MIDDLEWARE: A KEY COMPONENT FOR BROADBAND

Dr. Ken Morse, Azita Manson

PowerTV, Inc.

Abstract

There are many discussions in the digital interactive cable television community regarding middleware. Debate rages over requirements, technology choices and even terminology and definition. This paper presents an overview of the design decisions and engineering challenges faced during the development of the pTV Software Platform by PowerTV, Inc. While the paper's focus is primarily client-based, it also addresses the approach adopted on the server side of the equation.

INTRODUCTION

This paper presents an overview of the engineering challenges of the pTV Software Platform from inception to delivery and subsequent evolution through seven years of development by PowerTV, Inc. The hardware target for this software platform is interactive Digital Home Communications Terminals (DHCTs) of the Explorer 2000 class and above. Alternative software approaches have been followed by vendors addressing the DCT 2000 class DHCTs and is not in the scope of this paper.

Before discussing the software architecture it is important to review the target DHCT capabilities and network infrastructure in which the DHCT resides.

THE DHCT

The Explorer® 2000 DHCT provides support for digital services and traditional analog services delivered through a hybrid fiber/coax (HFC) network. The first version of the 2000 shipped in 1998 and was equipped with a 54MHz RISC microprocessor, 4MB of FLASH memory for system software, 2MB of memory for MPEG decompression and graphics, and 8MB of DRAM for system and application use.

The DHCT is equipped with 64 and 256 ITU J.83 [1] Annex B QAM demodulation support and a service tuner enabling both analog and MPEG 2 digital channels to be tuned and displayed. A DAVIC 1.1 [2] compatible out-of-band (OOB) system, operating at 1.544 Mbps is included to enable instantaneous, IP-based, "real-time" two-way communications between the DHCT and the headend.

Digital services are secured using the PowerKEY® conditional access system provided through an on-board security microprocessor.

Local devices may be attached to the DHCT through the Universal Serial Bus (USB) interface or an optional Ethernet

10BaseT interface. Together these support the connection of devices ranging from printers and digital cameras to personal computers.

DIGITAL BROADBAND DELIVERY SYSTEM

The DHCT terminates the digital broadband delivery system (DBDS) that combines video, audio, and data content from a variety of sources and distributes them to the subscribers' home. Given the scope of this paper, it is important only to understand the network connectivity as viewed by the DHCT.

Figure 1 illustrates the communications channels between the DBDS and the DHCT.

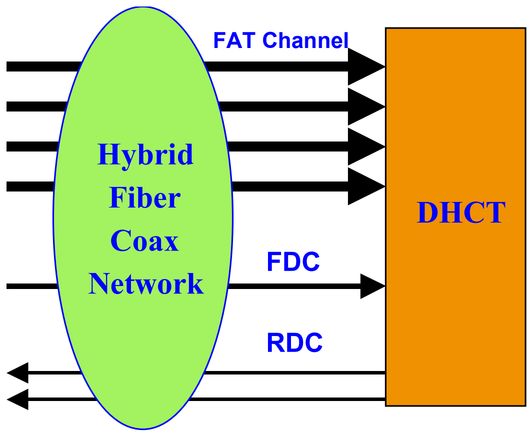

Figure 1 – Network Connectivity

- Forward Application Transport (FAT) channels. The DHCT can select any FAT channel by tuning to it.
- Forward Data Channel (FDC). The DHCT can always receive the FDC, even while tuned to analog services.
- Reverse Data Channels (RDC). The DHCT can only transmit in one RDC. However, more than one RDC may be defined per node for capacity reasons.

With the baseline understanding of the DHCT and the network connectivity a discussion of the software platform is presented.

SOFTWARE PLATFORM OVERVIEW

Once the DHCT platform and network were defined, it was possible to address the software architecture in the DHCT. The architecture follows a layered approach with clearly defined interfaces and responsibilities. Figure 2 provides a block diagram of the software components.

