BBC

R&D White Paper

WHP 030

May 2002

An introduction to MHP 1.0 and MHP 1.1

J.C. Newell

Research & Development
BRITISH BROADCASTING CORPORATION

BBC Research & Development
White Paper WHP 030

An Introduction to MHP 1.0 and MHP 1.1

J.C. Newell

Transmission Systems Group

Abstract

The DVB Multimedia Home Platform (MHP) is a new open standard platform for interactive TV and multimedia services. This document provides an update to the DVB work on the MHP following the recent publication by ETSI of two new specifications, MHP 1.0.1 and MHP 1.1.

The first section provides a general introduction to the MHP and describes the current activities and future direction of the MHP initiative. The second section provides a more detailed technical overview of the MHP specifications.

Key words: interactive, API, MHP, JavaTV

White Papers are distributed freely on request.

Authorisation of the Head of Research is required for publication.

BBC Research & Development
White Paper WHP 030

An Introduction to MHP 1.0 and MHP 1.1

Contents

BBC Research & Development
White Paper WHP 030

Section I

An Introduction to the DVB Multimedia Home Platform

This section provides an introduction to the DVB Multimedia Home Platform (MHP) and describes the current status and future direction of the MHP initiative.

1. Introduction

Whilst many broadcasters have launched or are planning digital television services there has not been a consensus on the format for interactive applications such as electronic programme guides and information services. This has led to fragmented markets, where consumers are unable to access all of the services available to them with a single receiver system. Broadcasters, including the BBC, have as a result been forced to duplicate their application development work. This situation threatens to prevent the development of a thriving international market for interactive TV content.

In response to this problem the DVB Project has developed a specification for a new open standard platform for interactive TV and multimedia services, designed to address the requirements of both commercial and public sector broadcasters. The primary goal of the MHP is to enable the birth of horizontal markets, where there is open competition between content providers, network operators and platform manufacturers at each level in the delivery chain. A key requirement is to achieve full interoperability between applications and different implementations of the MHP. Broadcasters must have confidence that their applications will behave predictably and consistently on all platforms that conform to the specification.

At first, the MHP is most likely to succeed in new markets for digital TV where there is no existing population of receivers to support and no migration problems to resolve. However, if successful in these new markets the MHP may look increasingly attractive in existing markets such as the UK, as a path for future evolution.

2. Commercial requirements

In 1997 the Commercial Module of the DVB Project formed the MHP group in order to draft a set of commercial requirements . A primary aim of the commercial requirements is to ensure that there will be free and non-discriminatory access to the MHP. Some other significant requirements are:

- The MHP must be based on open standards and utilise existing DVB specifications.
- The MHP must be independent of specific hardware technologies and operating systems.
- Support should be provided to facilitate the migration of existing interactive TV services to the MHP.
- A modular and scalable approach should be adopted, permitting the future evolution of the MHP whilst maintaining backwards compatibility.
- Sufficient information and compliance tests should be provided to enable the self-certification of implementations by manufacturers.

The development of commercial requirements by the MHP group is continuing and may eventually encompass the needs of PVRs, Home Networks and Broadband IP delivery.

3. Competing technical solutions

A technical group called TAM is responsible for developing technical specifications to satisfy the commercial requirements. Initially, many competing technologies for the MHP were proposed to TAM (e.g. MHEG-5, HTML, OpenTV, and MediaHighway). However, it became clear that it would be impossible to reach a consensus solution

based on any of these existing technologies, due to the extent that they were already deployed, and the commercial pressures that this situation created.

However, it was identified that all the responses had one element in common; they all proposed to use Java at some point in their future evolution path. DVB therefore took the decision to adopt the Java programming language for the MHP and develop a version called DVB-J that would be suitable for interactive TV applications.

Compared to the existing systems, Java can provide more capabilities and flexibility but at the expense of greater complexity and cost. The DVB decision therefore represented a change in focus to a more advanced system and a longer development timescale for the MHP. It remains to be seen whether this strategy will be successful whilst in the meantime other systems have had a greater opportunity to become established.

Another decision taken by DVB was to support optional “plug-in” decoders for “legacy” application formats that have already been deployed. DVB will not define plug-ins for specific systems, but has defined a generic interface and signalling to support them.

4. Why Java?

Given that the original submissions to the TAM group proposed such a variety of disparate, competing technologies it may seem remarkable that they all proposed Java in their future evolution path. However, there are good reasons why Java was able to achieve this consensus.

Firstly, Java has emerged at a time when broadcasters are concerned about the convergence of broadcasting and the Internet. The Java language was designed from the beginning for a networked environment. It has in-built support for internet protocols and has many security features to protect platforms from hostile or unintentionally damaging applications.

Secondly, the Java runtime environment does not favour one manufacturer’s microprocessor or operating system technology over another. The adoption of Java by an industry group therefore has fewer commercial and technical repercussions than would otherwise be the case.

These features have enabled Java to achieve a wide cross-industry support. This means that interactive TV application development can benefit from the tools and expertise that has been developed in other industries. However, it has to be acknowledged that the use of Java in consumer electronics (CE) equipment is relatively immature.

