

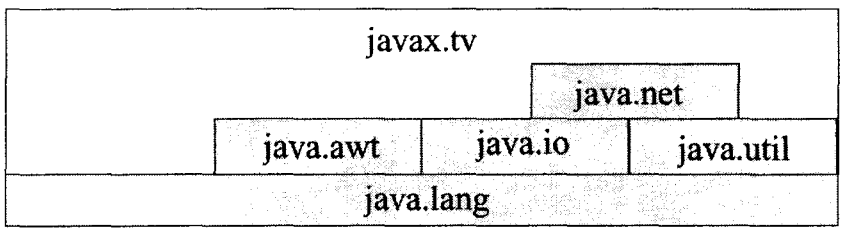
- Application Execution Environment
- Application Life Cycle
- Service Metadata
- Service Selection
- Media Stream Control
- Core Packages Extensions

Application Execution Environment

- Broadcast Platform
  - Virtual Machine
  - Broadcast Packages (javax.tv and javax.media)
  - Core Java Packages (java.io, java.lang, java.net, java.util)
  - Lightweight Widgets (java.awt with org.havi.ui)



Application Execution Environment



Signature Dependence

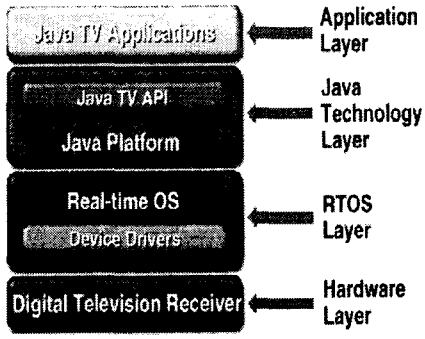
Package.Construct	Inheritance	Return	Args	Exception
java.awt.Color	X	X	X	
java.awt.Container		X		
java.awt.Dimension		X		
java.awt.Rectangle		X	X	
java.io.File	X			
java.io.Serializable	X			
java.io.IOException	X			X
java.lang.Object	X	X	X	
java.lang.String		X	X	
java.lang.Exception	X			

Signature Dependence

Package.Construct	Inheritance	Return	Args	Exception
java.net.InetAddress		X		
java.security.BasicPermission	X			
java.security.Permission	X		X	
java.util.EventListener	X			
java.util.Date		X	X	
java.util.EventObject	X			
javax.media.Control	X			
javax.media.Controller		X	X	
javax.media.Player	X			
javax.media.protocol.PushSourc	X			

Broadcast Environment

- Broadcast Platform
  - Operating System
  - Tuner Control
  - Demux Control
  - Conditional Access
  - Media Pipeline
  - Service Information Database



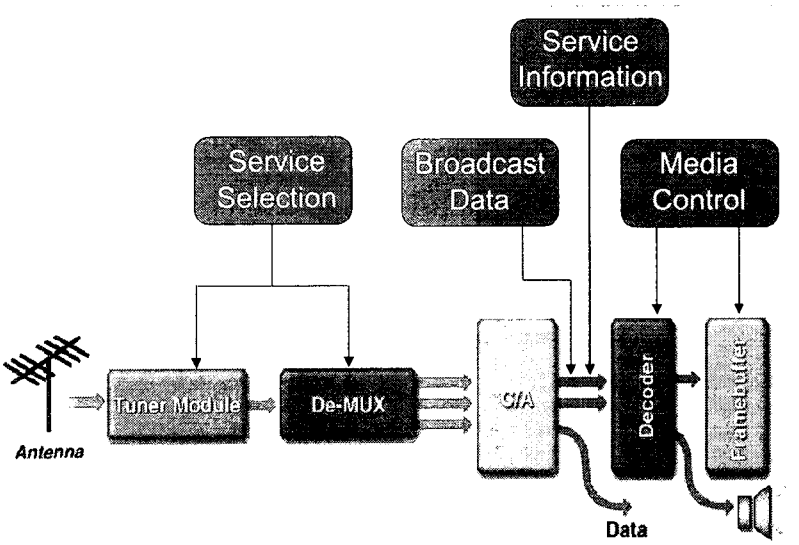
JavaOne

Broadcast Package Structure

Broadcast Package Scope

JavaOne

Functional Components



JavaOne

Execution Substrate

Procedural Application Environment

- Core Java Packages
  - java.io, java.lang, java.net, java.util
- Images, Graphics, Windows
  - java.awt.Color, java.awt.Component, java.awt.image
- Broadcast Packages
  - javax.tv
- Media Pipeline Packages
  - javax.media, javax.media.protocol