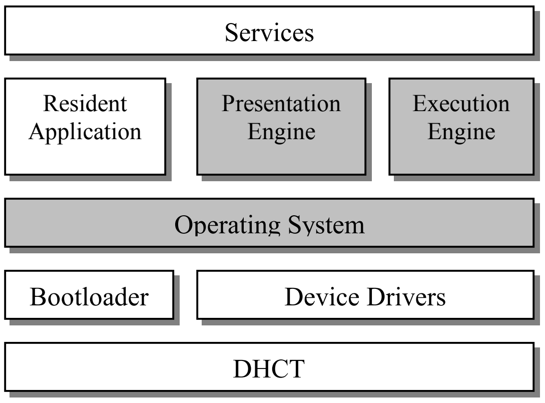

Figure 2 – Software Components

The items in gray make up the pTV Software Platform presented in this paper and are discussed in the following sections.

Bootloader

The bootloader is the gatekeeper of the DHCT and ensures the integrity of the system software stored in the

FLASH memory. It listens for communications from the headend to ascertain whether software updates are available for the particular hardware revision of the DHCT and also automatically recovers the correct version of software in the rare case of corruption in the DHCT image.

Device Drivers

Each instance of a DHCT (from the same or different manufacturers) may contain different components such as processor, memory, and custom semiconductors. To ensure that the software platform can operate across a range of DHCTs, a Hardware Abstraction Layer (HAL) is provided. This HAL is an Application Programming Interface (API) specification to which DHCT manufacturers author their device drivers to ensure compatibility with the software platform. It is important to note that the device driver implementation is the first layer of the software stack. Design and implementation decisions made at this level ripple through the entire software stack. Key implementations guidelines have to be provided at the device driver layer to address the efficient use of memory, interrupts, and thread prioritization. For example, since the DHCT deals with high bandwidth data flows, it is important to minimize the number of times a piece of data is copied as it traverses the software stack. In addition, information, such as diagnostics that may be required at higher levels in the software stack, is often generated within the device drivers and must be exported.

OPERATING SYSTEM COMPONENTS.

The Operating System (OS) provides the foundation on which all other software components are built. For this reason, it has three clear non-negotiable requirements.

Make the DHCT work

The DHCT must be robust and reliable as it is delivering core television services to the subscriber. Hence, the operating system must be designed to deal with the high data processing requirements in conjunction with the constrained memory environment while maintaining a high quality of service. In addition, the local interfaces on the box must be supported, such as USB and Ethernet.

Make the network work

The DHCT is the termination point in the network and as such the OS must support all the necessary protocols for interfacing with the rest of the network. This includes, but is not limited to, conditional access, system information table management, session management, carousel delivery and network management.

Provide a software platform

Since all other software components are built on top of the OS, it must provide an extensive API that provides access to all the functionality contained within the DHCT. For multiple software components to concurrently operate against the OS in a constrained DHCT, it must provide a comprehensive resource management environment enabling policy decisions to be made by higher-level components.

Core Services

The core OS services are used by all software components in the system including all other OS components. A real-time, pre-emptive and multi-tasking kernel is required to ensure the deterministic operation of the DHCT when dealing with high bandwidth data flows. The kernel is optimized to manage low context switch times and low latency response to interrupts. In some DHCT designs there can be upwards of two hundred different interrupt sources – clearly identifying the complex design within a DHCT. In addition, several kernel modifications were made by the team based on the requirements of higher-level software components, such as Java to ensure efficient operation of the overall system.

Many DHCTs, including Explorer 2000, do not include a Memory Management Unit (MMU). This makes the task of ensuring the integrity and availability of memory a larger problem. The OS addresses this through a range of memory management processes applied to different types of memory in the DHCT – application, graphics, MPEG – in addition to algorithms to minimize memory fragmentation. As part of the resource management architecture, different software components may be asked to yield memory resources when memory availability becomes critical. This implementation maximizes the number of concurrent software tasks that may be operating on the DHCT.

It is important to note that the services in the OS must be multi-thread safe and re-entrant as necessary.

