Java TV™ API Reference Implementation

Porting Guide

Version 1.0
November 15, 2000

Please send any comments on the *Java TV API Reference Implementation Porting Guide* to javatv-comments@sun.com.

Contents

Introduction

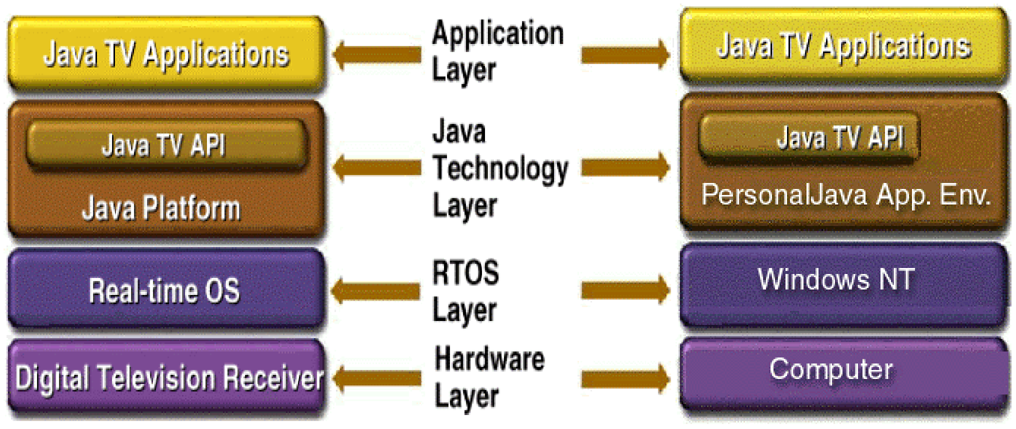
This guide documents the porting layer and the emulation layer included with the Java TV™ API reference implementation, version 1.0. The reference implementation is based on version 1.0 of the Java TV API. Included in this guide are descriptions of the classes and interfaces in the porting layer of the Java TV API reference implementation (RI) and the connection protocol between the porting layer and the platform-specific emulation layer.

The Java TV API defines a set of classes and interfaces to provide television-centric programs and services to set-top boxes. The features of the Java TV API support enhanced, interactive television and include access to the service information database, service selection, playback control and broadcast data. See the *Java TV API Technical Overview: The Java TV API Whitepaper* and the Java TV API javadocs at http://java.sun.com/products/javatv for more information about the Java TV API.

The Java TV API RI has been designed to be hardware platform agnostic and is written in the Java™ programming language, except for the MPEG-1 player provided as a part of the Java™ Media Framework implementation. This allows any customers with access to a Windows equipped PC to see the implementation's code working. It also makes the porting process simpler.

The following figure compares the architecture of the Java TV API running on a digital television receiver (set-top box) to the architecture of the RI running on a Windows NT computer.

Set-top Box Architecture vs. RI Architecture



The Java TV API in the RI consists of the public code, the porting layer and the emulation layer. The porting layer consists of code that is common to all implementations; it is both portable and platform independent. For information on the classes in the porting layer, see the section "Porting Layer". The emulation layer consists of code that is dependent on and specific to the supporting platform. For information on the classes in the emulation layer, see the section "Emulation Layer".

In the RI, all classes in the `com.sun.tv.*` packages are considered part of the porting layer, except for the receiver package which is in the emulation layer. The implementations of the `javax.tv.*` packages use the classes in `com.sun.tv.*` (as well as the classes in the PersonalJava™ application environment v3.1 and JMF). If you re-implement the classes in the porting layer, the `javax.tv.*` classes should function properly. For information on the RI's implementation of JMF, see the section "JMF Implementation".

Porting Layer

The porting layer of the RI provides portable classes that can be re-used for all implementations of the Java TV API. In the RI, the porting layer is located in the following directories:

- `com/sun/tv`
- `com/sun/tv/timer`
- `com/sun/tv/util`
- `com/sun/tv/si`
- `com/sun/tv/media`
- `com/sun/media/protocol`
- `com/sun/tv/net`
- `com/sun/tv/net/util`
- `java/net`

RI Directory com/sun/tv

The RI classes in the package `com.sun.tv` fall into three functional areas: application manager, locator, and service selection. The classes are listed below by functional area.