JavaOne

Functional Scope

The broadcast packages support:

- Application Life Cycle
- Service Metadata
- Service Selection
- Broadcast Data Control
- Media Pipeline Control

Package Scope

Package	Interfaces	Classes	Exceptions	Total
javax.tv.carousel	1	2		3
javax.tv.graphics		2		2
javax.tv.locator	1	1	2	4
javax.tv.media	3	6		9
javax.tv.media.protocol	1		1	2
javax.tv.net	1			1
javax.tv.service	9	7	1	17
javax.tv.service.guide	5	2		7
javax.tv.service.navigation	9	9	2	20
javax.tv.service.selection	4	10	3	17
javax.tv.service.transport	11	5		16
javax.tv.util	1	3	1	5
javax.tv.xlet	2		1	3
	48	47	11	106

Locators

Locators

- Mechanism to Reference Services and Resources
- Opaque References to
  - Broadcast Carousels (files)
  - Service Metadata
  - Services (collections of resources)
  - Service Components (audio, video, data)

Locators

Locators

- Opaque References to Services and Resources
- which the platform Creates from strings:
  - LocatorFactory.create(String) -> Locator
- or the platform converts to strings:
  - Locator.toExternalForm() -> String

Access to Broadcast Resources

Resource Access

- The framework does not define resource allocation policy: the network and the platform define such policy
- The platform enforces the policy
- Invoke and Refuse Model
- Resource access is visible to the application through exceptions

Application Life Cycle Concepts

Application Life Cycle Model

Application Life Cycle

Goal: Define a formalism for the execution of broadcast applications

- Learn from other domains (e.g. applet)
- Separate execution state machine from other functions (access to resources, media, windows)
- Develop broadcast specific execution state machine

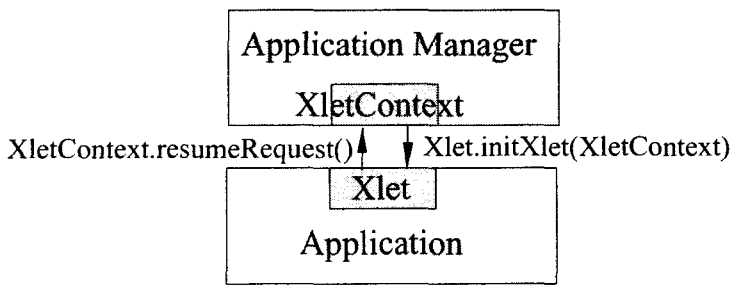
Application Life Cycle Components

- Application Manager
- Xlet
- XletContext
- Execution State Machine

Application Manager

- The Application Manager is a concept, not an concrete class
- The Application Manager controls the Xlet state machine
- The state of the execution state machine must be known to the Application Manager.
- Sponstneous state transitions can occur
- The Xlet must be able to report spontaneous transitions to the Application Manager

Application Manager Interactions



Application Life Cycle

Xlet Interface:

- Similar to applet model without inclusion of images, graphics, windows.
- The Application implements the Xlet interface.
- The Application Manager invokes the Xlet interface.
- The invocations cause state transitions

Application Life Cycle

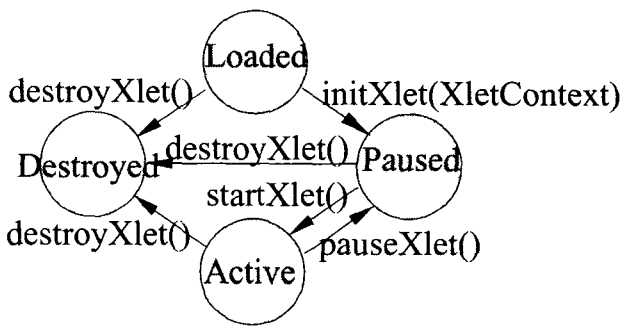
Four Application Execution States:

- Loaded
  - Code is loaded, initialized
- Paused
  - Application quiescent, minimal resource requirement
- Active
  - Application executes, ie can progess
- Destroyed
  - Application has released resources, terminated