Multimedia Services

Another often-misused term is Multimedia and in the scope of the support from the OS this incorporates video, audio and graphics display. Video services are provided at the MPEG transport level in the OS, other video format support may be built in higher-level software components. Similarly, MPEG and Dolby AC-3 audio streams are managed as part of the MPEG transport stream. Additional support is provided for PCM and ADPCM audio sources in the OS.

Graphics functionality is split into three layers. basic graphics management provides access to the frame buffer(s) of the DHCT and supports multiple color depths basic graphics primitives in addition to pixel operations. These constructs are then used by the Window Manager to interface with the resource management architecture to allow multiple software components to share the television display concurrently. This includes combining graphics and the video plane of the DHCT.

The Window Manager design is the result of rationalizing the requirements presented by a diverse set of needs including higher-level software components such as Java and the Abstract Windowing Toolkit (AWT).

Finally, the OS provides a Widget Manager component enabling consistent use of Widgets throughout the software platform. This aids consistency of look and feel in the higher-level software components. Once again requirements on the Widget Manager included the Java AWT and widgets used in HTML pages such as buttons, edit fields, and forms.

Network Services

The DHCT is a network device and the OS must carry a set of core networking protocols. Some of these are generic IP-based protocols; others are specific to the cable television environment.

In the system previously described a DAVIC out-of-band implementation is employed. The OS carries the DAVIC signaling implementation and is responsible for managing the connectivity to the cable headend. This is a critical area affecting Quality of Service (QoS) in the DHCT. Extreme failures, such as power outages, require managed sign-on to the network to avoid congestion and overload. An indication of the current operational state of the network is required by all higher-level software components. The OS works behind the scenes to manage the integrity of the network connections and resources for those components.

Certain services such as Video On-Demand (VOD) require sessions to be put in place across the network and maintained – and on occasion, re-negotiated. Once again, the OS manages these connections for the higher-level components.

IP-based services are supported in the environment and an optimized TCP/IP network stack is in place to manage the IP packet flow. Modifications have been made to support multiple interfaces to the stack. This enables in-band IP support over the FAT channel and local routing of packets to other physical interfaces in the box.

A core component of any integrated design is network management. The OS provides an extensive SNMP agent supporting a range of diagnostic information that may be queried over the network. It is also extensible and allows higher-level software components to dynamically install new Management Information Bases (MIBs) to report on items as diverse as QoS to subscriber patterns. The use of SNMP in the system greatly reduces the impact of introducing new services and components into an existing system.

Given the inherent broadcast nature of a cable television system, the digital broadband delivery system provides a Broadcast File System (BFS) for delivering data and objects to the set-top. This is a broadcast, carousel-based design and the OS maintains the available file list and retrieves objects as requested by higher-level components.

Television Services

Television services are the final high-level category in the feature set supported by the OS. These include the ability to acquire System Information (SI) tables from the network and tune to A/V programs. Once again, extensive QoS support is implemented in the case of SI outage on the network. The SI database is held in the DHCT and updated as new tables are delivered over the network.

Conditional Access (CA) is not limited to traditional television services and may be applied to any service available on the DHCT. CA may also be used for signing and authenticating software components delivered over the network to the DHCT.

Since traditional SI is limited to video/audio programming selections the

OS architecture was extended to support the dynamic selection of applications from remote sources, typically the BFS. On the client side, the Service Access Manager (SAM) provides the functions necessary to load, register, and launch an application on the DHCT. The API provides a method for loading an application into the DHCT memory, activating the application, and managing the application after it is loaded

RESIDENT APPLICATION

The Resident Application (RA) is built on top of the OS and provides the core digital television services to the subscriber. These include:

- Navigator
- Interactive Program Guide
- Impulse Pay-Per-View
- General Settings
- Emergency Alert System (EAS)
- Diagnostics
- Digital Music Service
- VCR Commander Support
- Virtual Channels

The RA utilizes the services provided by the OS to implement policy decisions for the set-top regarding service activation, suspension, and removal. The OS itself does not enforce policy; it provides the services through which another party may implement policy.

