The DVB Multimedia Home Platform (MHP) and Related Specifications

JON PIESING

Invited Paper

For some years, digital pay television services have been accompanied by software applications which are downloaded from the broadcast into the set-top boxes in the living room and executed there. The formats and protocols used for these are proprietary. In 2000, after a long and difficult development process, the DVB Project offered to the world an open standard specification to support the execution of such applications called the Multimedia Home Platform (MHP). It enables digital content providers and broadcast equipment suppliers to address MHP receivers, regardless of the manufacturer of the receiver or the developer of the MHP middleware implementation. Since the completion of version 1.0 of the MHP specification in January 2000, derivative specifications have been produced for non-DVB markets. One such derivative is the Open Cable Application Platform (OCAP) produced by CableLabs for cable TV in the United States. This removes completely those MHP features which are simply not applicable in that market. Other features are replaced with a U.S. equivalent. Extensions have been defined by CableLabs for additional requirements.

I. Introduction

Starting in 1997, the scope of the DVB Project was extended to work on a generic, common application programming interface (API) to enable interoperable applications to be downloaded from DVB broadcast networks and executed on receivers from any manufacturer [1]. This common API provides a platform independent interface between applications from different providers and the manufacturer-specific hardware and software implementation. It enables any digital content providers to address all types of terminals ranging from low-end to high-end set-top boxes, integrated digital TV sets, or multimedia PCs. This specification is known as the Multimedia Home Platform (MHP).

The first version of the MHP specification was completed in January 2000. This specification has been updated annually based on feedback from developers of applications, receiver implementations, and conformance tests. The final update to this series of specifications was produced in 2003 and is known as MHP 1.0.3 [2]. An upgrade to the first version of the MHP specification known as MHP 1.1 was produced in June 2001 [3]. Little market interest was shown in this upgrade until 2004 and the first version that is likely to be implemented is MHP 1.1.2, which was completed during 2005.

Since 2001, there has been an interest in using the MHP specification in markets that do not use the underlying DVB broadcast specifications. The first of these being the US digital cable market and their Open Cable Application Platform (OCAP) specification [4]. This is based on a subset of MHP created by the DVB Project called Globally Executable MHP (GEM) [5]. This subset removes a small number of features in MHP which are specifically related to particular DVB broadcast specifications not used outside the DVB world.

II. Scope

A. Example Applications

A number of target applications have been defined for the MHP. Some examples of these include the following:

- electronic program guides for the channels/services provided by a broadcaster;
- information services (superteletext, news tickers, stock tickers);
- enhancements to TV content—sporting applications (including statistics), voting applications, local play-along games;
- e-commerce applications, e-government applications and other applications relying upon secure transactions.

OCAP inherits support for these target applications from MHP and adds support for additional target applications provided by the cable multiple system operator (MSO). These include electronic program guides for all TV channels in a network and applications unrelated to any TV channel in the network such as video-on-demand applications.

Manuscript received February 16, 2005; August 3, 2005.

The author is with Philips Applied Technologies, Redhill RH1 5HA, U.K. (e-mail: jon@prl.research.philips.com).

Digital Object Identifier 10.1109/JPROC.2005.860997

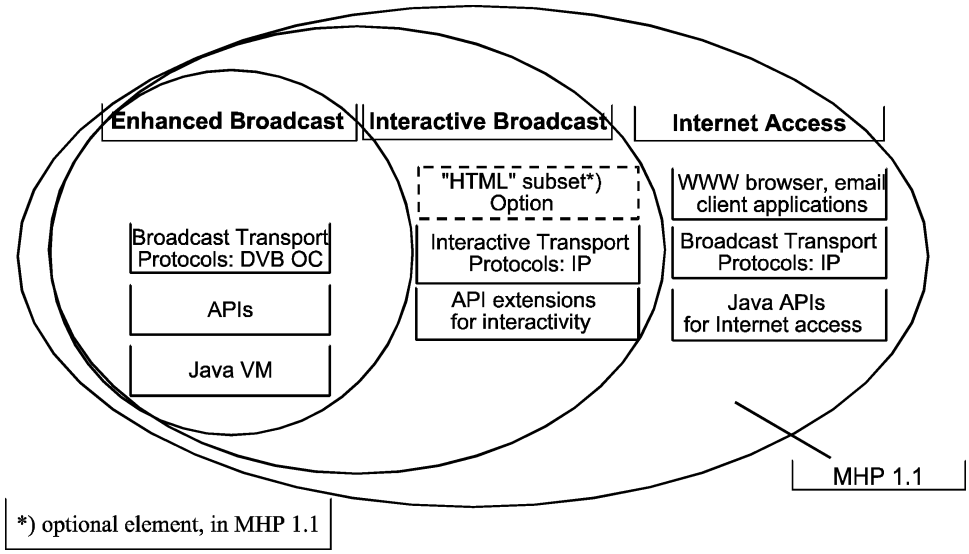

Fig. 1. MHP profiles diagram.

B. Profiles

Functional requirements have been defined for supporting for a wide range of application and receiver types. These have been divided into three profiles: enhanced broadcasting, interactive broadcasting, and Internet access. These are defined as follows.

- Enhanced broadcasting—this profile describes receivers and applications which only provide or need local interactivity within a receiver. In this profile, the only network interface used is a unidirectional broadcast one.
- Interactive broadcasting—this profile describes receivers and applications which provide or need interactive services using a (bidirectional) return channel. This profile is defined to be a superset of the enhanced broadcast profile.
- Internet access—this profile describes receivers and applications which provide or need access to Internet content and services. This is only defined in version 1.1 of the MHP specification.

Fig. 1 shows a summary of these profiles.