Application Manager Implementation Classes

The application manager is based on the Xlet application model. The classes implement the functionality for the Xlet state machine, send notification about state changes, load Xlet classes from carousel files, and communicate between Xlet and XletManager. See the *Java TV API Technical Overview: The Java TV API Whitepaper* at http://java.sun.com/products/javatv for more information about the Xlet application model.

AppSignalEvent
The ApplSignalEvent class is used by the service selection classes to notify the XletManager that the current service includes an Xlet to be signaled for execution.

AppSignalEventListener
AppSignalEventListener is an event listener interface implemented by classes to receive notification of AppSignalEvent objects.

AppSignalEventFactory
The AppSignalEventFactory class can be used by objects to add themselves as a listener to events that signal applications.

DisplayManager
The DisplayManager class controls access to the screen. Currently, this class assumes that the Xlets running on the platform are displaying themselves in a root container (an AWT Frame) that was created by DisplayManager. The DisplayManager class is used primarily by the XletManager.

Holder
Holder objects are used to hold Request objects, which change the state of Xlets.

Result
The Result class represents the result from performing an Xlet lifecycle action. The Result class is constructed after the action and put in a Holder for the XletManager thread to pick up. The result is either successful or unsuccessful. An XletStateChangeException is thrown for the unsuccessful result.

Request
Request objects are used to hold lifecycle change requests that are sent to Xlets.

XletContainer
The XletContainer class is an AWT container that can be used by an Xlet. The DisplayManager creates instances of the XletContainer class when it displays Xlets.

XletContextImpl
The XletContextImpl class implements `javax.tv.xlet.XletContext`, which is used by Xlet to communicate with an XletManager and access Xlet properties.

XletLoader
The XletLoader class is a `java.lang.Classloader` and loads an Xlet class from a carousel.

XletManager
XletManager is the primary class that manages the execution of Xlets within the Java TV API Reference Implementation. The XletManager class handles all aspects of Xlet management, from receiving and interpreting Xlet signaling, to loading an Xlet, and managing its state.

XletProxy

The XletProxy class provides a context for an Xlet running in an XletManager. In particular, it provides the XletManager with access to all the objects needed to support the normal operation of an Xlet.

XletRunnable
The XletRunnable class represents a class that implements the `java.lang.Runnable` interface. This implementation of the Runnable interface is used when new threads are created to call the lifecycle methods on an Xlet. Note that those methods are called on Xlets asynchronously to help ensure the integrity of XletManager.

XletState
The XletState class is used for tracking an Xlet object's current state.

Locator Implementation Classes

The implementation of the Locator-related classes, LocatorImpl and LocatorFactoryImpl, are in the `com/sun/tv` directory. These classes implement the locator functionality in the RI. The RI is protocol agnostic but still must define its own locator syntax. The syntax used by the RI is similar to that used for URLs.

In general, an implementation will define the supported protocols and syntax. Usually, these will be defined by a particular standards body.

PROTOCOL	**SYNTAX FORMAT**
Service Protocol	service:/<servicename>
Service Description Protocol	description:/<servicename>
Service Component Protocol	component:/<componentname>service:/<servicename>
Service Component Protocol	component:/<xletname>service:/<servicename>
Program Event Protocol	event:/<eventname>service:/<servicename>
Program Event Description Protocol	eventdescription:/<eventname>
Transport Stream Protocol	transport:/<transportstreamname>
Network Protocol	network:/<networkname>
Bouquet Protocol	bouquet:/<bouquetname>
Carousel Protocol	carousel:/<mountpoint_path><file_Path>
All Networks	network:/*
All Bouquets	bouquet:/*
All Transport Streams	transport:/*

The RI includes the following restriction:

- Only one Transport is supported.

Because of this restriction:

- Only one BouquetCollection is available
- Only one NetworkCollection is available

- Only one TransportStreamCollection is available.

LocatorFactoryImpl
The LocatorFactoryImpl class implements the class `javax.tv.locator.LocatorFactory`, which defines a factory object that creates instances of the locator interface.

LocatorImpl
The LocatorImpl class implements the interface `javax.tv.locator.Locator`. LocatorImpl represents a locator, which provides an opaque reference to the location information of an object addressable from the Java TV API. In addition to implementing the required methods of Locator, LocatorImpl includes a number of utility functions for locator comparison and identification.

Service Selection Implementation Classes

The service selection implementation classes implement the `javax.tv.service.selection` package. This package provides the functionality that allows applications to select services. The implementation of this package has a strong dependence on the Java Media Framework. The actual presentation of a service (the playing of an MPEG-1 video file) is accomplished with a JMF player. When a new service is selected, the service selection implementation must stop the current JMF player from playing and start another. Much of this functionality may be handled in hardware in other implementations.