Resident Applications are available from Pioneer Digital Technologies and Scientific-Atlanta for the platform described in this paper.

THIRD PARTY DEVELOPMENT

Given the consistent and open API of the OS, a range of third party developers have developed and deployed applications and services against the core platform. Each deployed application must go through a certification phase to ensure it is interoperable within the end-end system. Over one hundred companies are working against the OS today creating new and innovative services on the DHCT.

ALTERNATIVE SERVICE AUTHORING APPROACHES

While C-based services have been developed and deployed by a wide range of companies against the OS, it was clear early on that alternative authoring environments would have a place in the DHCT over time. Any authoring environment will have advantages and disadvantages. In deciding on alternative authoring approaches it is necessary to understand the benefits and deficits of C-based services. These are outlined below:

+ Performance
+ Footprint (for a given service)
+ Network utilization (self-contained)
+ Robustness
- Complexity of authoring

- Not cross-platform (processor-specific)

- Certification process

Alternative authoring approaches need to address some of the shortcomings of C-based services while not sacrificing some of the key advantages provided by the native approach.

As part of an investigation into alternative service authoring techniques in 1996, an existing public domain web browser was ported to the operating system outlined earlier. At this time the technologies involved were simple by today's standards; HTML 2.0 [8] for layout, GIF87 image format, and HTTP 1.0 for object requests. To validate the applicability of the browser environment an existing service, Video On-Demand, was modified, along with the browser, and tested on a real-world cable system. The modifications consisted of extending HTTP to provide control of the video server trick modes and adding several tags to HTML 2.0 [8] to enable the overlay of graphics and placement of MPEG video. It was clear that such a content authored approach with extensions for object request and control could form the basis of a future environment.

At about the same time, Sun Microsystems released the first version of its Java programming language in the form of a development kit, or JDK. By working closely with Sun, we were able to gain access to the source code of the Java environment and port it against the current software platform. This activity and the resulting work is detailed in the section, Execution Engine.

PRESENTATION ENGINE

Given the initial prototyping work on an HTML 2.0 [8] based browser, outlined above, a number of issues would have to be addressed if an HTML-leveraged solution was going to operate in the constraints of a home communications terminal. The approach decided upon was to design and develop a software component for presenting content driven services on-screen and communicating with known servers using modified data representations on HTTP. For this development, the term presentation engine (PE) was defined for the resulting product.

The requirements defined for the presentation engine are presented in the following paragraphs along with a description of the engineering challenges addressed in bringing the product to market.

Integrated into environment

The success of a platform is based on how well integrated the design is. With this in mind, the presentation engine must leverage the resource management architecture provided by the core platform and follow other good-citizen style guidelines. This also includes integrating with the existing Resident Application infrastructure.

Standards-based, where applicable

The goal of the development was to leverage existing standards as needed and extend to support platform-specific capabilities that were not addressed by the existing standard.

Meet footprint targets of DHCTs

Memory constraints in a DHCT are tight, typically 8MB to contain network buffers, platform state and IPG data with the remainder available for services and the use of the presentation engine state and the service content. The presentation engine itself would live in FLASH memory as part of the core platform.

Maximise leverage of existing content

It was clear that in the near-term there would be a lot of available content authored for the larger market, i.e., the Internet, and that there would be distinct advantages if this content could be leveraged into the platform in the early stages.

Enable platform capabilities

The DHCT provides a set of integrated features unlike other environments, such as Personal Computers. The presentation engine should provide access to these features.

Authoring tool availability

One of the complaints leveled against development in a standard procedural programming language is the complexity of the development. With this in mind it was important to ensure that the necessary tool chains were in place for a successful presentation engine product.

Robust and reliable operation

This is a television product and must be operational 24 hours a day, 7 days a week, 52 weeks a year - this cannot be over-communicated. The television experience is solid and dependable. The introduction of new services and technologies must not affect this in any way. It was clear from the start of development that ensuring this requirement is met would be the toughest by far. Existing web-leveraged environments are either unable to meet this criteria, e.g. personal computer, or do not address the memory and processing environment of a DHCT.