III. INFRASTRUCTURE

A. Application Model and Lifecycle

1) MHP 1.0: In MHP 1.0, the basic model for applications is a generalization of that for basic television. The contents of a TV service (i.e., TV channel) have been extended from video, audio, and subtitles to also include MHP applications. In the same way that a generic digital TV receiver automatically starts video and audio when presentation of a DVB service starts, MHP applications are also normally automatically started at that point. When the DVB service being presented changes, the video, audio, and MHP applications associated with the former service are automatically stopped and those associated with the new service started. This concept has subsequently been called "service bound" applications in that the lifecycle of these applications is bound to the lifecycle of a DVB service. There are, however, two key extensions to this simple model.

When the DVB service currently being presented changes, a MHP application running in the former service that also forms part of the new service can continue to run uninterrupted by the change of service. This permits broadcasters who operate multiple services to have applications bound to their set of channels rather than a single channel. Taken a step further, should it be possible to reach agreement that an application forms part of all services in a particular market, such an application would effectively be unbound from any particular service or group of services and would run regardless of the current service.

As well as applications which are automatically started when a service in which they are found is presented, services may also include applications that are not automatically started. A use case for this would be where a service contained multiple MHP applications from which the end user of the MHP receiver can make a choice, e.g., a set of games. An MHP portal application offering this choice to the end user would be one that is started automatically so that it runs as soon as such a service starts to be presented. The individual games within the service would only be started by the portal application and not automatically when the service containing them is presented.

The lifecycle of a single MHP application is similar to that used by Java applets in the Internet but follows an explicit state machine instead of an implicit one. This state machine is shown in Fig. 2.

The diagram also shows the mechanism by which a MHP application is notified of changes in its state, by the MHP implementation calling a method provided by the MHP application. Each state transition is labeled with the name of the method that will be called.

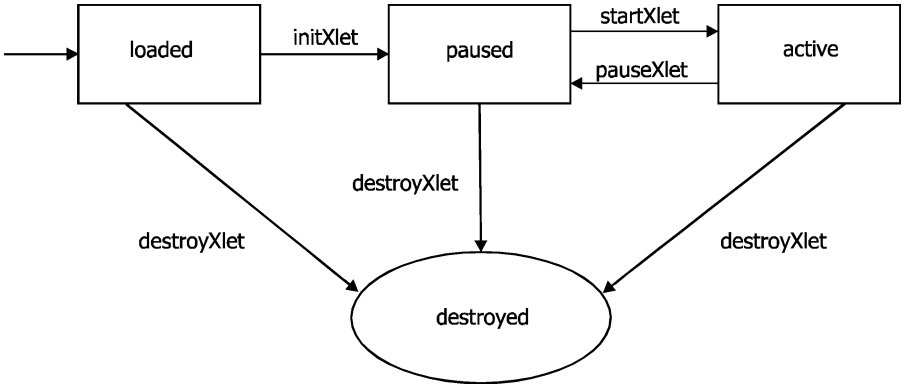

Fig. 2. Xlet state diagram.

2) MHP 1.1 and OCAP: In MHP version 1.1, this model is further generalized to support services that are not DVB services. One form of non-DVB service defined is a stored application service. These contain MHP applications stored in memory in the MHP receiver. As with DVB services, these applications can either automatically start when the service starts being presented or they can rely on another MHP application to start them. Applications are added to and removed from stored application services by other MHP applications using a new Java API added to MHP version 1.1 for this purpose. A second form of non-DVB service in MHP version 1.1 is one signalled via the return channel. The application signaling and the applications themselves are both downloaded over the return channel from a web server using the HTTP protocol.

OCAP also generalizes this model in a similar way to MHP version 1.1. OCAP has explicit support for "unbound" applications, i.e., ones not forming part of any television service. A concept is defined called "abstract services" which group sets of related unbound applications. OCAP abstract services are created and populated with applications either when the OCAP receiver first starts running (by special application signaling) or by another OCAP application using a special OCAP API for this purpose.

3) Design Issues: Support for running multiple MHP applications at the same time was a key issue in the design of the MHP specification. Limiting the specification to supporting a single application at one time would cause problems where interactive content from a number of different sources is required to run simultaneously. This is a very common situation; for example, a digital text application, an electronic program guide and an enhancement to a specific TV program all running together. If these all were required to be part of the same MHP application, it would force the broadcaster or operator to integrate code from a number of different suppliers as a matter of routine practice. However, such a limitation would significantly simplify both implementations and the specification. In the end, DVB decided to require support for this. Five years later, it remains unclear whether this was the right conclusion. If this had not been required, MHP receiver products would have been available on the market significantly earlier than they actually were. The MHP 1.0.2 conformance test suite would have been available significantly earlier than it was. In any case, OCAP requires support for running multiple applications at the same time. Any features left out of MHP due to this simplification would have needed to have been added by CableLabs for OCAP.

Table 1
MHP Application Signaling Descriptors for DVB-J Applications

Descriptor Name	Descriptor Function
application descriptor	Identifies applications, the MHP profiles on which they can run and the transport protocol on which they are carried
application name descriptor	Provides a human-readable name for an application
application icons descriptor	Defines the location of icons associated with an application where these are transmitted.
external application authorisation descriptor	Defines external applications which are allowed to continue to run in this service if already running but which cannot be started in this service.
transport protocol descriptor	Provides additional information about the transport protocol used to carry an application or its data.
DVB-J application descriptor	Defines run-time parameters for the application.
DVB-J application location descriptor	Defines location of DVB-J application, class path and name of first class file to execute.

B. Application Signaling

The MHP specification defines signaling carried in the broadcast network to connect MHP applications with DVB services. The basis of this signaling is an application information table (AIT) carried in an elementary stream identified as such in the PMT. The structure of this table follows the normal DVB-SI [6] structure with two loops containing descriptors. The first loop contains common descriptors that apply to a set of applications. The second loop is a loop of applications with a further loop inside this containing those descriptors that apply to individual applications.

Some of the main descriptors relating to DVB-J applications are shown in Table 1.