Handler
The Handler class implements a JMF player for handling content specified in a locator. A Handler object parses the locator string and delegates the handling of the content to a JMF player registered for the content.

ServiceContextFactoryImpl
The ServiceContextFactoryImpl class implements the class `javax.tv.service.selection.ServiceContextFactory` and serves as a factory for the creation of ServiceContext objects.

ServiceContextImpl
The ServiceContextImpl class implements the interface `javax.tv.service.selection.ServiceContext` and the state machine defined by it. The class also initiates the playing of media content when a particular service is selected in the context.

RI Directory com/sun/tv/timer

The package `com.sun.tv.timer` consists of one class, TVTimerImpl. The `javax.tv.util.TVTimer` class adheres to the semantics in the `com.sun.ptimer.PTimer` class included with the PersonalJava platform. The TVTimerImpl in the RI delegates to the PTimer class included in the PersonalJava platform because the implementation of the TVTimer class would almost certainly depend on native code and the Java TV API RI contains a minimum of native code.

Timer Implementation Class

The timer-related class TVTimerImpl is found in the `com.sun.tv.timer` package.The remainder of the timer implementation is in the `javax.tv.util` package.

TVTimerImpl
The TVTimerImpl class (in the `com.sun.tv.timer` package) implements the TVTimer class. TVTimerImpl includes two hash tables that are used to map TVTimerSpec classes to PTimerSpec classes. One hash table maps a TVTimerSpec object to a PTimerSpec object, and the other hash table maps a PTimerSpec object to a TVTimerSpec object. When a TVTimerSpec class is added to TVTimerImpl, an instance of PTimerSpec is created, added to the hash tables and registered with a PTimer. When that PTimer “goes off,” the TVTimerImpl class is notified. It then maps the PTimerSpec back to a TVTimerSpec and notifies the appropriate listeners.

RI Directory com/sun/tv/util

The classes found in this directory are utility classes used in the functioning of the RI. Specifically, the classes in this directory allow different types of objects to be sorted.

CompareInterface
The CompareInterface interface consists of one method that accepts two objects as arguments. The class implementing this interface returns the result of the compare as an integer.

QuickSort
The QuickSort class implements the QuickSort algorithm. The QuickSort class sorts objects that implement CompareInterface. This sorting code is used in a number of places within the RI.

RI Directory com/sun/tv/si

The service and service information classes handle service information stored in the SI database and represent the layout and content of audio/video/data streams. The directory `com/sun/tv/si` corresponds to the package `com.sun.tv.si`. Its classes are listed below in alphabetical order.

Service Information Implementation Classes

These classes implement the service information packages in the Java TV API (`javax.tv.service`, `javax.tv.service.guide`, `javax.tv.service.navigation`, and `javax.tv.service.transport`). The classes in these packages provide applications with information on the services that are available to an individual receiver. The RI's service information classes are populated by the SIEmulator class (see `com/sun/tv/receiver`). The RI contains examples that allow the SIEmulator class to be populated either programmatically or via XML files.

BouquetImpl
The BouquetImpl class implements the interface `javax.tv.service.transport.Bouquet`, which represents information about a bouquet (a collection of services that can span transport stream and network boundaries). Note that bouquets might not be supported by all protocols.

CacheManager
The CacheManager class manages the cache used to hold service information within the RI. The cache is implemented by subclassing `java.util.Hashtable` and is keyed on the external form (string representation) of locators. CacheManager can be found in the RI directory `com/sun/tv`.

ContentRatingAdvisoryImpl
The ContentRatingAdvisoryImpl class implements the interface `javax.tv.service.guide.ContentRatingAdvisory`, which is used to indicate, for a given program event, ratings for any or all of the rating dimensions defined in the Content Rating System for the local rating region.

FavoriteServicesNameImpl
The FavoriteServicesNameImpl class implements the FavoriteServicesName interface, which is used to create a collection of services based on a user preference for favorite services.

NetworkImpl
The NetworkImpl class implements the Network interface, which provides descriptive information about a network of transport streams.

ProgramEventDescriptionImpl
The ProgramEventDescriptionImpl class implements the ProgramEventDescription, which provides a textual description of a ProgramEvent.

ProgramEventImpl
The ProgramEventImpl class implements the ProgramEvent class, which provides structured information about a program event.