5. What is Java?

Java is a widely used general purpose programming language and run-time environment developed by Sun Microsystems. The source code for Java applications is compiled to produce Java byte code. This byte code contains instructions that can be interpreted by a Java Virtual Machine (VM), a software model of a hypothetical processor that can be implemented by any computer with sufficient resources. Java byte code can therefore run without modification on all computers that have implemented a JavaVM. This “write once, run everywhere” approach is one of the key advantages of Java but does have an impact on execution speed.

The Java run-time environment (JRE) consists of a Java VM and a set of function libraries called "packages". These packages supply the standard functions of the Java language and provide access to resources such as the user interface. DVB has defined additional packages for the MHP that provide TV-specific functions.

Sun Microsystems maintains consistency in the implementations of the JRE, by imposing an extensive test suite. Programmers therefore have a reasonable guarantee of the behaviour of their applications when they arrive on a Java platform. A similar test suite is under development for the MHP, incorporating Sun's test suite and providing additional tests for the DVB defined elements.

6. Profiles and options

DVB has initially specified three profiles for the MHP. These are not intended to supersede each other but are expected to co-exist in the marketplace. They provide a backwards-compatible hierarchy, as shown in Figure 1, supporting a range of capabilities suited to different application areas and product types.

The **Enhanced Broadcast profile** is the baseline profile of the MHP. It is intended for use in broadcast only systems where applications and data are delivered by the broadcast service but where there is no return channel. Applications designed for this profile provide local interactivity with the user e.g. EPGs and information services.

The **Interactive Broadcast profile** is a superset of the Enhanced Broadcast profile, adding support for a bi-directional data channel using any of the return channel systems defined by DVB. This allows data to be returned to the broadcaster and enables applications to support e-commerce and tele-voting.

The **Internet Access profile** is a superset of the Interactive Broadcast profile, adding resident applications that provide access to the most common types of Internet services, i.e. WWW, Email, and Usenet News (optional). The profile allows links to be provided between interactive TV applications and Internet services and vice-versa.
For a more detailed description of the Internet Access profile see BBC R&D White Paper WHP 018.

DVB has also defined an optional content format called **DVB-HTML**. This is a tightly specified HTML content format designed for broadcast use. DVB-HTML can be supported as an option in either the Interactive Broadcast or the Internet Access profiles.

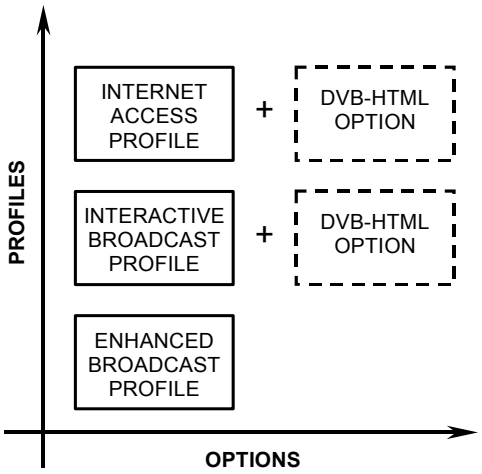


Figure 1 - The current family of MHP profiles and their respective options

It is likely that further profiles and options will be defined in future specifications to address new commercial requirements defined by the DVB MHP group. The philosophy will be to preserve backwards compatibility except in cases where an ambiguity or problem makes parts of the earlier specifications unusable.

Table 1 gives approximate hardware requirements of the three profiles and DVB-HTML option, compared to some current open and proprietary platforms[1]. The MHP clearly requires more memory and processing power than the current generation of receivers. However, the capabilities of receivers are continually increasing as technology evolves and it is expected that the MHP will become economically viable by the end of 2003. In the longer term the existence of the MHP as a single open standard should encourage competing manufacturers to reduce the cost significantly compared to proprietary systems.

PLATFORM	PROCESSOR[2]	RAM	FLASH/ROM
Basic STB	30 MHz+	1-2 MB	1-2 MB
MHEG-5	50 MHz+	4 MB	2 MB
Typical proprietary system (e.g. OpenTV)	50 MHz+	4-8 MB	4 MB
MHP Enhanced Broadcast Profile	80-130 MHz+	8-16 MB	4 MB

[1] Figures published by Philips, Sony, Panasonic and Nokia.
[2] 32-bit processor assumed in all cases.

MHP Interactive Broadcast Profile	80-130 MHz+	8-16 MB	8 MB
MHP Interactive Broadcast Profile + DVB-HTML option	150-200 MHz+	16-32 MB	16 MB
MHP Internet Access Profile	150-200 MHz+	16-32 MB	16 MB

Table 1 - Typical hardware requirements of MHP profiles and alternative platforms.

7. MHP specifications

At this point in time DVB has published three MHP specifications:

1. **ETSI TS 101 812 V1.1.1: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0"**

   This was the first MHP specification to be published and defined the Enhanced Broadcast and the Interactive Broadcast profiles. It has been superseded by MHP 1.0.1 and is unlikely to be used in any commercial implementations.

2. **ETSI TS 101 812 V1.1.2 : "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.1"**

   This was the second MHP specification to be published and corrects over 1500 errors, inconsistencies and omissions found in the first version of MHP 1.0. Again it defines only the Enhanced Broadcast and Interactive Broadcast profiles. This specification will be superseded by MHP 1.0.2, due to be published in mid-2002, which includes further corrigenda. The first examples of commercial MHP products will be based on MHP 1.0.2 and this will become the reference version of MHP 1.0.

