## Tutorials

These tutorials will hopefully give you an overview of some of the interactive TV standards that are being deployed today.

These tutorials assume that you either know something about digital broadcasting already, you've read my (or someone else's) introductory page, or that you're hoping things will get easier as they go along and that you don't really need to understand all these silly acronyms.

In the latter case, good luck - I've tried to keep these pages fairly acronym-free, but that's not always possible.

From this point on, I'll only be describing the highlights of the specification. After all, the MHP specification is over one thousand pages long, and any tutorial can only cover so much of that. The full specification is available for free [download](http://www.interactivetvweb.org/tutorials) from the [ETSI web site](http://pda.etsi.org/pda/home.asp?wki_id=12127.). Many other specifications relating to DVB services can also be downloaded from there.

The [JavaTV specification](http://java.sun.com/products/javatv/) is similarly available from the Sun web site Reading the specs themselves is the only way to grasp all of the subtleties of the specifications, and at the end of the day you'll need to do this anyway if you're going to be developing software using one of these standards.

If you're fairly familiar with the specifications, but want to learn how to use a specific feature, take a look at the [code samples](http://www.interactivetvweb.org/resources/code_samples). These commented examples will show you how to do some of the most common tasks that an MHP [application](http://www.interactivetvweb.org/tutorials) needs to do.

Please choose which [tutorial](http://www.interactivetvweb.org/tutorials) to view:

* [Digital TV Introduction](http://www.interactivetvweb.org/tutorials/dtv_intro)
* [Getting Started](http://www.interactivetvweb.org/tutorials/getting_started)
* [The MHP Tutorial](http://www.interactivetvweb.org/tutorials/mhp)
* [The OCAP Tutorial](http://www.interactivetvweb.org/tutorials/ocap)
* [The JavaTV Tutorial](http://www.interactivetvweb.org/tutorials/javatv)

## The JavaTV Tutorial

<http://www.interactivetvweb.org/tutorials>

<http://www.interactivetvweb.org/tutorials/javatv>

<http://www.interactivetvweb.org/tutorials/dtv_intro>

<http://www.interactivetvweb.org/tutorials/dtv_intro/broadcast_engineering_basics>

<http://www.interactivetvweb.org/tutorials/mhp>

<http://www.interactivetvweb.org/tutorials/ocap>

Sun have been working in digital TV standardisation for several years. Several open standards such as MHEG-6 and the DAVIC standard had used [Java](http://www.interactivetvweb.org/tutorials/javatv) in digital TV middleware, but none of these standards had used Java as the primary application model - Java was always treated as a way of adding more advanced scripting functionality to declarative technologies.

This had led to a complicated and clumsy [application](http://www.interactivetvweb.org/tutorials/javatv) model, and so Sun felt that it was time to build a pure Java solution. The result of this was the JavaTV specification. Since DVB was working towards the same end, there was a lot of cross-talk between Sun and DVB, and many companies were involved in both specifications. MHP uses the application model from JavaTV as a central piece of the MHP specification, and all of the JavaTV APIs are included in MHP.

Given this overlap, why have a separate JavaTV specification at all? Well, MHP is specific to the DVB family of standards. JavaTV, on the other hand, is not. JavaTV describes a set of digital TV concepts such as accessing service information, selecting a new service, and loading files from a carousel rather than a normal filesystem, but it does so in a way that's not tied to any digital TV standard. It's possible to implement JavaTV on an ATSC system, on a DVB system or on an ARIB system - in fact, this is now happening with the spread of open standards such as MHP, OCAP, ACAP and ARIB B23.

While JavaTV describes many of the core concepts that are needed, it is not a [complete](http://www.interactivetvweb.org/tutorials/javatv) specification of a digital TV platform. There are elements missing that must be defined before JavaTV can be used in the real world (e.g. JavaTV says nothing about what underlying Java platform is in use), and so JavaTV is used more as a component of other standards than as a middleware platform in its own right. For this reason, it may appear that learning JavaTV development is less useful than learning OCAP or MHP development.