ProgramScheduleImpl
The ProgramScheduleImpl class implements the ProgramSchedule interface, which provides information about the current, next and future events.

RatingDimensionImpl

The RatingDimensionImpl class implements the RatingDimension interface, which provides information about supported multiple-rating dimensions.

ReceiverListener
The ReceiverListener class extends EventListener to listen to SIChangeEvent. SIManagerImpl uses the ReceiverListener class to receive notification of an SIChangeEvent generated in the emulation layer. The ReceiverListener class can be found in the RI directory `com/sun/tv`.

ServiceComponentImpl
The ServiceComponentImpl class implements the ServiceComponent interface, which provides information about individual components of the service.

ServiceDescriptionImpl
The ServiceDescriptionImpl class implements the ServiceDescription interface, which provides a textual description of a service.

ServiceDetailsImpl
The ServiceDetailsImpl class implements the ServiceDetails interface, which provides detailed information about a service bound to a transport stream.

ServiceImpl
The ServiceImpl class implements the Service interface, which provides structured information about a service.

ServiceIteratorImpl
The ServiceIteratorImpl class implements the ServiceIterator interface. These objects are returned by ServiceList objects to allow applications to interate through the Service objects contained in a service list.

ServiceListImpl
The ServiceListImpl class implements the `javax.tv.service.navigation.ServiceList` class. Services are stored in a Vector and sorts are performed by the QuickSort class in the `com/sun/tv/util` directory.

SIChangeEventImpl
The SIChangeEventImpl class implements the SIChangeEvent class. These objects are created and sent to SIChangeListener objects to signal changes detected in the SI database.

SIManagerImpl
The SIManagerImpl class implements the SIManager class. The SIManageImpl class represents the implementation of the central managing entity that has knowledge of the entire network or a collection of networks. The SIManageImpl class can create a collection of services based on the ServiceGroup filtering rules. This class must handle a

number of service information requests. The reference implementation caches these requests and services them asynchronously as the specification requires.

SIRequestImpl
The SIRequestImpl class implements the SIRequest interface. SIRequestImpl registers itself at the time of the asynchronous call for a single request and is automatically unregistered when the request is completed.

TransportImpl
The TransportImpl class implements the Transport interface. TransportImpl is used for notifications about SIChange events and may expose various types of entities (e.g., bouquets, networks and/or TransportStreams).

TransportStreamImpl
The TransportStreamImpl class implements the TransportStream interface and provides information about a transport stream.

RI Directory com/sun/tv/media

The classes in this directory implement the APIs included in the `javax.tv.media` package.

TV Media Package Implementation Classes

The `javax.tv.media` package provides controls and events for the management of real-time media in a television environment.

AWTVideoSizeControlImpl
The AWTVideoSizeControl class implements the AWTVideoSizeControl class from the `javax.tv.media` package. The AWTVideoSizeControlImpl class is used by applications to manipulate the size of video windows. Because of limitations on the PC platform, this RI does not allow resizing of the video output, which is allowable by the specification.

MediaSelectControlImpl
The MediaSelectControlImpl class implements the MediaSelectControl class from the `javax.tv.media` package. MediaSelectControl objects are generally acquired by applications via their ServiceContext to allow them to perform more fine-grained selection operations. For example, MediaSelectControl objects allow audio and video components to be selectively added to or removed from a display.

RI Directory com/sun/tv/media/protocol

This directory contains the classes that implement the classes in the `javax.tv.media.protocol` package.

DataSource

The DataSource class implements the DataSource interface. The `com/sun/tv/media/protocol` directory contains two versions of the DataSource class, one in its Component sub-directory, and the other in its Service sub-directory. The former is used to implement the DataSource that is associated with ServiceComponent objects, and the latter is used to implement Service objects.

PushSourceStream2Impl
The PushSourceStream2Impl class implements the PushSourceStream2 interface. The `com/sun/tv/media/protocol` directory contains two versions of the PushSourceStream2Impl class, one in its Component sub-directory, and the other in its Service sub-directory. The former is used to implement the PushSourceStream2 that is associated with ServiceComponent objects, and the latter is used to implement Service objects.

RI Directory com/sun/tv/net

The classes in the `com.sun.tv.net` package implement the functions in the `javax.tv.net` package and provide access to IP datagrams transmitted in the broadcast stream. Together with the `java/net` socket implementation, these classes allow the user to obtain a locally-obtained IP address assigned to a ServiceComponent locator, and then access an IP datagram associated with that ServiceComponent.