Application Life Cycle

```
Package javax.tv.xlet;
public interface Xlet {
  void initXlet(XletContext);
  void pauseXlet();
  void startXlet();
  void destroyXlet(...);
}
```

Application Life Cycle



Application Life Cycle

XletContext

- The Application Manager provides the XletContext through the initXlet method.
- The properties interface of XletContext allows the Xlet to extract environment objects
- The state transition methods let the Xlet report spontaneous transitions to the application manager

Application Life Cycle

```
package javax.tv.xlet;
public interface XletContext {
  Object getXletProperty(String);
  void notifyPaused();
  void resumeRequest();
  void notifyDestroyed();
}
```

Service Information Concepts

Service Information

Service Information

- The Service Information:
  - is metadata about the available Services
  - is found in the broadcast as (often domain specific) protocol
- The Service:
  - Is a collection of content (executables, resources, audio, video, data)
  - which can be available on multiple transports
  - and is often referred to as a "channel"

Service Information

- Can be thought of as a database that the broadcast populates
- The application obtains the metadata through standard (not domain, not transport specific) interfaces
- The protocol specific format is not visible to the application

Service Information

Features

- Basic mechanism to find Services
- Protocol and Transport Independent
- Cache verus NonCashe is Configurable
- Synchronous and Asynchronous Access
- Extensible to other metadata

Service Information

There are three "views" of Service Information:

- Navigation Package
  - Traverse through hierarchical service metadata
- Guide Package (Program Guide Support)
  - Filters, schedules, events
- Transport Package
  - Exposes transport mechanism

Service Information

Asynchronous Service Information Access

- Mechanism permits applications to queue requests and continue execution
- Access methods are prefixed with 'retrieve' to reinforce semantics:
  - `retrieveProgramEvent(...)`

Service Information

Asynchronous Service Information Access

- The application implements SIRequestor to receive the metadata
- The platform forwards the metadata when it becomes available
  - `void notifySuccess(SIRetreivable[])`
  - `void notifyFailure(...)`

Service Information

Asynchronous Service Information Access

- The generic SIRetrievable is subclassed for specific metadata
  - Bouquet
  - Network
  - ProgramEvent
  - ServiceDetails

Service Information

- The platform retuns a SIRequest object to the application so as to scoreboard specific requests:
  - `SIRequest retrieveProgramEvent(Locator, SIRequestor);`
- This allows the platform to later cancel unrealizable requests.
  - `boolean cancel();`

Service Information

Asynchronous Service Information Access: Summary

- Clients implement SIRequestor to receive asynchronous data
- Platform returns the data as SIRetrievable instance
- Platform provides SIRequest objects through which it can cancel the request

Service Information

Service Information Manager

- Provides access to service information database
- Supports ServiceFilters to isolate services subject to certain criteria
- Generates events for changes to service information

Service Information

```
Package javax.tv.service.navigation;
public class SIManager {
  ServiceCollection
  createServiceCollection(ServiceFilter);
  Service getService(Locator);
  SIRequest retrieveSIElement(Locator,
  SIRequestor);
  Transport[] getTransports();
}
```

Service Interface

Service:

- Is a source of content
- Selectable through service selection operations
- Synchronous access to (cashed) basic metadata (name,locator)
- Asynchronous access to (noncashed) metadata details

Service Details Interface

ServiceDetails

- Metadata about the Service
  - Is an instance of a Service in the broadcast
  - Describes the schedule, transport
  - Traverses to ServiceComponents (executables, resources, audio, video, data)
  - Is extensible for new metadata

Service Selection

Service Selection Package

Service Selection

Features

- Abstracts the demodulation (tune) portion of the pipeline
- Asynchronous semantics
- Conditional access results exposed
- Support for multiple selection "contexts"

Service Selection

Inportant Service Selection Classes

- ServiceContext
  - Object to select a Service
  - Often relates to a specific, finite hardware resource; platform must support at least one
- ServiceContentHandler
  - Object to present a Service
  - Often extends javax.media for broadcast specific semantics