The AIT itself contains a control code for each application. One value of this control code allows some applications to be defined as "auto-start" applications. The MHP receiver will automatically start these applications when the service concerned is started. Other values of this control code allow applications to be stopped with varying degrees of force. Another value of the control code allows applications to be signalled which are never started automatically. These can be started by already running applications using the DVB-J application listing & launching API. An MHP receiver is required to monitor the AIT(s) for services which it is presenting in order to look for changes in these control codes and the addition or removal of applications.

OCAP defines some additional signaling for creating and populating the abstract services when an OCAP receiver first starts running. This additional signaling is known as the XAIT. The XAIT syntax is the same as the AIT with some additional descriptors relating to storage of applications and sorting of applications into abstract services. The XAIT is carried in the Open Cable out-of-band channel.

C. Security Model

The MHP security model has a number of goals. One key goal is to protect the end user of the MHP receiver from attacks which could directly or indirectly harm them. The simplest example of this is attacks which could directly cost the end user money, for example, dialing premium rate phone numbers. Less direct examples include privacy attacks, for example, attempting to send confidential information like credit card numbers or a home address to someone who should not be allowed to receive them. Also considered are attacks that would result in unacceptable interruptions to the end user's television watching experience. A second goal is to protect the interests of MHP application providers and broadcasters from attacks by other applications. One example is obtaining access to sensitive information from one application provider held in the MHP receiver. Another example would be an interactive advert changing channel without a commercial agreement with the broadcaster of the original channel.

There are a number of aspects to the MHP security model and its enforcement. The choice of Java itself is a major part of this, since the Java virtual machine has security as part of its core design. The Java byte code verifier plays a major role in enforcing strong typing and preventing applications from executing common attacks such as stack overflows. There are many analyses about the basic security mechanisms of Java, one example being [7].

The MHP specification defines a mechanism for application providers or broadcasters to sign their applications before transmission and for an MHP receiver to authenticate an application using those signatures and supporting certificates. This is shown in Fig. 3.

Only applications that are authenticated by this mechanism can request access to various sensitive features of the MHP receiver, e.g., access to the interaction channel or to persistent storage. This is done by including a special file in the top-level directory of the application which requests

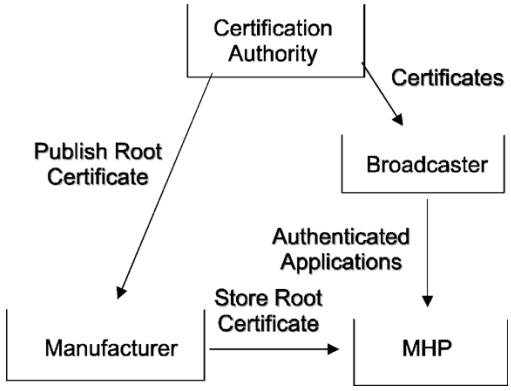

Fig. 3. MHP code signing diagram.

those permissions which the application wants. Applications that are not signed only have access to a well-defined set of MHP receiver features that is only useful for applications providing information to the end user. For example, they are not permitted to save information in persistent memory in the MHP receiver or use the return channel to send information outside the receiver.

One significant challenge in the MHP security specification is how to handle revocation and replacement of certificates in a potentially unidirectional environment. Protocols for certificate revocation like OCSP [8] cannot be used since they require a bidirectional connection. Instead a two part solution has been designed. Revocation of nonroot certificates is addressed using the conventional certificate revocation list (CRL) mechanism. CRLs are transmitted as part of the broadcast signal and the most recent CRL for a certificate authority is cached in the receiver. An attacker with the secret key for a compromised certificate may remove the CRL revoking that certificate from the transmission carrying the compromised application, however, this will be pointless as that CRL will still be available due to being cached. Many people in DVB hope that it will never be necessary to replace the MHP root certificates although a mechanism called root certificate management messages (RCMMs) has been defined to permit this. All MHP terminals have three root certificates and any pair of certificates can be used to replace a third.

For security in the return channel, TLS (the IETF standardized version of SSL) is used [9]. This provides support for secure electronic commerce and services in the same way as it is used in the Internet and enables reuse of existing Internet server systems.

D. Graphics Model

Graphics capabilities and performance in digital TV receivers is very different in philosophy and capabilities from graphics in personal computers. In part this is a matter of history as many ICs used in digital TV receivers started primarily as MPEG-2 video decoders with graphics being very much secondary. This history has a number of practical consequences.

- Graphics pixels in set-top boxes are often the same aspect ratio as video pixels, (i.e., not square). Square

pixels are very much the norm in personal computer graphics systems.
- Graphics and video are often handled as separate planes with graphics being logically in front of the video. Video in personal computers is often either decoded into the graphics plane or is logically in front of the graphics.
- Often the MPEG-2 video decoder is exploited to give better graphics than the actual graphics system is capable of. If the graphics system only has a color resolution of 4 bits per pixel, use of the MPEG-2 video decoder to show a still MPEG i-frame will give a lot better quality natural image than using the graphics system.
- The graphics system is normally interlaced like the video. Applications using the graphics system must be careful to avoid single pixel wide horizontal lines that would flicker badly.

One key aspect of the MHP graphics model is the definition of the composition of graphics and video. The model chosen is a relatively close fit with the ICs used in set-top boxes—graphics is rendered in one or more graphics planes. These are always in front of one or more video planes where video is rendered. Behind the video planes are one or more background planes. Background planes can be a single color or can be used to hold MPEG I-frame stills. Composition within the graphics planes is handled with the normal Porter–Duff rules [10]. Composition within video and background planes always uses the SRC_OVER rule; however, only alpha values of zero and one are defined for these planes. The results of the composition within the graphics planes and within the video and background planes are also composed using SRC_OVER (with the graphics results as the source and the results of the background/video composition as the destination).

In addition to composition, the MHP graphics model addresses coordinate spaces. It defines a single normalized logical coordinate space for the video signal resulting from the composition process described above. This enables the description of a variety of possible relationships between the coordinate spaces of the graphics and video systems. This allows for implementations that scale video and graphics separately, for example so that what appears to be full screen graphics or video does not cover the whole of the area of the video signal resulting from the composition process.