PRESENTATION ENGINE DESIGN

From the previously stated set of requirements the design of a PE was undertaken. This section identifies the design choices made.

Approach

The decision was made early on to focus on what is meant by an engine. Even though the majority of content presentation solutions are based around some form of browser, the design team decided to architect the PE along the lines of the OS; i.e., a set of services that can be manipulated by higher-level components through well-defined interfaces. This approach ensures that resource management issues (such as multiple use instances) for the PE are addressed in the core design.

Content Support

Based on previous experience with the HTML 2.0 [8] based browser and the explosive growth of the World Wide Web in 1996 it was decided to implement along the lines of an HTML-based solution. This ensured that the availability of authoring tools would not be a problem. In addition the free-form nature of HTML page definition lends itself to manipulation to fit the television display.

Given the decision to utilize HTML the next question arose of what version of the specification to implement. This was decided based on the memory profiles projected in the DHCT arena over the initial product lifespan. Based on assumptions of 8MB DRAM platforms, HTML 3.2 [8] was chosen. HTML 3.2 [8] includes feature sets not normally supported on constrained memory devices such as frames and forms but given the expected utilization of television services (such as t-commerce and walled gardens) it was decided to support this feature set. The use of frames required focus on core navigation issues such as movement between different frames.

HTML pages can include a range of different objects beyond the HTML page description. These fall into two categories; MIME types, and scripting.

With a focus on leveraging existing tools and content within the constraints of the DHCT, the MIME types supported include:

- GIF
- JPEG
- AIFF
- WAV
- Text

These MIME types are supportable across the range of DHCTs deployed. Additional MIME types such as Macromedia Flash are supportable on more capable DHCTs.

HTML provides a very static environment and given the non-static nature of the television environment it was decided that interactivity must be provided through the addition of scripting. ECMAScript was chosen due to its prevalence in the developer community and historical use on the World Wide Web. Traditional implementations of ECMAScript assumed an environment with extensive memory and memory management in place. Modifying ECMAScript to operate within the constraints of the DHCT environment provided to be an extensive task to ensure that no memory leaks were present and that ECMAScripts could run robustly.

Content Translation

HTML content may be authored for arbitrary page sizes; it does not have to conform to a given constraint since existing desktop browsers support scrolling in two dimensions. It was deemed that asking a subscriber to deal with two-dimensional scrolling in a television environment would be unacceptable. To solve this problem, all pages would need to be scaled to fit the available horizontal resolution of the window on-screen. This operation potentially involves extensive image scaling and table re-calculation, depending on the content. The DHCT is not equipped with the necessary processor bandwidth to support such tasks adequately and so the solution was to create a proxy/transcoder component that resides in the network. This proxy/transcoder intercepts DHCT requests, acquires the page and associated assets and scales them accordingly, passing the results to the DHCT (and to a local cache to maximize performance of subsequent requests).

Another key design point was to ensure that the proxy/transcoder did not have to scale based on the number of attached clients. The proxy/transcoder could be instead placed at the Internet Point-Of-Presence and scale to the content pipe, a much simpler task, resulting in tremendous cost savings to the operator.

In addition, the proxy/transcoder was designed to operate in an offline mode enabling “walled gardens” to be automatically created from existing content.

Performance

After robustness, most of the development activity centered around performance. This was classified in two primary areas:

- On-screen display
- Network performance

On-screen drawing performance was the key metric since this is what the subscriber sees and determines whether the product is adopted or not. Standard techniques such as double buffering provided difficult at high graphics resolution given the limited memory available for graphics buffers. This resulted in alternative drawing algorithms being implemented. In two-way systems, the Proxy/Transcoder could be utilized to ensure content delivered to the DHCT was scaled appropriately, if necessary, resulting in a performance improvement in perceived drawing speed.