3. **ETSI TS 102 812 V1.1.1: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1"**

   This is the most recent MHP specification to be published. It is a superset of MHP 1.0.1 and introduces the following extensions:

   - Internet Access profile
   - DVB-HTML option
   - Application download over the interaction channel
   - Persistent storage and caching of MHP applications
   - A "plug-ins" interface for legacy application formats
   - Smart Card reader (for e-commerce rather than Conditional Access purposes)

8. Current and future activities

Several groups within the DVB Technical Module are currently working on MHP related activities:

- DVB-TAM (Technical Aspects of the MHP) is responsible for developing the technical specifications for the MHP. It is currently inactive except for the MHP-SPEC (MHP Specification maintenance) subgroup which is dealing with corrigenda and will produce the next specification (MHP 1.0.2) early in 2002.

- MEG (MHP Experts Group) is responsible for the provision and maintenance of test suites for conformance testing MHP implementations.

- MUG (MHP Umbrella Group) is providing support to organisations outside DVB wishing to achieve interoperability with the MHP.

Within the DVB Commercial Module the MHP group is currently developing further commercial requirements for:

- Content authoring and interchange specifications to support MHP migration.

- DVB-HTML compliance testing, interoperability and authoring guidelines.

- The MHP-AUTH (MHP Authentication) subgroup is studying the commercial requirements for security authorities issuing authentication certificates for MHP applications.

Beyond this work supporting the existing MHP specifications, the MHP group is also studying commercial requirements that would extend the capabilities of the MHP. Some of these have arisen as a result of recent discussions

on the future direction of DVB as a whole, where the emphasis has been on the integration of the IP world and the DVB world. Future extensions to the MHP specifications under discussion include:

- Support for Personal Video Recorders.
- Home networks.
- Broadband IP services.
- Support for alternative audio/video streaming formats (e.g. MPEG-4).
- A profile of the MHP for mobile applications.

There is some concern that the last of these is out of scope for DVB or might undermine the success of the current MHP specifications.

9. Conclusions

The DVB MHP is a new standard open platform for interactive television and multimedia services that can be implemented in set-top boxes, integrated digital televisions and PCs. It is suitable for all means of delivery and should meet the current needs of most broadcasters.

DVB has adopted the Java programming language for the MHP and has created a lightweight version suitable for broadcast applications called DVB-Java. It is possible to support existing application formats using optional plug-in decoders.

Currently, three profiles have been defined and an optional content format called DVB-HTML, allowing manufacturers to develop a range of products providing different capabilities.

Section II

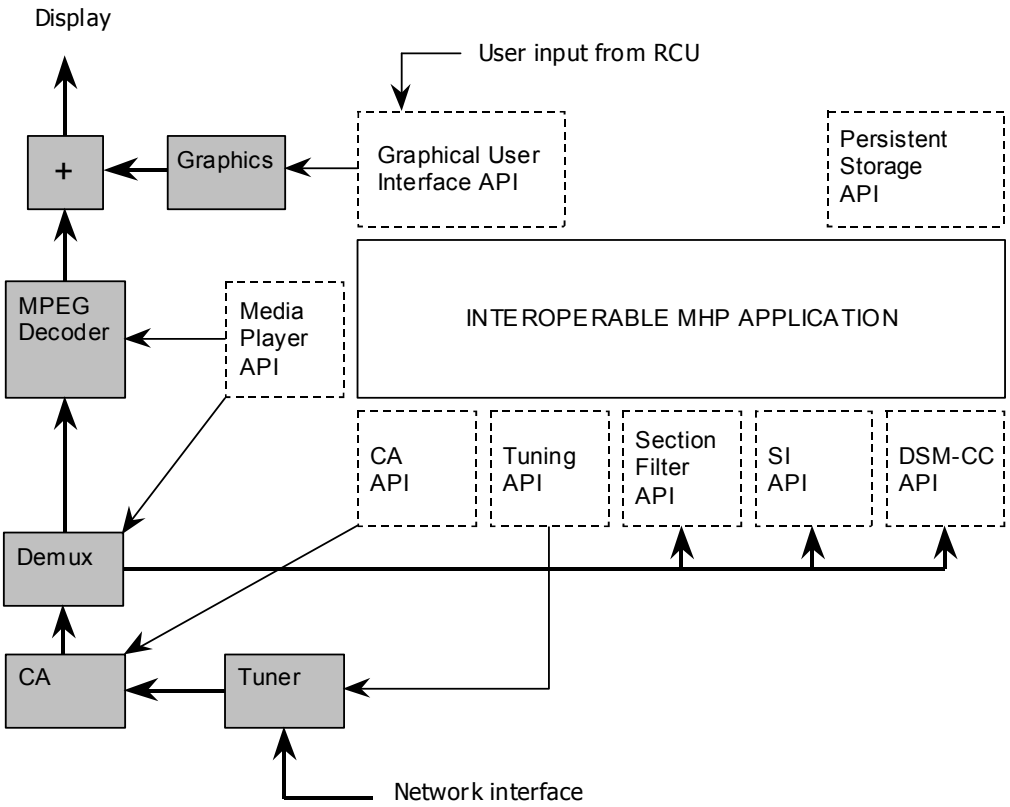
A detailed technical overview of the MHP specifications

This section provides a detailed technical overview of the current MHP specifications.