Service Selection

ServiceContext

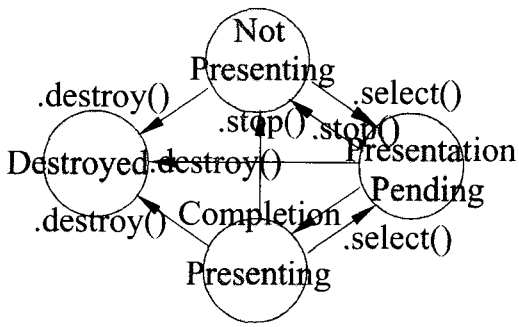
- Provides the formalism for multiple service execution
- Provides access to service specific content "handlers"
- Signals current state via events for completion, redirection, failure

Service Context States

Service Context State Machine

- NotPresenting: before selection or after completion
- PresentationPending: after selection and before completion
- Presenting: NormalContentEvent or AlternativeContentEvent
- Destroyed: ServiceContextDestroyedEvent

Service Context States



Broadcast Data

Broadcast Data Package

Broadcast Data

CarouselFile

- CarouselFile access to broadcast files
- Push Semantics for streams
- DatagramSocket for broadcast internet protocol

Broadcast Carousels

CarouselFile

- Access to data in broadcast carousels
- Extends File: Scales to flat or tree structures
- Abstracts domain specific protocols
  - Data Carousel (ATSC)
  - Object Carousel (DVB)
  - UHTTP (ATVEF)

Broadcast Carousels

CarouselFile

- Exploits familiar classes of java.io: File InputStream,RandomAccessFile,FileReader
- Carousel files map to local file name space
- Supports broadcast semantics: Creation of CarouselFile alerts the platform to prefetch and cache the data
- Provides events for content changes

Broadcast Data Streams

PushSourceStream

- Is a data stream source
- Client acquires the stream through javax.media.Manager
- Supports rate control, rather than flow control semantics: tells client when data arrives
- Subinterface raises exceptions for data loss

Broadcast Datagrams

Internet Protocol Encapsulation

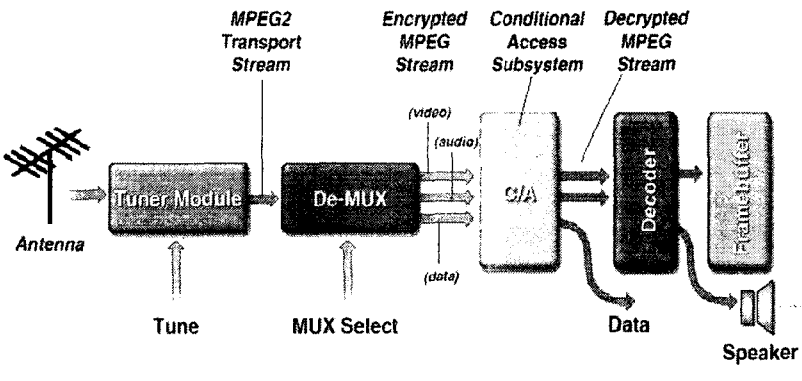
- Access to broadcast internet protocol
- Supports unicast and multicast through familiar DatagramSocket, MulticastSocket
- Translates Locator to broadcast internet protocol into private local internet protocol address

Media Control

Media Control Package

Media Control

- The Java Media Framework abstracts the presentation pipeline
- Player abstracts decode, presentation
- DataSource abstracts demodulation



Media Control

- Player
  - Presents content streams
  - Supports one or more media formats
  - Often abstracts hardware pipeline
  - Manages presentation state machine and presentation events

Media Control

- Controller
  - Subinterface of Player
  - Controls presentation state machine
  - Reports presentation state transitions

Media Control

- DataSource
  - Abstracts the media stream source
  - Media Format alerts platform as to which decoder to attach
  - Location of media stream is an opaque reference

Java Media Framework for Broadcast

- The implementation builds the pipeline as a side effect of service creation
- The client can elect to obtain the Players for fine grain control
  - ServiceContext.getServiceContentHandlers();
- Some standards extend Java Media Framework for their own controls

Graphics and Windows

Graphics and Windows Package

Graphics and Windows Extensions

- AlphaColor: Extends opaque color space to include alpha component.
- TVContainer: Provides the root window container
- Timer: Calls client after a certain time has elapsed

Additional Information

- http://java.sun.com/products
- http://java.sun.com/products/javatv
- http://java.sun.com/products/java-media/jmf/index.html