A caching subsystem was implemented to handle pages and assets (in both compressed and decompressed forms). The caching system increased the availability of content and therefore reduced perceived subscriber access time. Through HTML extensions it is also possible to assign a caching priority to pages and assets. In conjunction, a scheme for pre-fetching was introduced. These two features, when used in combination, are useful in “walled garden” environments to ensure fast access to frequently used assets and offers a degree of deterministic content control in the client.

Finally, the PE can also receive content delivered through other mechanisms than HTTP. These include the BFS, MPEG private sections and the Vertical Blanking Interval (VBI). Such broadcast asset delivery mechanisms can be used to minimize un-necessary upstream traffic on the network.

Secure Connections

Given the expected use of t-commerce on the platform a secure communication mechanism was required. Once again, the standard of the Web, Secure Sockets Layer (SSL) was adopted. SSL is supported in two versions (2.0 and 3.0) with a range of different strength ciphers. The Presentation Engine design enables selective loading of ciphers and protocols to address the memory constraints of current generation DHCTs.

While the proxy/transcoder can provide translation facilities for non-secure pages it cannot translate content contained on secure web pages (using SSL). Two approaches were considered to address this. One approach is to introduce a proxy that terminates the SSL connection at the headend. In-the-clear page content can then be transcoded and then a separate proxy may be utilized to form a secure connection with the DHCT. This obviously breaks the end-end security from the merchant to the subscriber and introduces legal issues that make such a solution undesirable. The solution employed is to provide client-side re-layout of the pages in the case of secure connections. This has the downside of potentially affecting performance but does ensure the integrity of the end-end security.

ATVEF

The Advanced Television Enhancement Forum (ATVEF) specification provides an environment for program synchronous content delivery. The content may either be included with the video/audio content or stored separately on a remote server. ATVEF support is designed into the pTV Software Platform and requires no specific support from the PE beyond supporting the *tv:* URL. As intended, the PE merely acts as a service to present the necessary ATVEF content on-screen.

Status

The PE is part of the pTV Software Platform and has been shipped in over seven million DHCTs to date. The memory footprint is configurable from 700KB up to 1.4MB.

EXECUTION ENGINE

The Presentation Engine is designed to display web content documents to the users. This model yet useful and simple to implement cannot accommodate for all the needs of an interactive application, so there is need for a procedural programming language, which is easy and fast to implement. The industry has chosen the Personal Java for the Execution Engine as it is defined by the DVB-MHP and OCAP.

PowerTV and Sun Microsystem have collaborated to create a PJava implementation that is optimized to run in an interactive two-way cable network.

PJava is designed for a constrained network connected devices. The small memory footprint of PowerTV Operating System combined with Personal Java enables this design to run on the low-cost, two-way capable DHCTs. Write Once, Run Anywhere™ design of PJava allows for the fast and low cost development of applications. With PJava design, a prototype may be built and debugged on computer and deployed on the DHCTs.

pTV Software Platform is comprised of the Java Virtual Machine and set of standard class libraries defined by the PJava, version 1.1.

By using the common underlying resource management, pTV Software Platform has been able to achieve an environment, which allows for both presentation engine and execution engine to coexist harmoniously.

Some of the enhancements made to the pTV Software Platform, while implementing the JVM, was to increase the support for additional threads in the kernel. Memory management has also been modified to accommodate for the JVM memory requirements. The existing Window Manager has been modified to fully support the AWT user interface model.

In future, pTV Software Platform shall implement support for the JavaTV APIs. This effort is minimal due to the full support of all core TV functionality in the existing pTV Software Platform.

FUTURES

New Functionality

With the current status presented the question is "what of the future?" As ever, the key drivers in the interactive cable arena are the trends in DHCT design and deployment mechanisms. New functionality is being introduced into DHCTs including PVR, Home Gateway, etc. These features will drive a necessary extension of the middleware interfaces on the DHCT.