1. Introduction

Figure 2 shows a functional diagram of the Enhanced Broadcast profile. The shaded boxes represent the hardware elements of a typical digital receiver and the thick line shows the audio/video pipeline from the network interface to the display. An interoperable MHP application, represented by the white box with the solid border, is running on the receiver, having been loaded from the broadcast stream. The application can control or receive information from the hardware elements of the box via the DVB-J APIs (the boxes with dashed borders). Not shown are many additional DVB-J APIs that provide higher level functions (e.g. service discovery and service selection). Standardising the MHP APIs enables applications to run consistently on different receivers, independently of how they have been implemented.

Figure 2 - Functional diagram of the Enhanced Broadcast profile.



The MHP specification describes how applications are signalled, delivered and loaded, including the associated lifecycle and security issues. It then describes the run-time environment in which the applications operate once they are running.

In the rest of this section an overview is given of all the elements of the MHP specifications. Unless otherwise stated, all elements are part of the baseline MHP 1.0 specification and will be supported by all MHP profiles.

2. MHP application model

An important feature of the Java language is the application model provided for Java Applets (Java applications launched from within a web page). This model describes the complete lifecycle of Java applets; how they are downloaded across the Internet, how they are launched and how they are stopped. Unfortunately, the Applet model is unsuitable for interactive TV applications because it requires a bi-directional delivery channel. DVB has therefore created a new application model for MHP applications (called "Xlets") that is suitable for both unidirectional broadcast systems and bi-directional networks.

2.1 Application signalling

The lifecycle of MHP applications is closely related to the process of service selection, which can either occur when requested by the user (using the manufacturers' resident navigator software) or when requested by an MHP application.

When a new service is selected the platform immediately looks in the broadcast stream for an Application Information Table (AIT). This carries information about the applications that are permitted to run in the new service and in particular identifies "autostart" applications that must be launched as soon as a service is selected. The AIT also identifies the location of applications and carries signalling that enables broadcasters to control their lifecycle after they have been launched.

If an application is already running at the time that service selection takes place it may continue to run if it is signalled as a valid application for the new service. After service selection, it is possible for alternative applications signalled to be launched by the user, via the resident navigator software, or by an MHP application using the Application Listing and Launching API.

2.2 Application lifecycle

Once it is running, the lifecycle of an application can be controlled either by the manufacturers' resident navigator software, by the broadcaster via the signalling in the AIT, or by other MHP applications.

The MHP application model defines a Java Interface (`javax.tv.xlet.Xlet`) that must be implemented in the class used to initiate an MHP application. The Xlet interface provides four methods that are called by the platform to inform the MHP application of impending lifecycle changes.

A state diagram showing the lifecycle of MHP applications is shown in Figure 3. Initially, after the application has been downloaded it is in the "Loaded" state.

- The initXlet method may then be called to notify the application that it should initialise itself and move to the "Paused" state.
- The startXlet method may then be called to tell the application that it should move to the "Active" state and start running.
- The pauseXlet method may be called to inform the application that it should move back to the "Paused" state and that it should minimise its use of resources. It may move back to the Active state if the startXlet method is called again.
- The destroyXlet method can be called in any state and is used to warn the application that it is about to be killed. It should therefore save state information and release resources as soon as possible.

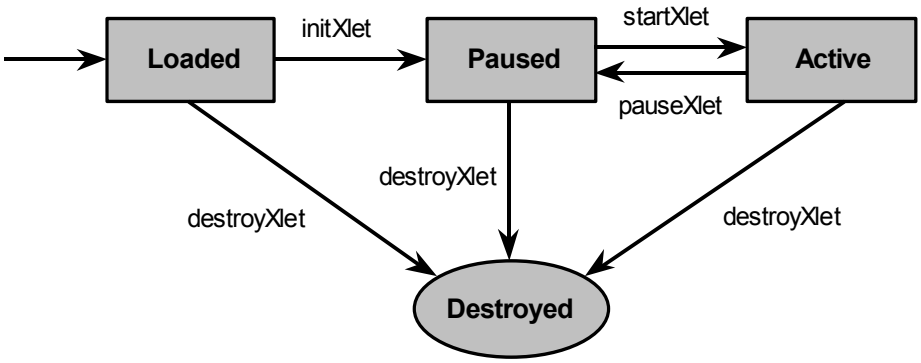


Figure 3 - Xlet lifecycle state diagram.

2.3 Extensibility and MHP 1.1 extensions

The MHP application model has been designed to be extensible. It is possible for an MHP to support multiple, concurrent applications that are either co-operating (designed to communicate with each other and share resources) or non-cooperating (independent and competing for resources). However, for the time being it is unlikely that commercial MHP implementations will have the resources to run more than one MHP application at a time.

MHP 1.1 extends the application model to allow applications to be stored and launched from persistent memory. Applications may either be cached pro-actively by an MHP or stored in response to a request from another MHP application, subject to user consent and platform resource limitations. In both cases the aim is to improve the speed by which applications can be loaded.

MHP 1.1 also provides extensions to the application signalling to allow applications to be downloaded across the bi-directional interaction channel and to allow legacy application formats to be delegated to "plug-in" decoders.