Since it forms a common core for other standards, though, learning to develop JavaTV applications can be very valuable when it comes to improving portability: using the JavaTV APIs can help reduce the number of places where your application must be modified, and can help you to avoid making any platform-specific assumptions about your code.

This tutorial is designed for people who already have a knowledge of Java and of the basics of digital TV. If you're not sure that your digital TV knowledge is suitable for this tutorial, please take a few moments to review the [introduction to digital TV](http://www.interactivetvweb.org/tutorials/dtv_intro) elsewhere on this site.

**A Developer's Guide To Digital TV**

This [tutorial](http://www.interactivetvweb.org/tutorials/dtv_intro) is aimed at those of you who are moving to the interactive  
TV world, either from a background in PC software development, or as a  
manager who needs to know more about this funny [DTV](http://www.interactivetvweb.org/tutorials/dtv_intro) stuff.

This tutorial can be split into two main sections: the first section  
covers the basics of [digital TV](http://www.interactivetvweb.org/tutorials/dtv_intro) systems and how DTV signals are transmitted,  
while the second section takes a detailed look at the DSM-CC standard that  
is used by most digital TV systems for broadcasting data streams.

There's a lot of technical stuff in the next few pages, but we'll try  
and keep that to a minimum unless we explicitly have to. Some parts of  
this are more technical than others: only people who really need to know  
the gory details of DSM-CC should try to read past the introduction to  
DSM-CC, for instance.

At the moment, some sections are fairly specific to the DVB system of  
digital TV broadcasting. To stop anyone dealing with OCAP from feeling  
left out, however, we do have a tutorial on ATSC service information under  
development.

## A little More About DVB And MHP

The [Digital Video Broadcasting](http://www.dvb.org/) consortium is an industry consortium that standardizes various aspects of digital TV broadcasting. In the past, they've standardized issues such as how digital TV signals are transmitted over cable, satellite or terrestrial broadcasting networks, how information describing a digital TV transmission is encoded in the bitstream, and generally how to make the various parts of a digital TV system work together.

They've been doing this for several years now, and DVB standards are pretty widely used across Europe and Asia, with some take-up in the US. The DVB web site's page on [the use of DVB standards worldwide](http://www.dvb.org/about_dvb/dvb_worldwide/) gives you an idea of how many countries are now using DVB standards. A couple of years ago, they [started](http://www.interactivetvweb.org/tutorials/dtv_intro/dvb_background) to get the ball rolling on an open standard for interactive digital TV.

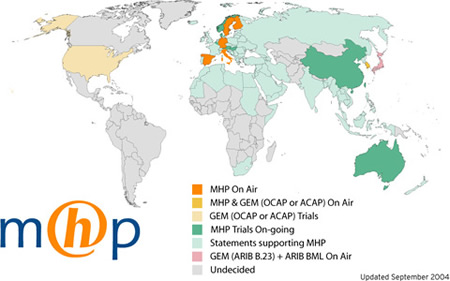
There have been attempts to do this before, by ISO with the [MHEG](http://www.km.giti.waseda.ac.jp/WG12/) standards, by [DAVIC](http://www.davic.org/), and by [ATSC](http://www.atsc.org) with the [DASE](http://www.itl.nist.gov/div895/cmr/dase/) standard. Unfortunately these were not very successful for a variety of reasons, but DVB's attempt seems to have been timed just about right. Previous standards had either been finalized just at the start of the [Java](http://www.interactivetvweb.org/tutorials/dtv_intro/dvb_background) wave, or as attempts to extend and adapt older standards to include Java. These timing issues meant that their chances of success were greatly reduced, but the world wasn't really ready for an open standard at that time.

These older open standards are still used in some cases (e.g. one of the UK digital broadcasters uses MHEG), but companies like OpenTV, NDS, and Microsoft (who have proprietary iTV solutions) still dominate the iTV market at the moment.