Note that the RI expects the user to provide a file that contains data in the IP datagram format. Those who wish to port this package on a set-top box should modify a few files in this package, especially EncapIPStream, because this is where the data is obtained by opening a local file rather than from IP encapsulated data in an MPEG-2 transport stream.

EncapDatagramSocketImpl
The EncapDatagramSocketImpl class implements an Encapsulated IP Datagram Socket. The RI does not implement any security checks, but it does support SOCKS, Version 4. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

EncapIP
The EncapIP class implements a parser for Encapsulated IP datagrams in an MPEG-2 transport stream. The EncapIP class is used in the implementation of `javax.tv.net.InterfaceMap`.

EncapIPDataSource
The EncapIPDataSource class implements a DataSource object for Encapsulated IP. In the RI, this class reads from an instance of the EncapIPStream class, which reads from a file. The EncapIPDataSource class is used in the implementation of `javax.tv.net.InterfaceMap`.

EncapIPStream

The EncapIPStream class represents an Encapsulated IP Stream. It implements the PushSourceStream2 interface and reads from a file. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

GenericPacket
The GenericPacket class is a subclass of the Packet class and implements a general-purpose, network packet buffer mechanism. The GenericPacket class has been modified slightly for the RI. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

InterfaceMapImpl
The InterfaceMapImpl class implements the encapsulated IP functionality in the RI (`javax.tv.net.InterfaceMap`).

IPReass
The IPReass class reassembles IP packets. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

Packet
The Packet class is an abstract class representing a network packet. Subclasses must define the general behavior of the packet they wish to emulate. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

PacketDiscardedException
The PacketDiscardedException exception is thrown when an IP packet passed up by a lower layer protocol is discarded. This class is used in the implementation of `javax.tv.net.InterfaceMap`.

RI Directory com/sun/tv/net/util

These classes are utility classes used in the implementation of `javax.tv.net.InterfaceMap`.

SystemThread
The SystemThread class is a subclass of the `java.lang.Thread` class, which allows non-standard priority levels to be set. The SystemThread class also has a specifiable stack size. This class is used in the implementation of the Timer class.

Timer
The Timer class provides a general-purpose event timer. It is used by subclassing it and overriding the callback method. This timer is not as robust as the timer included in `javax.tv.util`.

RI Directory java/net

The `javax.tv.net` package defines an API that allows applications to access IP datagrams that are transmitted in the broadcast stream. These classes overwrite the socket implementation in the PersonalJava platform. The `javax.tv.net.InterfaceMap` class implements this functionality. This class depends on `java.net.DatagramSocket` and `java.net.MulticastSocket`, which are found in the PersonalJava platform. In order to simulate the correct behavior of these classes in the RI, alternate implementations of these classes are provided. Other implementations may choose to modify the actual implementation of the `java.net` classes.

DatagramSocket
In addition to providing the functionality of the `java.net.DatagramSocket` class from the PersonalJava application environment, this class also represents a socket for receiving datagram sockets with the InetAddress reported at `InterfaceMap.getLocalAddress()`. Note that the port number is ignored when the socket is used for receiving IP datagram for InterfaceMap.

MulticastSocket
In addition to providing the functionality of the `java.net.MulticastSocket` class from the PersonalJava application environment, this class also represents a socket for receiving datagram sockets with the InetAddress reported at `InterfaceMap.getLocalAddress()`. Note that the port number is ignored when the socket is used for receiving IP datagram for InterfaceMap.

Emulation Layer

This section describes the RI emulation layer and its connection to the porting layer. The emulation layer consists of classes that are platform specific and cannot be re-used on all platforms. The functionality in the emulation layer includes storing and generating SI events, playing media content using JMF, and mechanisms for selecting. In many implementations a significant amount of the emulation layer will be re-implemented in hardware. This RI targets the PC platform so certain hardware is not generally available (such as MPEG section tables and hardware MPEG-2 players).

The emulation layer consists of the directory com/sun/tv/receiver. The following section describes the implementation of the platform-specific mechanisms in the RI. The classes described here could be implemented in native code on a digital television receiver, or, in some cases, in the hardware.

RI Directory com/sun/tv/receiver

This directory corresponds to the package `com.sun.tv.receiver`. The RI includes classes to emulate SI storage, generate SIChange events, and then to propagate those events to the porting layer. The primary class used for this is the SIEmulator class. The SIEmulator class is used to populate the SI database in the RI.