3. Broadcast Data Transport system

Applications and data are delivered to an MHP across the broadcast channel using the DVB Object Carousel protocol. This uses a set of messages[3] that either contain a file, a directory or a Service Gateway object (equivalent to a root directory). The messages are mapped into modules of the DVB Data Carousel and are broken down into Download Data Blocks (DDBs), which are broadcast repeatedly in MPEG Sections. The DDBs are interleaved and can be broadcast at different rates, depending on the priority of the data. The MHP profile of the Object Carousel is compatible with the UK MHEG-5 profile.

From the application developer's point of view the Object Carousel behaves just like any other file system, except that there may be a high latency if a file isn't currently held in the local cache. Access to files can be obtained using the standard `java.io` API or the lower level `org.dvb.dsmcc` API.

4. Security model

The MHP is an open system and can receive applications from a wide range of sources, including the Internet in some profiles. It therefore needs a comprehensive security model to protect the platform from damage and to ensure fair and equal access by all service providers. It also needs to protect the privacy of the end-user and prevent any unauthorised use of resources that might have cost implications e.g. the modem.

4.1 Java security mechanisms

The MHP utilises the standard security mechanisms provided in Java:

[3] The format (BIOP messages) is taken from the CORBA specification.

- Before loading Java classes are scanned and vetted by a Byte Code Verifier (BCV) that identifies and rejects illegal Java byte code.

- Java Class Loaders manage the namespace of downloaded classes separately from that of the resident system classes to ensure that application cannot "spoof" a resident class.

- The BCV and Java VM work together to oppose some common security attack strategies, aided by the fact that Java does not permit any direct memory access.

4.2 Application authentication

The MHP provides a system of digital signatures and certificates that can be used to verify the source and integrity of MHP applications and data. The system allows organisations to sign the applications and data within a complete “tree” of the file system carried in an Object Carousel, guaranteeing the validity of all the content using a single signature.

Certificates stored in the MHP provide public keys that are used to verify application signatures. A hierarchical certificate management scheme has been defined with one or more central authorities providing the root certificates.

Applications do not necessarily have to make use of these security mechanisms and may be broadcast without being signed. However, unsigned applications are prevented from using APIs that have security, privacy or cost implications and must operate using a restricted subset called the "sandbox".

4.3 Permissions system

Signed and authenticated MHP applications can request permission to access a specific resource that is outside the sandbox (e.g. the modem). The user is able to grant or deny access to these resources on an application, organisation or global basis. Each permission is granted individually so that an application could be given access to the modem but not to the CA system for example.

4.4 Interaction channel security

The MHP supports standard Java and Internet security mechanisms and protocols to ensure the security and privacy of communications via the bi-directional interaction channel.

5. Mono-media content formats and graphics model

The MHP supports a superset of the mono-media content formats (e.g. for fonts and images) supported in the UK MHEG-5 profile. In particular, the same resident font is supported (Tiresias) with an extended Latin character set. The MHP also supports downloadable fonts using the PFR file format.

As well as supporting the ability to display full screen MPEG I-frames the MHP supports "drip-feed" MPEG decoding, allowing I and P frames to be fed one by one, at any desired rate, in order to produce bandwidth efficient animations.

The graphics model used for the MHP was developed from the UK DTG MHEG-5 profile and is backwards compatible.

6. DVB-J

DVB has created a version of Java called DVB-J for the MHP. The goal has been to produce a version that is suitable for embedding in consumer electronics equipment, replacing PC-centric elements with more suitable alternatives and providing support for TV and DVB specific requirements.

The definition of DVB-J consists of a list of all the Java APIs that are available to MHP applications, together with a description of their semantics and required behaviour. This section provides an overview of the DVB-J APIs whilst a complete listing and description of the individual packages is given in Appendix A.

6.1 Core Java language

DVB-J takes as its starting point the JDK 1.1 version of the standard Java language with some omissions and with a few elements of JDK 1.2 and pJava 1.2. This includes the core packages of the Java language (i.e. `java.lang`, `java.util`, `java.io`, etc) that are familiar to all Java programmers.

6.2 User interface

The standard Java graphical user interface provided by `java.awt` is highly computer-centric and is not very appropriate for television displays and remote control units. For this reason, the standard GUI widget set (visible user interface components) provided in `java.awt` is omitted and is replaced by the set defined in the HAVi specification for networking digital home entertainment products. The MHP does not currently support any other elements of HAVi but may do so in the future if DVB defines extensions to support home networks. The HAVi Level 2 User Interface provides a basic set of GUI widgets, such as text boxes.

It is also possible to design and download custom user interface components for the MHP using the standard Java "Lightweight Components" model [II-1].

6.3 Media player

DVB-J uses the media player provided in the standard Java Media Framework (JMF 1.0) to control the presentation of broadcast audio and video streams. It is also possible to play audio clips from downloaded files.

6.4 TV-specific APIs

DVB-J provides a substantial set of Java packages to support requirements that are specific to the TV environment. Some of these APIs are specific to DVB protocols e.g. providing access to DVB Service Information. However, in other cases the APIs are more abstract and could also be applied in non-DVB systems (e.g. ATSC DASE). In particular, the MHP partially supports a set of packages referred to as "JavaTV", which was developed in an open initiative led by Sun Microsystems.

The decision to support JavaTV in DVB-J was quite controversial because it duplicates some of the functionality provided by the DVB-specific APIs. However, it is hoped that this will allow applications that conform to certain restrictions to run on both the MHP and non-DVB systems without modification.