MHP is an attempt to change this. In it's simplest form, it's a set of Java APIs that let you write interoperable applications. These are broadcast as part of the MPEG-2 stream that makes up a digital TV signal, and an MHP-compliant receiver can run these applications on your TV. Since MHP uses Java (and some API extensions that are specific to digital TV), you can do almost anything you want to.

People have written MHP games, information services and electronic program guides, as well as news tickers, stock tickers and services related to TV shows such as sports broadcasts. I don't have the rights to show any of these applications here, but the [example applications page on the MHP site](http://www.mhp.org/content_creation/examp_app.html) shows a small selection of what's out there. There are lots of others that aren't shown there, but that's a pretty good selection. MHP is starting to become more widely adopted, at least by governments and public terrestrial broadcasters. It's still got a long way to go in the area of commercial broadcasting (e.g. cable and satellite operators), but it's making a few inroads. The [map](http://www.interactivetvweb.org/tutorials/dtv_intro/dvb_background) below (taken from the MHP web site) shows MHP adoption worldwide:

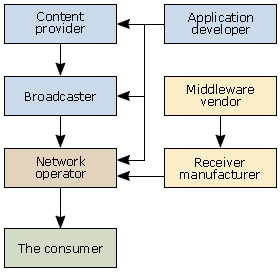
Adoption of MHP worldwide. Source: the [DVB web site](http://www.dvb.org/graphics/internal/WAM-DVB-MHP.png).

[](http://www.interactivetvweb.org/images/tutorials/dtv_intro/mhp_adoption_map.jpg)

OK, so you there's an open standard out there that's starting to compete with the existing proprietary solutions. So what? What does MHP have to offer over the current solutions?

Before we answer this, lets take a look at the content delivery chain for a digital TV signal. This particular diagram illustrates a situation that's fairly common at the moment:

A vertical (closed) market for digital television.



This a vertical market - the broadcasters rent the set-top boxes to the consumer as part of the subscription deal, which means that they control the specification for the receivers. So, all the receiver manufacturers can do is build boxes to the specification of the broadcaster, and all the consumer can do is take the box that the broadcaster provides to them.

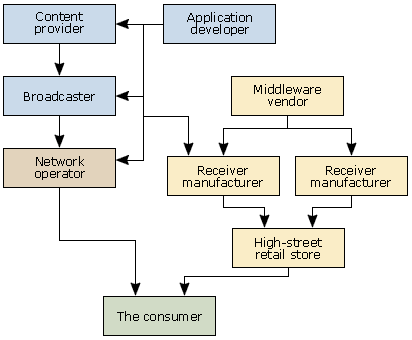
Since the broadcasters control the receiver specification, they also control the applications that will run on those receivers by their choice of middleware. Since different broadcasters can choose different middleware systems, a content provider may have to re-author any applications if they want to sell that content to other broadcasters.

So, in the current DTV markets, the broadcasters are pretty much full control , and everyone else has to go along with their decisions. Which is good for the broadcasters, right? Yes and no. The broadcasters get the control, but because they own the receivers, that can mean that a lot of capital is invested in hardware and the associated costs (maintaining inventory, technical support, upgrades and replacements, etc.)

For this reason, this situation isn't necessarily that desirable for the broadcasters. Ideally, they would like to dump these kinds of costs on to someone else. So the people who actually have the least risk in this situation is the middleware supplier, since there aren't many proprietary middleware vendors and no-one really wants to develop a new middleware stack on their own. The cost of competing with companies like NDS, OpenTV, and Canal+ is so high that very few companies can afford to do it, especially since existing middleware vendors have pre-existing relationships with broadcasters, STB vendors, CA companies and other players in the DTV business.

Now that we've seen the vertical market, let's look at the picture of how an open standard would work in an ideal  
world:

A horizontal (open) market for digital television.