Two examples of classes that populate the SI database via SIEmulator have been included:

- **`ReceiverFile.java`**: The `ReceiverFile.java` file parses an XML file that describes SI. An example XML file and sample DTD can be found in the `lib` directory. The file `ReceiverFile.java` can be found in the `com/sun/tv/receiver` directory.
- **`SampleData_01.java`**: The `SampleData_01.java` file represents programmatic population of the SI database. SampleDate_01 can be found in the `samples/db` directory.

Considerable control over the contents of the SI database is via programmatic population (such as by using `SampleData_01.java`). This class or classes with similar functionality call private methods in the SIEmulator class. You could modify the `SampleData01` class to read the SI from another source (such as an MPEG-2 transport stream or a network socket) and populate the database from that source.

ReceiverFile reads an XML file that contains a description of the SI database. The XML example is essentially a programmatic example, except that population is triggered via XML files. ReceiverFile parses the XML file specified in the `JavaTV.properties` file. This file is parsed once every 10 seconds. This allows the RI to simulate a dynamically changing SI database as would be found on an actual network. Most functions of the service information interface in the Java TV API can be simulated in this manner. For more information on using XML with the RI and the details of the XML syntax, see the `docs` directory.

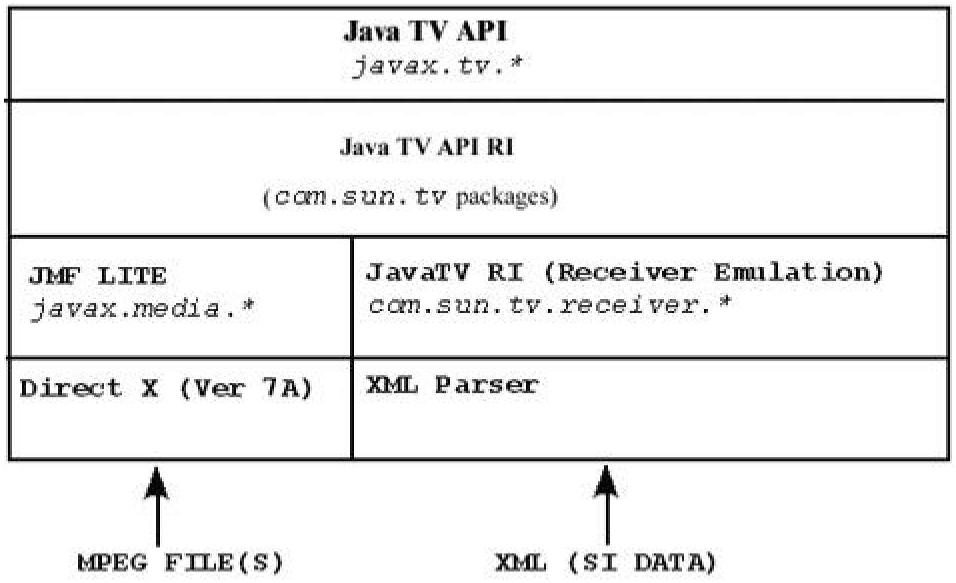
The RI looks for the `JavaTV.properties` file when the API is initially accessed. This file defines the property:

```
ServiceFileHandler=SampleData_01
```

ServiceFileHandler refers to a class file that implements SampleDataInterface. The RI creates an instance of the class specified and calls methods on this interface that will populate the SI database.

The following figure illustrates the relationship between the porting layer and emulation layers in the Java TV API reference implementation. The Java TV API is built on the Java TV API porting layer (the `com.sun.tv` packages). These packages represent a platform-independent collection of classes that provides the majority of the RI's functionality. These classes in turn rely on functionality usually provided by software or hardware included in the receiver (hardware MPEG players and MPEG section tables).

These features are unavailable to the RI, so they are included in the emulation layer. The emulation layer consists of a JMF implementation (JMF Lite) and emulation of the service information tables (the classes in `com.sun.tv.receiver`). Video information is provided by MPEG-1 movies and service information is usually provided in the form of XML files.



To notify the SIManager about the SI events, the emulator calls `notifyChange()` in the SIManagerImpl class. SIEmulator uses two alternative sources for passing emulated events to the porting layer. One source of events is an XML file that contains service information. ReceiverFile is used to read the information from the XML source and pass it to the emulator.