6.5 Profiles, options and versions

All compliant MHP implementations can be assumed to support the baseline Enhanced Broadcast profile defined in MHP 1.0. If MHP applications wish to exploit the capabilities of other profiles or options it is necessary to do a run-time check to confirm that these are supported.

DVB-J supports a set of properties that can be queried by applications at run-time using the `System.getProperty()` method. Some of these properties are used to indicate the profile and specification version supported by the platform whilst others are used to indicate support for a specific option (e.g. DVB-HTML).

Alternatively, applications may be signalled in the AIT as requiring a particular profile to be supported. In this case they will not be loaded by platforms that are unable to satisfy their requirements.

7. MHP 1.1 extensions

The MHP 1.1 specification defines a new "Internet Access" profile, an optional content format called "DVB-HTML" and some additional APIs for DVB-J listed in Appendix A.7.

7.1 Internet Access profile

The Internet Access profile allows manufacturers to incorporate a set of "built-in" or "resident" applications that can provide access to specific internet services. The profile requires that a web browser and an email client application are supported, with the option to include a Usenet newsgroup client and provide support for streaming media services. Since these internet client applications are permanently installed within the platform they can be implemented in a manufacturer specific way and are free from the constraints that apply to MHP applications. Very little is specified about the characteristics of the internet client applications to allow manufacturers some freedom of implementation.

The profile supports a set of Internet Client Control APIs that allow MHP applications to launch the internet client applications and direct them to specific locations or addresses. Similarly, the resident applications are expected to

support locators pointing at DVB services e.g. hyperlinks within web pages allowing the user to trigger service selection. A detailed overview of the profile is given in Reference II-2.

7.2 DVB-HTML

DVB-HTML is a tightly specified X-HTML based content format[4] intended for broadcasters who wish to deliver HTML content to MHP receivers over the broadcast and interaction channels. It is an option in either the Interactive Broadcast or Internet Access profiles and therefore implementations must also support the Java part of the MHP specifications.

In contrast to the internet client applications required in the Internet Access profile, the characteristics of the “browser” in the DVB-HTML option is specified in great detail to ensure interoperability in a broadcast system. It also includes the definition of a "Java bridge" to allow DVB-HTML to exploit DVB-J APIs and DVB-J applications to control the display of DVB-HTML.

It is unlikely that early MHP receivers will support DVB-HTML due to the additional resources required, as detailed in Table 1.

8. Application authoring

8.1 Basic characteristics of Xlets

All Xlets must have an initial launching class that implements the `javax.tv.xlet.Xlet` interface and must have a default (no argument) constructor. This constructor is provided invisibly in the Java language by default unless any alternative constructors are defined.

8.2 Presentation

An application must create a single container called an HScene in which to display itself. This is usually full-screen in size and will normally persist for the lifetime of the application. The HScene is typically used to display a sequence of full-screen containers, each representing one of the user interface "scenes" of the application.

Each container can contain a number of visible components and containers, either HAVi components or custom "lightweight" components created by extending the `java.awt.Component` class.

8.3 Callbacks

An important aspect of the MHP is the extensive use of "callbacks" where application programmers are required to provide an implementation of a Java Interface defined in the MHP specification. These are then registered as "listeners" for specific types of event and are called by the platform when these events occur e.g. user input events or the capture of specific data from the broadcast stream. To create responsive applications it is important that these callbacks are written efficiently and return quickly since they will run in a high priority system thread and may block other critical activities. Processor intensive tasks must be confined to separate low priority threads managed by the application.

9. Conclusions

DVB has created a version of Java suitable for broadcast applications called DVB-J. This consists of the core of the standard Java language with the addition of TV-specific and DVB-specific APIs.

DVB has also defined a new application model for MHP applications and has defined an extensive security framework.

Many elements of the MHP specification have been developed from the UK MHEG-5 profile and are backwards compatible. This should make the introduction of support for the MHP on UK DTT services considerably easier if this becomes desirable in the future.

10. References

[4] DVB-HTML has been developed from modularised X-HTML, a recent W3C specification which is a reformulation of HTML 4.0 in XML.

[II-1] http://java.sun.com/products/jdk/1.1/docs/guide/awt/demos/lightweight

[II-2] "The DVB MHP Internet Access Profile", BBC R&D White Paper WHP 018, J.C. Newell, September 2001.

Appendix A

Summary of DVB-J packages in MHP 1.0 & MHP 1.1

This appendix describes all the Java packages defined in MHP 1.0.1 and MHP 1.1. The descriptions have been extracted from the Javadoc documentation of each individual package. Where significant elements of an established set of packages have been omitted by DVB it is described as a "DVB subset".

1. Core Java APIs (DVB subset)

The core of the Java language for the MHP specification is provided by the standard packages found in JDK 1.1.8 (i.e `. java.lang, java.util, java.io`, etc) with some restrictions and omissions. The most noticeable omissions for Java programmers are the `java.awt` GUI widget set (i.e. all classes below Component in the inheritance hierarchy except Container) and the `java.applet` package. However, both of these are supported in the Internet Access profile.

2. JMF 1.0 APIs

The MHP uses the standard Java Media Framework API (JMF 1.0) for presenting and controlling audio, video and other time-based media. The following packages are supported with certain restrictions:

`javax.media`
`javax.media.protocol`

3. Java Secure Sockets Extension APIs

The Java Secure Socket Extension (JSSE) is a set of packages that enable secure Internet communications. It implements a Java version of SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication.