OpenCable

Perhaps a larger influence is the Federal Communications Committee (FCC) mandate for cable operators to give subscribers the opportunity to acquire their DHCT through retail outlets. CableLabs® has created a sub-committee to address this requirement. In addition to defining the hardware platform requirements and conditional access interfaces, OpenCable® has a working group defining the application platform for the retail DHCTs.

OPENCABLE APPLICATION PLATFORM (OCAP)

The OCAP specification defines the Application Programming Interface for an OpenCable™ retail device. The Middleware API, defined by OCAP, is independent of the underlying hardware. Furthermore, the OCAP middleware layer provides an abstraction layer to the set-top box's operating system that manages the underlying hardware's resources.

The OCAP specification is based on the DVB-MHP specification. OCAP specification describes in details the deviation from DVB-MHP, where applicable, in addition to the new components definition. OCAP specification has been designed to focus on the needs of the North America's cable market.

The OCAP specification is intended to create an open environment for application developers to write applications, which are platform independent.

DVB-MHP

DVB-MHP's architecture is based on the DVB-J platform, which includes a Java Virtual Machine as defined by Sun Microsystem. DVB-MHP is comprised of three layers, applications, system software or middleware, and the resources on the set-top box.

OCAP Basic Architecture

The OCAP architecture resembles closely to the architecture illustrated in Figure 2. The OCAP architecture is comprised of two major components: the Presentation Engine and the Execution Engine.

What is OCAP Presentation Engine?

The Presentation Engine supports a set of declarative applications. The Presentation Engine is required to support the content formats for markup (HTML, XHTML) [8], cascading styling sheet (CSS) [8], plus Application Programming Interfaces (DOM) [8].

What is OCAP Execution Engine?

The Execution Engine is based on the Java Virtual Machine and it is comprised of set of APIs defined by PersonalJava Application Environment (PJAE) [3], Java TV API Specification [4], Java Media Framework (JMF) Specification [5], Home Audio/Video Interoperability (HAVi) Architecture Specification [6], and Digital Audio-Visual Council (DAVIC) Specification Part 9 [7].

OCAP Minimum Platform Requirements

The minimal set of required device resolutions that OCAP terminals shall support is 640x480 (square pixels) for Background, video, and graphics. These resolutions shall be supported for display aspect ratios of 4:3 and 16:9.

The minimum processor capability has been defined as 200 Mega Instructions Per Second (MIPs).

The memory requirement is 64 M Bytes of RAM and 16M Bytes of ROM. The NVRAM shall provide at least 16K Bits maximum storage.

SUMMARY

In closing, it is clear that middleware plays an active role in the increasing adoption of interactive television systems. While it is important to have consistent interfaces to which services can be authored it is also clear that an optimized implementation of the software stack providing these interfaces is necessary. This ensures performance, maintainability, extensibility and time to market – all key requirements to future success of the industry. While different middleware standards have been implemented over the past few years there is hope for a common interface through activities in the retail space such as the OpenCable initiative. However, for current non-retail systems, the challenges of building an optimized middleware stack remain and integrated platforms present the best path forward.

ACKNOWLEDGEMENTS

The authors would like to thank the entire team at PowerTV. Without them and their developments this paper would not exist.

REFERENCES

[1] ITU Recommendation J.83 (04/97) - Digital multi-programme systems for television, sound and data services for cable distribution; Annex B

[2] Digital Audio Visual Council™ (DAVIC), version 1.1

[3] Sun Microsystems, PersonalJava Application Environment Specification Version 1.2a: http://java.sun.com/products/personaljava/

[4] Java TV API Specification; http://java.sun.com/products/javatv/

[5] Java Media Framework, Version 1.0, May 11, 1998; Sun Microsystems Java Media Framework Specification http://java.sun.com/products/java-media/jmf/1.0/apidocs/packages.html

[6] Home Audio/Video Interoperability (HAVi) Architecture Specification; http://www.havi.org

[7] Digital Audio Visual Council™ (DAVIC), version 1.4.1 Part 9.

[8] www.w3.org