Another source of the events is the SampleData class. The emulation data are hardcoded in the class. The SampleData class can be replaced with other classes that implement the SampleDataInterface interface. The name of the source class is defined in the Settings class.

All other implementations must call the same method in the SIManagerImpl class to notify the listeners about the SIEvent changes.

The following classes are included in the emulation layer for SI events:

ReceiverFile
The ReceiverFile class implements SampleDataInterface and is used as a data source for SIEmulator that reads data from an XML file. ReceiverFile is an internal class for the emulation layer and is not used for the connection to the porting layer.

SampleDataInterface
The SampleDataInterface interface is implemented by emulated data sources to pass the SIChange events to the emulator. SampleDataInterface is an internal interface for the emulation layer and is not used for the connection to the porting layer.

Settings
The Settings class defines settings for the emulation. Upon bootup, the class looks for a `JavaTV.properties` file in the `lib` directory to read in the user-defined settings. If the file is not present or the necessary variables are missing from the file, the Settings class uses its own default values for the emulation. In particular, the class specifies a SampleData_01 file as a data source for SIEmulator to use hard coded emulation data.

SIEmulator
The SIEmulator class emulates a source for SIChange events. SIEmulator can emulate source events, reading them either from an XML source or hard-coded in the SampleData_01 file. The emulator should be replaced on a real source of events.

JMF Implementation

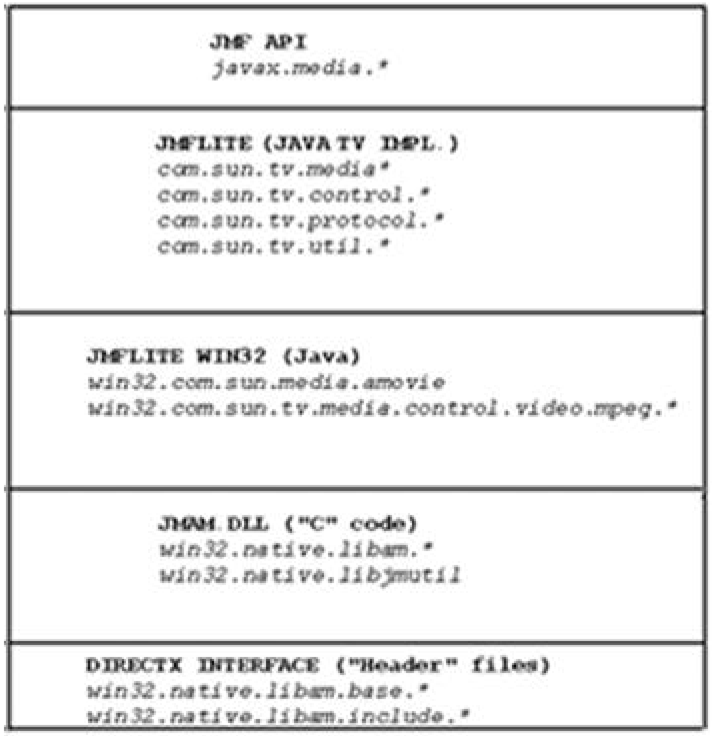
The Java Media Framework is used by the Java TV API to play the content of the selected media. For more information on JMF see http://java.sun.com/products/java-media/jmf.

A platform-independent version of the framework portion of JMF has been included in the RI. This portion of JMF matches content with players and manages those players.

Also included is an MPEG-1 player that is written to the DirectShow libraries included in Windows NT. To compile the player code, you must have access to the DirectShow developer's kit, version 7.A (available from www.microsoft.com) and Visual C, version 6.

The following figure shows the JMF architecture in the RI. The JMF implementation code in the RI can be found in the `jmflite/src` directory. The three main JMF packages in the RI are:

- `javax.media.*` -- JMF 1.0 API (public API)
- `com.sun.tv.media.*` -- The RI's implementation of JMF in the porting layer
- `win32.*` -- Windows NT-specific implementation of JMF



RI Directory com/sun/tv/media

The `com.sun.tv.media` package contains base classes for JMF operation, including the implementation of a Player, Controller, Clock, and TimeBase.

ControllerAdapter
The ControllerAdapter class is an event adapter that receives JMF `javax.media.ControllerEvent` and dispatches them to an appropriate stub method.

DataLostErrorEvent
The DataLostErrorEvent class is posted when a Controller has lost data.