`javax.net`
Provides optional classes for networking applications. These classes include factories for creating sockets. Using socket factories you can encapsulate socket creation and configuration behaviour.

`javax.net.ssl`
Provides the classes for the secure socket optional package. Using the secure socket classes, you can communicate using SSL or a related security protocol to reliably detect any errors introduced into the network byte stream and to optionally encrypt the data and/or authenticate the communicating peers.

`javax.security.cert`
Provides classes and interfaces for parsing and managing certificates.

4. JavaTV 1.0 APIs (DVB subset)

JavaTV is a set of generic, TV-related packages. They address the common elements of an interactive TV platform that can be defined independently of specific transport and signalling protocols i.e. the APIs are equally valid in DVB and non-DVB systems.

`javax.tv.graphics`

Provides a mechanism by which xlets may discover their root container and describes a mechanism for alpha blending.

`javax.tv.locator`
Provides a means for referencing data and resources accessible via the Java TV APIs. The Locator interface serves as an opaque reference to the location information of entities which are addressable within a television receiver. For example, locators can reference broadcast file systems, portions of service information, sources of audio/video content, and interactive applications.

`javax.tv.media`
The Java TV API uses the Java Media Framework (JMF) APIs for managing the broadcast media pipeline. This package defines extensions to JMF for use a television environment. The DAVIC media package defines additional JMF extensions.

`javax.tv.net`
This package is supported only where the option to support for IP over the broadcast channel is included. Provides access to IP datagrams transmitted in the broadcast stream. Applications implemented using the Java TV API can access IP datagrams transmitted in the broadcast stream through use of the normal datagram reception mechanisms of the `java.net` package. Unicast IP datagrams are received via the `java.net.DatagramSocket` class; multicast IP datagrams are received via the `java.net.MulticastSocket` class.

`javax.tv.service`
Provides mechanisms for accessing the service information (SI) database and APIs representing the SI elements it contains. The `javax.tv.service` package and its sub-packages are organized into different "views" of service information, as follows:

– The guide package provides APIs useful for electronic program guides (EPGs), including program schedules, program events and program ratings.

– The navigation package provides APIs to navigate through services and hierarchical service information.

– The selection package provides a mechanism to select a service, that is, tune to a service and present its content.

– The transport package provides information concerning the broadcast delivery mechanisms available to the receiver.

– The top-level service package (this package) provides basic SI APIs common to the service sub-packages.

`javax.tv.service.guide`
Provides APIs to support electronic program guides (EPGs), including program schedules, program events and program ratings. The `ProgramSchedule` interface can be used to retrieve the current `ProgramEvent`, the next `ProgramEvent`, or a `ProgramEvent` in the future. Each `ProgramEvent` can be queried for its name, start time and end time, description, and rating.

`javax.tv.service.navigation`
Provides APIs to navigate through services and hierarchical service information. The Navigation package serves two main functions:

– It provides access to detailed SI data about each Service.
– It provides a mechanism to group Services based on various filtering criteria.

`javax.tv.service.selection`

Provides a mechanism to select a `Service` for presentation. The `ServiceContext` interface abstracts receiver resources required to tune to and present the content of a service, such a tuner, MPEG decoder, screen real estate, and interactive content handlers. The `ServiceContext.select()` method constitutes a high-level mechanism by which applications may cause the receiver to tune to a desired service, demultiplex the necessary service components, present the audio and video, and launch any associated applications. Applications launched by the `ServiceContext.select()` method are said to "run" within the `ServiceContext`.

`javax.tv.service.transport`
Provides additional information about the transport mechanisms that deliver the content the SI data describes. The `SIManager` provides access to instances of the Transport interface, which is extended to represent an MPEG-2 multiplex. The subtypes of Transport are:

- `TransportStreamCollection`, representing one or more transport streams. A `TransportStream` delivers services and service information to the receiver.

- `NetworkCollection`, representing one or more networks. A `Network` typically aggregates transport streams.

- `BouquetCollection`, representing one or more bouquets. A `Bouquet` aggregates services, and may span transport stream and network boundaries.

One may use the `SIElementFilter` API to determine the services carried by individual bouquets, networks, and transport streams.

`javax.tv.util`
Provides APIs for creating and managing timer events.

`javax.tv.xlet`
Provides interfaces used by applications and the application manager to communicate. This package defines two entities: the `Xlet` and the `XletContext`. The central function of this set of interfaces is to model and manage the states an application can be in. `Xlet` is an interface that must be implemented by the primary class of an application. When an application is loaded, one instance of this primary class is created. The application is controlled via method calls on this instance. Methods of `Xlet` are used by the application manager to deliver state change requests to the application. An application is provided with an instance of `XletContext`, which it can use to deliver state change notifications to the application manager. Each `Xlet` receives its own instance of `XletContext`.

5. DVB & DAVIC APIs

`org.davic.media`
Provides various extensions to the Java Media Framework for the control of TV oriented audio / video content. Several classes and interfaces not required in the MHP specification have been removed.

`org.davic.mpeg`
Provides utility classes for common MPEG concepts. The class `ApplicationOrigin` has been removed since it is not required in the MHP specification.

`org.davic.mpeg.dvb`
Provides utility classes for common MPEG concepts as used in DVB.