As well as a model for graphics and video, the MHP specification defines some minimum requirements on the implementation of the graphics system. For MHP products in markets where the video system is 625 line, 50 Hz, graphics and video resolutions of 720 $\times$ 576 must be supported. In contrast, in OCAP the resolutions that must be supported are a graphics resolution of 640 $\times$ 480 (which gives square pixels on a 4 : 3 display) and a video resolution of 704 $\times$ 480. For composition between graphics and video, MHP receivers are allowed to make approximations but must support at least 0% transparency (opaque), 100% transparency (completely transparent), and an intermediate value of approximately 30%. At the time of writing, work is in progress within the DVB Project to define graphics resolutions to be supported when high-definition video is being decoded.

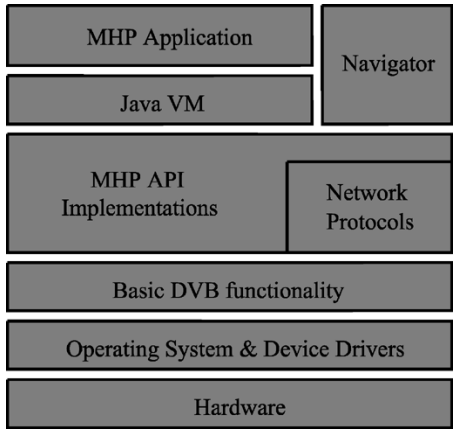

Fig. 4. MHP architecture diagram.

IV. MHP Architecture

Fig. 4 shows a simple view of the architecture of an MHP receiver. Implementations of the various Java APIs included in the MHP specification need an implementation of basic DVB functionality to build upon. Implementations of the network protocols are also needed which may not form part of a basic DVB receiver implementation. On top of these is the Java virtual machine that runs applications. The receiver manufacturer will include a resident application (known as the "Navigator") that provides basic TV functionality. This may fit above the MHP APIs and be implemented in Java or may fit alongside the MHP APIs and be implemented in some other technology.

V. The DVB-J Platform

The core of the DVB-J platform is the Java virtual machine as defined in specifications from Sun Microsystems. This is then extended with a number of APIs to provide access to the various features of a digital TV receiver. In addition to APIs defined by DVB itself, APIs in MHP come from a number of sources—already existing Java specifications (e.g., Java for desktop computers), the Digital Audio Visual Council (DAVIC) [11], the Home Audio Visual Initiative (HAVi) [12], and a Sun Microsystems specification for television called JavaTV [13].

A. Use of Java

The term "Java" is frequently used in a generic sense and can in fact mean a number of different things:
- a computer programming language;
- a standard format for Java code once compiled;
- a "virtual machine" that runs compiled code in this standard format;
- a common set of APIs usable by Java programs.

In the MHP, all four of these are present in varying extents. In particular, the set of APIs has been subsetted to provide an appropriate compromise between code size in an

Table 2
Generic Java Packages Included in MHP

Package Name	Package Description
java.lang	Provides the basis of the Java environment including the object model and multi-threading
java.util	Provides general purpose utility facilities for application developers
java.io	Provides input and output facilities including reading data from files
java.net	Provides access to IP based networks
java.security	Provides the basic framework for the Java security model
java.security.cert	Provides support for public key certificates
java.awt	Provides the basis of the UI framework and event model although extensively subsetted as described in more detail below.
javax.net.ssl	Provides secure connections over IP based networks using SSL/TLS
javax.security.cert	Provides support for public key certificates used in SSL/TLS

MHP receiver and commonality with the wider world of Java outside the DVB Project. The most significant elements of the common set of Java APIs included in MHP are listed in Table 2.

In the MHP 1.0 series of specifications, all the above are based on a Sun specification called "PersonalJava" which in turn is largely based on version 1.1.8 of the Java platform for desktop computers—a version that dates from 1998. This specification has largely been superseded in MHP products today by a successor called "Personal Basis Profile" [14] defined under the Java Community Process (JCP). MHP implementers played a major role in the definition of this in order to optimize the fit with the MHP specification and the needs of MHP implementations. It copies the more significant examples of subsetting done by DVB so there is no risk of MHP implementers being forced by their Java supplier to implement irrelevant features purely to be a complete implementation of a specification. It includes a number of enhancements to simplify integration between an MHP implementation and a Java implementation in the common situation where organizations implementing MHP buy a Java implementation from a specialist supplier.

At the time of writing, the DVB Project is considering requiring support for a version of the Personal Basis Profile specification in a subsequent version of the MHP specification in order that MHP applications may rely on the presence of the additional features it defines.

B. User Interface

The basis of the UI model in MHP is inherited from that of Java in desktop computers—it is built from components that inherit from the java.awt.Component class. These components draw themselves and receive user input events when they have focus. Components exist in containers represented by instances of the java.awt.Container class or subclasses. Positioning of components within containers is done either explicitly using x and y coordinates or algorithmically by specifying a "layout manager" which fits the components inside a container based on a defined algorithm.

Where MHP differs from Java in desktop computers is the resident UI widget set, i.e., a set of classes of components with predefined appearance and behavior whose presence applications can rely on. Java for desktop computers contains two such widget sets, an older (text oriented) widget set forming part of the java.awt package and a newer widget set known as "Swing." Neither of these are included in MHP. The older one that forms part of the java.awt package was felt to be irrelevant in a television environment due to being very text oriented. The omission of this is the largest single subsetting of the PersonalJava specification performed by the DVB Project. The "Swing" widget set was considered, but was felt to require an unreasonable amount of memory in an MHP receiver and DVB did not wish to invest time in fine-grained subsetting of Swing.

The resident UI widget set required by MHP is one developed by the Home Audio Visual Interoperability (HAVi) organization for consumer electronics devices. It builds on the basic framework of components and containers inherited from Java in desktop computers. It provides graphically oriented versions of the classical button, list and slider widget, as well as single and multiline text widgets. It provides applications the ability to query the configuration of the graphics, video, and background planes as defined by the MHP graphics model. It provides support for a variety of mechanisms that are specific to the remote controls normally used in consumer electronics products including the following.