GainControlAdapter
The class GainControlAdapter implements `javax.media.GainControl`. The GainControlAdapter class provides methods for:

- manipulating the amplitude of the audio signal and performing math required to map linear gain specifiers to decibels. (The relationship between a linear gain multiplier and the gain specified in decibels is: value = pow(10.0, gainDB/20.0)).
- registering, unregistering and informing registered listeners about changes in gain value of the audio signal by posting a GainChangeEvent.
- manipulating the muted state of the audio signal.

MediaClock

The MediaClock class implements the math and maintains the correct states to perform the computations from media time to time-base time.

MediaController
The MediaController class implements the basic functionality of `javax.media.Controller`, including:

- clock calculations using the MediaClock helper class.
- RealizeWorkThread and PrefetchWorkThread to implement `realize()` and `prefetch()` in the correct unblocking manner.
- ListenerList to maintain the list of ControllerListener.
- two ThreadedEventQueues for incoming and outgoing ControllerEvents.

MediaPlayer
The MediaPlayer class handles all event handling and management of any Controller under its control.

MediaTimeBase
The MediaTimeBase class is the abstract base class to create a TimeBase object out of the media time of a component. Because TimeBase ticks even when the media has stopped, the MediaTimeBase class internally maintains a system time base that takes over when the media has ended. MediaTimeBase is extended by the class `win32.com.sun.media.amovie.AMController`.

SeekFailedEvent
The class SeekFailedEvent is used to indicate that the Controller could not start at the current media time (set using setMediaTime).

SystemTimeBase
The class SystemTimeBase is the implementation of the default JMF `javax.media.TimeBase` interface. In the RI's JMF implementation, this class is used by MediaClock, MediaPlayer, and MediaTimeBase in the `com.sun.tv.media` package.

RI Directory com/sun/tv/media/controls

The package `com.sun.tv.media.controls` contains classes and interfaces to extend the basic JMF `javax.media.Controller` functionality for the RI. The directory also includes classes necessary to monitor the changes in the controller state.

AtomicControl
The AtomicControl interface specifies functionality common to all JMF controls.

AtomicControlAdapter
The AtomicControlAdapter class implements functionality common to all JMF controls.

ControlChangeEvent
The ControlChangeEvent event contains information about which control has changed.

ControlChangeListener
The ControlChangeListener interface specifies a listener for changes in the state of a control.

NumericControl
The NumericControl interface specifies the functionality of a control that represents the state by a numeric value.

NumericControlAdapter
The NumericControlAdapter class implements the functionality of a control that represents the state by a numeric value.

RI Directory com/sun/tv/media/protocol

The class in this directory provides an implementation of data transfer protocols.

InputSourceStream
The InputSourceStream class implements the JMF class `javax.media.protocol.PullSourceStream` to build a source stream out of an input stream.

RI Directory com/sun/tv/media/protocol/file

This directory provides a pull data source implementation for a file protocol.

DataSource
The DataSource class implements the JMF class `javax.media.protocol.PullDataSource` to provide a pull data source for a file protocol. Note that the RI provides two additional DataSource classes in the porting layer to customize a DataSource for `javax.tv.media.protocol.PushDataSource2`.

RI Directory com/sun/tv/media/util

This directory contains utility classes, including threads, security, and settings classes.

JMFI18N
The JMFI18N class specifies the resource bundle, i.e., the class that contains locale-specific objects.

JMFProperties
The JMFProperties class is used to access and modify information about JMF settings.

JMFSecurity
The JMFSecurity class is used to monitor security when calling thread, connection, file access, event queue and window and object methods.

LoopThread
The LoopThread class is a base class for a looping thread, which implements a safe way of pausing and restarting. Instead of using `suspend()` and `resume()` from the PersonalJava platform `java.lang.Thread` class, the LoopThread class provides a `pause()` and `restart()` method. This class is extended by the class `win32.com.sun.media.amovie.AMController`.

MediaThread
The MediaThread class implements a thread class that all JMF-created threads should be based on.

MediaThreadGroup
The MediaThreadGroup class is a base thread class from which all JMF-created threads should derived.

ThreadedEventQueue
The ThreadEventQueue class is a utility class to manage an event queue in a thread.

RI Directory com/sun/tv/media/util/locale

The JMF class in the package `com.sun.tv.media.util.locale` is JMFProps, which defines locale properties for JMF.

JMFProps
The class JMFProps contains hard-coded, locale-specific information used by `com.sun.tv.media.util.JMFI18N`.

Index