`org.davic.mpeg.sections`
Provides access to MPEG-2 section filtering.

`org.davic.net`

Provides general content referencing.

`org.davic.net.ca`
Provides an interface to various features of a conditional access system.

`org.davic.net.dvb`
Provides DVB specific content referencing.

`org.davic.net.tuning`
Provides access to tuning - MPEG multiplex selection. Note that in a digital TV system the role of tuning is different from its role in an analogue TV system. The service selection API in MHP provides functionality closer to the role of tuning in an analogue TV system.

`org.davic.resources`
Provides a framework for scarce resource management.

`org.dvb.application`
Provides access to lists of applications that are available in this context and the ability to launch those applications.

`org.dvb.dsmcc`
Provides extended access to files carried in the broadcast stream. It includes some extensions to `java.io` which are generic to long-latency file systems and some concepts which are specific to the DSMCC object carousel.

`org.dvb.event`
Provides access to user input events before they are processed through the event mechanism of the `java.awt` package.

`org.dvb.io.ixc`
Provides support for inter-application communication.

`org.dvb.io.persistent`
Provides extensions to the `java.io` package for access to files held in persistent storage.

`org.dvb.lang`
Provides those core platform related features not found in the `java.lang` package.

`org.dvb.media`
Provides DVB specific extensions to the Java Media Framework.

`org.dvb.net`
Provides general networking features not found elsewhere.

`org.dvb.net.ca`
Provides extensions to the conditional access API from DAVIC.

`org.dvb.net.rc`
Provides session management for bi-directional IP connections which are session based from the point of view of an application.

`org.dvb.net.tuning`
Provides extensions to the tuning API from DAVIC.

`org.dvb.si`
Provides access to DVB service information.

`org.dvb.test`
Allows test applications to log messages during their execution and to indicate their termination condition in a platform independent manner.

`org.dvb.ui`
Provides extended graphics functionality.

`org.dvb.user`
Provides access to settings and preferences configured by the end-user.

6. HAVi level 2 GUI APIs

`org.havi.ui`
Provides classes for creating user interfaces and for painting graphics and images.

`org.havi.ui.event`
Provides interfaces and classes for dealing with events fired by AWT components.

7. Additional APIs in MHP 1.1

`org.dvb.application.inner`
Provides support for embedding other interactive applications within the user interface of a DVB-J application.

`org.dvb.application.plugins`
Provides support for inter-operable plugins in the MHP specification. Some classes and interfaces are intended for generic plug-ins and are not specific to any particular content format whilst others intended specifically for DVB-HTML plug-ins.

`org.dvb.application.storage`
Provides support for storage of applications.

`org.dvb.dom.bootstrap`
Provides the entry point to the W3C DOM APIs for Java applications.

`org.dvb.dom.inner`
Provides support for embedding DVB-HTML applications within the user interface of a DVB-J application.

`org.dvb.internet` (Internet Access profile only)
Provides a mechanism for MHP applications to control internet clients that may be present on an MHP such as a web browser or email client. The API makes no assumptions about whether an internet client and downloaded MHP application can run simultaneously, and in some cases starting an internet client may result in the calling application being killed.

`org.dvb.io.ixc`
Provides support for inter-application communication.

`org.dvb.smartcard`
Provides DVB specific support for smart card reader APIs.

Appendix B

Abbreviations

AIT	Application Information Table.
API	Application Programming Interface.
ATSC	US Advanced Television Standards Committee.
BCV	Java Byte Code Verifier.
BIOP	Broadcast Inter ORB Protocol.
CA	Conditional Access.
CE	Consumer Electronics.
CI	The DVB Common Interface.
CORBA	Common Object Request Broker Architecture.
DASE	Digital Application Software Environment defined by the ATSC.
DAVIC	Digital Audio-Visual Council.
DDB	Download Data Block.
DSM-CC	Digital Storage Media Command and Control.
DTG	Digital Television Group.
DVB	Digital Video Broadcasting Project.
DVB-HTML	A DVB defined version of HTML for interactive TV applications.
DVB-J	A DVB defined version of Java for interactive TV applications.
EPG	Electronic Programme Guide.
ETSI	European Telecommunications Standards Institute.
FLASH	A type of re-programmable ROM.
GUI	Graphical User Interface.
HAVi	Home Audio Video Interoperability specification.
HLN	Home Local Network.
IDTV	Integrated Digital Television.
IP	Internet Protocol.
IRD	Integrated Receiver Decoder.
Javadoc	A standard form of HTML documentation for Java APIs.
JavaTV	A set of generic TV-related APIs.

JDK	Java Development Kit.
JRE	Java Runtime Environment.
MEG	MHP Experts Group.
MHEG	Multimedia & Hypermedia Experts Group.
MHP	Multimedia Home Platform.
MPEG	Motion Picture Experts Group.
MUG	MHP Umbrella Group
ORB	Object Request Broker.
PFR	Portable Font Resource.
pJava	A version of Java for embedded applications.
PVR	Personal Video Recorder.
RAM	Random Access Memory.
RCU	Remote Control Unit.
ROM	Read Only Memory.
SI	Service Information.
STB	Set-Top Box.
TAM	DVB group responsible for "Technical Aspects of the MHP".
URL	Uniform Resource Locator.
VM	Virtual Machine.
Xlet	The name given to DVB-J applications.