- Navigation between components using up/down/left/right buttons rather than a free moving cursor.
- Extensions to the normal Java user input event mechanism for keys typically found on remote controls but not on computer keyboards, such as fast-forward, fast-rewind, pause, etc.
- Control of which user input events an application is prepared to handle, such as whether an application is interested in receiving the number keys or whether these should be left to the platform to process.

In addition to the features that MHP obtains from the HAVi specification, there are a number of UI features defined by the DVB Project itself. The three most significant of these are the following.

- support for semitransparent colors which enable the video or graphics beneath a graphics object to be partly visible;
- simple formatted text display based a limited set of markup codes;
- support for applications receiving user input events regardless of whether the component with focus is a part of that application or another application.

C. Presentation of Video and Audio

A key feature of any television system that is not normally present in Java for desktop computers is the presen-

tation of video and associated audio. Two mechanisms exist in MHP for this. One mechanism, called "service selection" API causes a television channel (called "services") to be presented as if the end user had directly selected that channel using the remote control. In this mechanism, if the television channel has components in addition to than video and audio, such as an MHP application, those components are also presented to the end user automatically. Using this API, the calling application may be killed if required by the MHP application model and associated application signaling described above.

The second mechanism is used when it is necessary that no MHP application be started and that the calling application continue to run uninterrupted. This mechanism uses an existing (but largely ignored) Java API called the Java Media Framework (JMF) [15]. An MHP application using JMF to present some video or audio does not cause other MHP applications to be started. MHP itself defines some television oriented extensions to JMF (called Controls) as well as including ones defined in the DAVIC and JavaTV specifications. Examples of the features provided via JMF Controls in MHP include the following:

- control of subtitles (e.g., enabling/disabling, language selection);
- scaling and position of video;
- selection of which video and audio streams are to be presented where more than one is present;
- notification of changes in properties of the received video such as aspect ratio.

MHP applications may present video and audio using JMF in two ways. The simplest way is to create a new instance of a JMF Player for the video and audio content to be presented. Alternatively an MHP application running as a result of a service selection operation may obtain an instance of a JMF player for the video/audio of the service being presented via a link from the service selection API to JMF.

D. Digital Television Specific Features

The MHP specification includes APIs for a number of digital television specific features. As with the APIs discussed above, some of these are defined by the DVB Project itself and others are reused from outside sources, specifically the DAVIC and JavaTV specifications. These APIs are the ones most impacted by the changes between MHP and OCAP. Table 3 lists the APIs concerned, where they are defined and indicates how they are handled in OCAP.

The number and nature of SI APIs was a major technical design issue during the creation of the MHP specification. DVB centric interests preferred there to be a DVB SI API and objected to the presence of a protocol independent SI API on the grounds that this was adding unnecessary complexity to the specification by providing two solutions for the same problem. Other interests felt it was necessary to have an SI API which was independent of the protocol and format used to carry the SI information in order to permit applications using SI to be moved between DVB and non-DVB markets. In the end, the latter interests prevailed and the MHP specification includes two SI APIs as described above.

Table 3
Digital Television Specific APIs in MHP and OCAP

API	Description	Source	OCAP
Protocol independent SI API	an abstract API for service information (SI) which hides the nature of the underlying protocol or data format	JavaTV	Supported
DVB SI API	Provides access to information carried in the DVB SI protocol and data formats.	DVB	Omitted
Broadcast transport protocol API	Extends the file API support in java.io with support for various features useful for accessing files carried in broadcast carousels in general and the DSM-CC object carousel in particular.	DVB	Supported
Tuning API	This allows MHP applications to tune a network interface between different transport streams.	DAVIC	Supported
MPEG-2 section filter API	This allows MHP applications to filter data from private sections carried in an MPEG-2 transport stream.	DAVIC	Supported
Conditional Access API	This allows MHP applications to list and purchase entitlements from a conditional access system where this is allowed. It also allows applications which are aware of specific CA systems to establish a private dialogue with the CA system in an implementation independent manner.	DAVIC	Omitted
Content referencing API	This provides a Java encapsulation for a URL format to reference DVB network components such as transport streams, services and elementary streams.	DAVIC & JavaTV	Partly supported

Four years after this discussion, it is still unclear whether the right conclusion was reached. Having two SI APIs certainly makes MHP implementations more complex than would otherwise be the case; however, OCAP demonstrates the value of a core part of the MHP specification that can be used in markets not using basic DVB specifications like DVB-SI. In the U.S. cable environment, SI is only used to provide system information (e.g., about available channels), mostly for the operation of the receiver. Information for end users about the schedules of those channels is normally provided by proprietary formats which are not supported by the protocol independent SI API. Given this reality, it remains unclear whether there will ever be applications that use the protocol independent SI API to move between MHP and OCAP.

E. Application Model and Lifecycle APIs

The most significant MHP API relating to application model and lifecycle is that which provides the initial entry

point to an MHP application when it is first started. This is the Xlet interface provided by the javax.tv.xlet package that forms part of JavaTV. As MHP applications transition through the various states of their lifecycle, the MHP implementation calls the corresponding methods of this interface. This package also includes the XletContext interface that allows an MHP application to request a change in its own state, for example if there is no reason for it to continue running.

The second significant MHP API relating to application model and lifecycle is the application discovery and launching API. This allows one application to obtain a list of available applications based on the application signaling. Information can be obtained about these applications and applications that are not already running can be started. Subject to security restrictions, the lifecycle of running applications can also be controlled.

A more specialised MHP API relating to applications is the Inter-Xlet communication API. This provides a mechanism for communication between two MHP applications based on a remote procedure call mechanism. The MHP specification reuses a concept called remote method invocation (RMI) originally used in Java to provide remote procedure calls between applications running on different machines across a network.

F. Miscellaneous Other APIs

The MHP specification defines a number of APIs in addition to those which are included from external sources. These include the following.