In this case, the consumer can buy a receiver from any electronics store and expect it to work with any digital TV broadcaster, just like the current situation with analogue broadcasting. The subscription to the broadcaster simply gets the consumer a decryption module that allows you to watch the broadcasts. The user now no longer requires a separate set-top box for every broadcaster that he or she wishes to subscribe to, so it's possible to subscribe to different broadcasters for different packages in a much easier way than is currently possible.

Since the broadcaster no longer controls the specification of the receivers, receiver manufacturers are in a better position to add value and compete with each other, driving improvements for the customer. Similarly, since the middleware is now standardized and open, content developers can distribute their content to more broadcasters without having to re-author all their applications.

Middleware vendors are the people who suffer most in this picture, because the barrier to entry is now much lower and it's easier for new companies to implement middleware and compete with existing vendors. This drives down the cost of the middleware, but it is also likely to dramatically increase the size of the overall market as broadcasters who can't use proprietary middleware solutions (e.g. public broadcasters) start to deploy digital services.

Ideal worlds are nothing like the real world, of course, and so this situation isn't one that's likely to happen any time soon. The more likely scenario is a mixture of horizontal and vertical markets where broadcasters use open standards for the middleware. This enables the content developers to sell their content without having to re-author it, and makes life easier for receiver manufacturers, since they have a wider choice of middleware suppliers.

In the long run, this model would probably move closer to our ideal situation, but this will take some time.

So, having learned a little of the politics and reasons behind MHP, hopefully the rest of this site can give you some more information about the MHP standard. This site assumes that you know at least a little about digital TV broadcasting and in particular DVB. If you're unfamiliar with any of the following terms, it's probably best if you take a look at the [introduction to digital broadcasting](http://www.interactivetvweb.org/tutorials/dtv_intro) before exploring the rest of the site:

* DVB service
* Service information
* Section filter
* MPEG-2
* Transport stream
* DSM-CC
* Object carouse

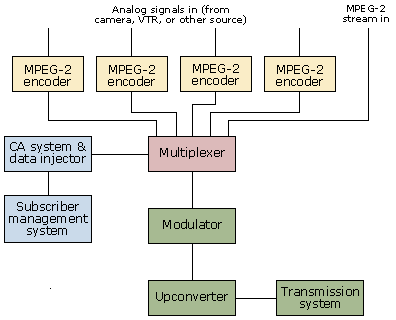
## Broadcast Engineering Basics

The purpose of this section is to provide an introduction to [digital TV](http://www.interactivetvweb.org/tutorials/dtv_intro/broadcast_engineering_basics) transmission issues for those of you who are interested. None of the material in here is really needed for an MHP developer, or even for someone developing MHP middleware. If you want that, take a look at the [introduction to MPEG and digital TV systems](http://www.interactivetvweb.org/tutorials/dtv_intro/dtv_intro).

Instead, this is an introduction to how data gets from the camera or tape to the viewer's screen. This is mainly about broadcast engineering, and the elements of the system that software developers don't usually see. This is by no means [complete](http://www.interactivetvweb.org/tutorials/dtv_intro/broadcast_engineering_basics), and is nothing more than a sketch to show you some of what goes on in the rest of the system.

A typical digital TV transmission setup looks something like this:

The components that make up a typical broadcast system.



This equipment is normally all connected together using high-speed connections like SDI (Serial Digital Interface) or ASI (Asynchronous Serial Interface) which are standard in the TV field. In addition to this, all of the equipment will be connected via ethernet to a control system and monitoring equipment to make sure that nothing goes wrong (or that if something does go wrong, the viewer doesn't see it). There will normally be a large number of some of these components, including some redundant spares in the event of problems. A typical head-end will contain many MPEG encoders and multiplexers, for instance. Now that we've seen how it's put together, let's examine each of these components in more detail.

## The encoder

The encoder is used to take an analog signal and convert it to MPEG-2. This is more commonly used in live shows - for other shows, we may have a selection of pre-encoded MPEG streams that we can [play](http://www.interactivetvweb.org/tutorials/dtv_intro/broadcast_engineering_basics) out from a dedicated playout system. This playout system is usually a highly customized PC or workstation with a large high-speed disk array and a number of digital interfaces for transmitting the data to the rest of the transmission system.