- Persistent storage API—This extends the file access support in the java.io package with support for various features relevant to files held in a persistent storage device such as flash ROM.
- User settings and preferences API—This allows applications to query and manipulate certain standardised user preferences such as user language, country, and default font size.
- Return channel API—This allows applications to set up, modify, and monitor return channel connections with an emphasis on return channels using modems. This includes dialing particular phone numbers and monitoring for service interruption.
- Testing API—This provides support for conformance testing applications to report their results to an automated test environment.
- Smart card reader API—MHP 1.1 includes an API providing access to smart card readers for applications such as government services, health services, and access to supermarket loyalty cards.

One of these APIs relates to a very complex discussion during the development of the specification—persistent storage. Broadcasters and application developers were interested in having guarantees about several issues regarding how persistent storage is managed within an MHP receiver. For example, proposals were made for each broadcaster to have a fixed size quota within the persistent storage. Manufacturers wanted to preserve maximum flexibility for how to specify and implement products. Organizations interested in MHP receiver conformance testing wanted to ensure the specification contained enough guarantees to enable the persistent storage API to be validly conformance tested. The conclusion was to provide very limited guarantees purely for conformance testing purposes. These were the following.

- Minimum size of persistent storage of 4096 bytes.
- Persistent storage must persist between successive runs of an application (e.g., changing to a different channel and back again) but need not persist when power to the MHP receiver is turned off.
- The persistent storage management function provided by MHP implementers shall not delete any contents of persistent storage while the persistent storage used by MHP applications is less than 75% of the minimum size.

During the four years since these initial conclusions were reached, the level of guarantees available to broadcasters and application developers has been improved as part of the MHP specification maintenance process. For example, persistent storage is now required to be nonvolatile. The MHP 1.1.2 specification will increase the minimum size of persistent storage and define more rules for the persistent storage management function provided by MHP implementers.

G. Additional APIs Specific to OCAP

Table 4 summarizes the Java APIs that are defined by OCAP in addition to those from MHP listed above which are included in that specification.

As can be seen, many of these are restricted to MSO applications. These must be requested in the permission request file as normal but can only be granted to specially authenticated applications.

VI. Content Formats

The MHP specification includes a full range of content formats that are required or optionally included in MHP decoders. Some examples of these include PNG, JPEG, MPEG-2 I frames, UTF-8, MPEG-2 video, MPEG-1/2 audio, and DVB subtitles.

For text handing, the MHP specification defines a single resident font that must be present on all MHP receivers in certain specified sizes and shapes. This allows application developers to rely on the presence of this font in their applications. As well as this resident font, service providers will have the option to download their own fonts to provide better control of the look and feel of applications. The MHP specification includes a chapter describing various basic text rendering rules in order to obtain predictable text layout. This chapter is heavily influenced by the UK digital terrestrial TV specification [16] and the experience of the people who developed it.

Version 1.1 of the MHP specification includes an optional declarative content environment called “DVB-HTML.” This is mainly based on a number of specifications from the Worldwide Web Consortium (W3C)—“Modularization of XHTML” [17], Cascading Style Sheets [18], and Document Object Model level 2 [19]. To this are added ECMAScript [20] and integration between HTML and the existing DVB-J

Table 4
Java APIs Defined in the OCAP Specification

Package name	Function
org.ocap.application	Enables MSO applications to control the lifecycle of other applications.
org.ocap.event	Enables MSO applications to manage the distribution of user input events.
org.ocap.hardware	Gives access to low level information about various hardware features of the OCAP terminal
org.ocap.hardware.pod	Gives MSO applications a means to set and get OpenCable CableCARD Resource related parameters.
org.ocap.media	OCAP specific extensions to video & audio APIs
org.ocap.mpeg	Gives access to the Open Cable "Out of Band" channel
org.ocap.net	Small OCAP specific extensions to various MHP APIs
org.ocap.resource	Enables MSO applications to control scarce resource usage by other applications and to resolve resource contention between applications.
org.ocap.service	Provides support for applications unrelated to any TV channel in the network
org.ocap.si	Replacement for the part of the DVB specific SI API which relates to the MPEG-2 PSI
org.ocap.storage	Extends MHP persistent storage API with support for access to multiple persistent storage devices including detachable ones.
org.ocap.system	Enables MSO applications to replace various system functions normally provided by the manufacturer of the digital TV receiver
org.ocap.system.event	Enables MSO applications to receive notification of critical error conditions
org.ocap.test	Extends MHP testing support API
org.ocap.ui.event	Extends HAVi UI API with extra key codes found on remote controls used in US cable products

platform. This environment was designed in 2000 and 2001 for a generation of high-end set-top boxes that were generally never introduced into the market. At the time of writing, there is little evidence of implementation of DVB-HTML (except for those subsets of it which intersect with normal web technologies) and it may be removed in a future update of MHP version 1.1.

VII. Network Protocols

The MHP specification includes mandatory and optional network protocols for both the broadcast and the return channel network interfaces. For the broadcast network interface, the required protocol is the DSM-CC object carousel as defined by MPEG [21] and later modified in the DVB data broadcast specification. The MHP specification extends this with features such as additional descriptors to specify the extent to which particular files may or may not be cached in receivers. Support for multicast IP encapsulated in MPEG-2 is optional. This is shown in Fig. 5.

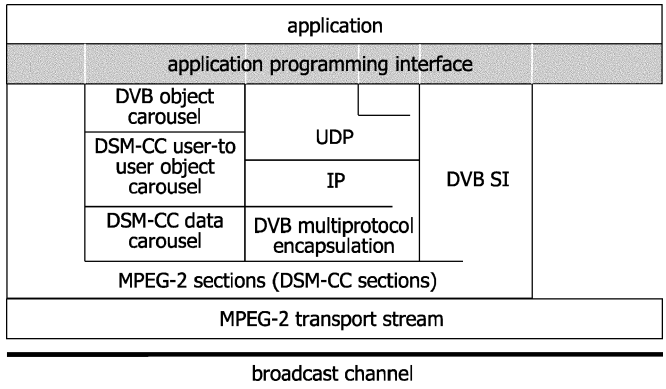

Fig. 5. Broadcast protocols diagram.

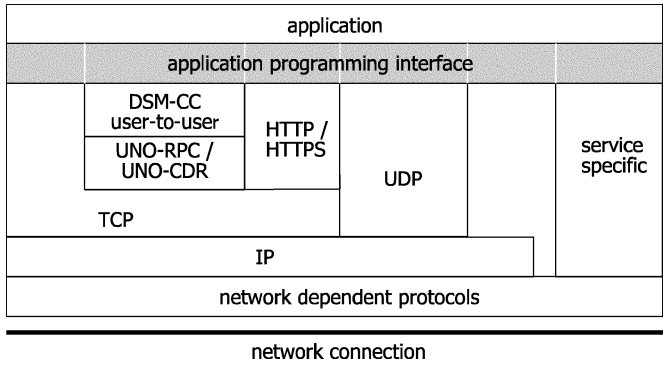

Fig. 6. Return channel protocols diagram.

The extent of support for multicast IP encapsulated in MPEG-2 was a major design issue in the development of the MHP specification. One viewpoint was that this duplicates facilities provided by DSM-CC in general and the DSM-CC object carousel in particular. A second viewpoint was that the world will be moving to IP as the solution for all transport mechanisms and not supporting multicast IP encapsulated in MPEG-2 would be short sighted. In the end, the conclusion was that basic support for multicast IP would be optional and not mandatory. Additionally it was decided not to define or select a protocol equivalent to DSM-CC object carousel for carriage of files via multicast IP encapsulated in MPEG-2. Four years after this discussion, this would appear to have been the correct conclusion. Multicast IP has not become pervasive. Only in October 2004 did the IETF publish a specification for file transfer on top of multicast IP [22] and this is only an experimental RFC at the time of writing. The extent to which this will be adopted remains to be seen.

For the return channel network interface, the mandatory protocols are the obvious IP, TCP, UDP, and DNS. HTTP 1.1 is optional in version 1.0 of the MHP specification but mandatory in version 1.1. DSM-CC carried over IP using IIOP is optional. This is shown in Fig. 6.

VIII. Experience in Maintenance of the MHP Specification

In the four years between the completion of the first version of the MHP 1.0 specification in January 2000 and

the release of the second errata to MHP version 1.0.3, just under 2500 changes were implemented to the specification as identified by the change tracking mechanism. For an 800-page document, this works out as an average of about three changes per page. While a significant number of these are editorial, there are obviously many technical changes as well. Some examples of the changes include the following.

- Simplifying some parts of the specification. As a concrete example, there are two mechanisms for governing access to files in persistent storage; java.io.FilePermission and the MHP defined "credentials" mechanism. During the maintenance process, the credentials mechanism was simplified and aligned with the FilePermission mechanism to enable some reuse of code in implementations.
- Reversing some decisions made in the original specification development. One concrete example of this is requiring persistent storage to be nonvolatile as mentioned above. Another example concerns text rendering. As mentioned above, the MHP text rendering rules are derived from those in the UK digital terrestrial TV specification. That specification includes a simplifying assumption that all text is rendered as if a 14 : 9 display was being used. The original MHP 1.0 specification removed this assumption and supports 4 : 3 and 16 : 9 displays explicitly. The pixel aspect ratio corresponding to the actual display is used in the conversion of outline fonts to pixels. Hence, a piece of text will occupy differing numbers of pixels on the screen depending on the actual display used. During the maintenance process, this decision has been partly reversed following feedback from application developers who wanted the facility for text to occupy the same number of pixels regardless of the actual display used.
- The policies and details for issuing certificates for signing MHP applications were not finalized until some time after the specification was completed [23]. A number of changes were made to the MHP specification resulting from this finalization. The minimum number of CRLs which MHP implementations are required to support was increased from five to eight. A DVB-SI network identifier was embedded in certificates to limit the scope of certificates to signing applications in particular networks and not all MHP networks in general.
- The largest single contributor of specification issues was the consortium developing the MHP conformance tests—the MHP Test Consortium. Of the 2500 implemented changes referred to above, more than 1000 result from issues originally reported by this consortium. The most common significant issue is a description of behavior if a condition happens but with the omission of any description of behavior if that condition does not happen.
- The original smart card reader API in MHP 1.1 turned out to be a dead specification. It does not seem to have been implemented in the real world. As far as is known, no conformance tests have been developed for it. Since MHP 1.1 has not yet been deployed, this was replaced in MHP 1.1.2 with an API developed for Java in mobile phones which is being implemented and has conformance tests available.

Many of these changes are a consequence of the DVB Project practice of releasing specifications as a "1.0" version without waiting for proof of implementation or for development of conformance tests (where appropriate). Other specification development organizations have a process where specifications persist in a semifinal condition until some defined criteria for availability of implementations and/or conformance tests has been achieved. DVB does not have this process or the resulting criteria.

IX. Digital Video Recording Extension for MHP and OCAP

A key benefit of digital TV to many consumers is digital video recording. This offers many benefits to consumers compared to analog VCR's, particularly in terms of ease of use. It is desirable for MHP and OCAP applications to be able to use a number of features provided by digital video recorders. Examples of these include the following;

- allowing electronic program guides to offer end users the ability to schedule the recording of a program;
- allowing MHP applications associated with a recorded program to offer random access to specific points in that recording;
- allowing an MHP application to assemble a "virtual channel" from a set of recordings and to sequence the playback of these in an order defined by the application.

The DVB Project and CableLabs have cooperated on developing a joint specification for the features of digital video recording common to both environments. Both parties also have their own specifications that extend this common specification to address environment specific requirements. The DVB specific specification includes a Java API for TVAnytime [24] and the integration of that with the features provided by the joint specification. This includes the delivery of TV-Anytime metadata and content referencing information both via a unidirectional broadcast channel and via a bidirectional TCP/IP channel.