An encoder can generate two types of MPEG stream. Constant bit-rate streams always have the same bit-rate, no matter what the complexity of the scene they contain. If the signal is too complex to be coded at the specified bit-rate, the quality of the encoding will be reduced. If the scene takes less data to code than the specified bit-rate, it will be stuffed with null packets until the correct bit-rate is reached. This makes later parts of the processing easier, because the fact the bit-rate does not change makes things easier to predict later, but it does waste bandwidth.

Most encoders can now produce variable bit-rate MPEG streams as well. In this case, the bit-rate of the stream can be adjusted dynamically, as more or less bandwidth is needed to encode the images with a given picture quality. Since some scenes take significantly more bandwidth to encode than others, this lets the picture quality be maintained throughout a show while the bandwidth changes. The fact that the bit-rate of the stream can change doesn't mean that it will reach higher levels than a constant bit-rate encoding of the same stream of course: the operator can usually set the maximum bit-rate that the encoder can use, and the encoder will reduce the quality of the encoded output, if necessary to meet this.

Most broadcasters today use variable bit-rate encoding because it offers better quality while using lower bandwidth. In particular, variable bit-rate encoding lets us make maximum use of the available bandwidth at the multiplexing stage.

## The multiplexer

|  |  |  |
| --- | --- | --- |
| Java ME and Java Card Technology |  | Java Logo |

Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on embedded and mobile: microcontrollers, sensors, gateways, mobile phones, personal digital assistants (PDAs), TV set-top boxes, printers and more. Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. Applications based on Java ME are portable across many devices, yet leverage each device's native capabilities. Red Arrow [Read More](http://www.oracle.com/technetwork/java/javame/about-java-me-395899.html)

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| http://www.oracleimg.com/us/assets/spacer.gif |  | http://www.oracleimg.com/us/assets/spacer.gif |

|  |  |  |
| --- | --- | --- |
| [[Phone](http://www.oracle.com/technetwork/java/javame/javamobile/index.html)](http://www.oracle.com/technetwork/java/javame/javamobile/index.html)  **Java for Mobile Devices** | What is Java for Mobile Devices? Java for Mobile Devices relies on Java Platform, Micro Edition (Java ME) to deliver applications and services to all types of mobile handsets, ranging from price-efficient feature phones to the latest smartphones. Java is currently running on over 3 billion phones and offers unrivaled potential for mobile developers worldwide. Red Arrow [Read More](http://www.oracle.com/technetwork/java/javame/javamobile/overview/about/index.html) |  |
| [[Embedded Client](http://www.oracle.com/technetwork/java/javame/embedded/index.html)](http://www.oracle.com/technetwork/java/javame/embedded/index.html)  **Java Embedded** | What is Java for Embedded? Using Java Technology for Embedded enables you to develop highly functional, reliable, portable and secure applications for today's more powerful embedded systems. Oracle offers a full range of products, services, and support that makes it easy for you to develop using Java Technology in your embedded projects.  Red Arrow[Read More](http://www.oracle.com/technetwork/java/javame/embedded/overview/index.html) |  |
| [Java TV](http://www.oracle.com/technetwork/java/javame/javatv/index.html)  **Java TV** | What is Java for TV? Java TV refers to JSR-927, the Java Community Process (JCP) specification providing API's for digital TV-related capabilities for set-top boxes, Blu-ray Disc players, and other digital media devices. Java TV is an optional package which sits atop the Connected Device Configuration, Foundation Profile, and Personal Basis Profile (CDC/FP and PBP).  Red Arrow[Read More](http://www.oracle.com/technetwork/java/javame/javatv/overview/about/index.html) |  |
| [[Java Card](http://www.oracle.com/technetwork/java/javame/javacard/index.html)](http://www.oracle.com/technetwork/java/javame/javacard/index.html)  **Java Card Technology** | What is Java Card Technology? Java Card technology provides a secure environment for applications that run on smart cards and other devices with very limited memory and processing capabilities. Multiple applications can be deployed on a single card, and new ones can be added to it even after it has been issued to the end user. Java Card also includes a set of unique tools for developing new products. Red Arrow[Read More](http://www.oracle.com/technetwork/java/javame/javacard/overview/about/index.html) | Red Arrow |