Probably the largest issue in the development and implementation of these specifications is the extent of the support for the recording and playback of interactive applications—particularly dynamic data such as changing files in an object carousel, DSM-CC stream events, MPEG-2 private sections (accessed via the MHP section filter API), and changes in the MHP application signaling. Ensuring that an application which relies on these behaves identically when played back from a recording device to when played from a live broadcast is a significant issue. This is particularly true for implementations of the digital video recording feature that extract information from a MPEG-2 transport stream and store it in some other (possibly optimized) format. At the time of writing, the MHP digital video recording specification does not require this recording and playback. As a consequence of this, it is only required to record applications that do not rely on dynamic data.

Another key issue is the provision of a timeline to identify specific points within a recording in order to support random access. MHP 1.0 includes the normal play time (NPT) from DSM-CC; however, this is known to be vulnerable to disruption in many digital TV distribution networks. Existing deployed network equipment that regenerates the MPEG system time clock (STC) is unlikely to be aware of NPT and hence will not make the necessary corresponding modification to STC values inside NPT reference descriptors. This would result in random access to the wrong points in a recording, or even the target point for random access appearing to be outside the recording. To address this problem, a new timeline mechanism has been defined [25] where the messages are carried in PES packets. PES packets include the MPEG defined time in a standard location in the packet header. Equipment regenerating the MPEG time will likely be aware of the need to update this header and capable of doing so. Hence, the probability of the timeline passing through the distribution network without disruption is substantially increased.

ACKNOWLEDGMENT

The MHP specifications represent the work of too large a number of individuals to practically list. The author would like to thank all of these people.

REFERENCES

[1] U. Reimers, "DVB—The family of international standards for digital video broadcasting," *Proc. IEEE* , vol. 94, no. 1, pp. 173–182.
[2] *Digital video broadcasting (DVB); Multimedia Home Platform (MHP) specification 1.0.3*, ETSI Doc. No. ES 201 812 V1.1.1, 2003.
[3] *Digital video broadcasting (DVB); Multimedia Home Platform (MHP) specification 1.1*, ETSI Doc. No. TS 102 812 V1.1.1, 2001.
[4] *OpenCable Application Platform Specification*, OCAP 1.0 profile, CableLabs, 2004.
[5] *Digital video broadcasting (DVB); Globally Executable MHP (GEM)*, ETSI Doc. No. TS 102 819 V1.2.1, 2004.
[6] *Digital video broadcasting (DVB); Specification for service information (SI) in DVB systems*, ETSI Doc. No. EN 300 468 V1.6.1, 2004.
[7] G. McGraw and E. Felten, *Securing Java: Getting Down to Business With Mobile Code*, 2nd ed. New York: Wiley, 1999.
[8] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "RFC 2560: X.509 Internet public key infrastructure online certificate status protocol—OCSP," IETF, 1999.
[9] T. Dierks and C. Allen, "RFC 2246: The TLS Protocol version 1.0," IETF, 1999.
[10] T. Porter and T. Duff, "Compositing digital images," in *Proc. SIGGRAPH 84* pp. 253–259.
[11] "DAVIC 1.4.1 Specification Part 9: Information representation," Digital Audio-Visual Council, 1999.
[12] HAVi, Inc., "Specification of the home audio/video interoperability (HAVi) architecture," 2001.
[13] Sun Microsystems, "JavaTV API 1.0," 2000.
[14] Sun Microsystems, "Personal basis profile specification (JSR-217), ver. 1.1," 2005.
[15] Sun Microsystems, "Java media framework," 1999.
[16] Digital Television Group, "Digital terrestrial television MHEG-5 specification," 2003.
[17] M. Altheim, F. Boumphrey, S. Dooley, S. McCarron, S. Schnitzenbaumer, and T. Wugofski, "Modularization of XHTML," W3C, 2001.
[18] Bos, W. Lie, Lilley, and Jacobs, "Cascading Style Sheets, Level 2," W3C, 1998.
[19] Le Hors, Le Hégaret, Wood, Nicol, Robie, Champion, and Byrne, "Document object model (DOM) level 2 core specification," W3C, 2000.
[20] European Computer Manufacturers Association (ECMA), "ECMAScript language specification," ECMA Doc. No. ECMA-262, 1998.
[21] ISO/IEC, "Generic coding of moving pictures and associated audio information; Part 6: Extensions for digital storage media command and control," 1998.
[22] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, "RFC 3926: FLUTE—File delivery over unidirectional transport," IETF, 2004.
[23] DVB Services Sarl, "DVB MHP application signing and authentication certificates certificate policy version 1.0," 2004.
[24] TV-Anytime Forum, "Broadcast and on-line services: Search, select and rightful use of content on personal storage systems ("TV-Anytime Phase 1")," ETSI Doc. No. TS 102 822, 2003.
[25] DVB, "Digital video broadcasting (DVB); Specification for the carriage of synchronised auxiliary data in DVB transport streams," ETSI Doc. No. TS 102 823, 2005.

Jon Piesing received the B.A. degree in physics from Keble College, Oxford, U.K.

He is currently a Senior Technical Consultant at Philips Applied Technologies, Redhill, U.K. From 1996 to 2003 he managed the project in Philips Corporate Research responsible for interactive digital television and Java, which was responsible for much of the technical work in Philips on the MHP. From 1998 to 2001 this project developed the first prototype MHP receiver and he was responsible for sales of these prototypes to application developers, broadcasters, and operators worldwide. He has been participating in the DVB Project's work on the MHP since 1997 and has been chair of the technical group responsible for the MHP specification since 2001. He also represents Philips in a number of other DVB committees, in the MHP Test Consortium, in CableLabs activities connected to OCAP and in the Java Community Process where he sits in the executive committee for J2ME. Before interactive digital TV, he worked for Philips on the "CD-i" interactive compact disk system.