|  |  |
| --- | --- |
| JAVA FOR MOBILE DEVICES |  |
|  | |  | | --- | | Java for Mobile Devices is a set of technologies that let developers deliver applications and services to all types of mobile handsets, ranging from price efficient feature-phones to the latest smartphones. Java is is currently running on over 3 billion phones worldwide, and growing. It offers unrivaled potential for the distribution and monetization of mobile applications.  At the core of the Java Mobile Platform is Java Platform, Micro Edition (Java ME). Java ME provides a robust, flexible environment for applications running on mobile and other embedded devices: mobile phones, TV set-top boxes, e-readers, Blu-Ray readers, printers and more. For over a decade, Oracle has been working along with leading mobile and embedded companies to develop the Java ME Platform through the [Java Community Process](http://www.jcp.org) (JCP). A key achievement has been the definition of the Mobile Services Architecture (MSA), setting a baseline of mobile APIs that developer can target within their applications. In 2011, Oracle and partners will be working within JCP to drive Java ME.next - a proposal for the modernization of Java ME .  In addition to its role within JCP, Oracle is also a provider of high performance Java ME implementations and developer technologies being used to deploy tens of thousands of applications worldwidein the mobile and embedded markets, including:   * [Oracle Java Wireless Client](http://www.oracle.com/us/technologies/java/ojwc-170429.html): a multitasking Java ME runtime optimized for the leading mobile phone platforms. * [Java ME SDK](http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk/index.html): a state-of-the-art toolbox for developing and testing mobile applications. * [Light Weight UI Toolkit (LWUIT)](http://www.oracle.com/technetwork/java/javame/javamobile/download/lwuit/index.html): a compact library for the creation of rich user interfaces. * [Oracle Java ME Embedded](http://www.oracle.com/technetwork/java/embedded/downloads/javame/index.html): designed and optimized to meet the unique requirements of small, low power devices. | |

|  |  |
| --- | --- |
| JAVA TV | Java Logo |

Java TV is a Java ME-based technology that provides a performant, secure, and easy to implement solution for developing Java applications that run on TV and set top box devices. Using the Java TV runtime, a developers can easily create applications, such as Electronic Program Guides (EPG's), Video-on-Demand (VOD) clients, games and educational applications, applications for accessing internet data (e.g. weather, news tickers, social networking), and, on most Blu-ray Disc titles, the user interface and bonus content.

http://www.oracle.com/technetwork/java/javame/index.html

http://www.oracle.com/technetwork/java/javame/javatv/download/index.html

http://www.oracle.com/technetwork/java/javame/javatv/overview/getstarted/index.html

http://www.oracle.com/us/technologies/java/mobile/wireless-client/overview/index.html

http://www.oracle.com/technetwork/java/javame/about-java-me-395899.html

http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk/index.html

http://www.oracle.com/technetwork/java/javame/javamobile/download/overview/index.html

http://www.oracle.com/technetwork/java/javame/javamobile/download/lwuit/index.html

Java Tutorial: Java TV and Java ME (Version 8)

<http://www.interactivetvweb.org/tutorials/javatv>

<https://today.java.net/pub/a/today/2005/02/09/j2me1.html>

<http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html#sun_java_me_sdk-3.0-rr-oth-JPR>

<http://docs.oracle.com/javame/config/cdc/cdc-opt-impl/ojmeec/1.0/reference/html/z4000c841293984.html>

<http://www.code4tv.com/c/downloads>

<http://www.oracle.com/technetwork/java/javame/javatv/documentation/index.html>