The Challenge of a Standard Software API

Mark Eyer
Sony Electronics

Cable Multiple-System Operators (MSOs) would like to supply not only digital audio, video, and data services, but also software applications that can run on customer-owned equipment. With this goal in mind, CableLabs® issued a Request for Proposal in 1999 to help identify and standardize a software Application Program Interface (API) for OpenCable compliant retail boxes. This paper explores the challenges involved in this effort, and identifies some of the pitfalls and obstacles that must be overcome. These include issues of platform independence, the cost and complexity of the platform, the challenge to support an evolving digital world, and the need on the part of consumer electronics manufacturers to differentiate their products in the marketplace. Suggestions for resolution of some of these dilemmas are presented for consideration.

THE MSO'S VISION

The eventual availability in the retail market of digital cable-compatible consumer devices offers the cable MSO a number of significant benefits. Whenever a customer buys a retail cable-ready device, the operator's capital expense is reduced. Due to competitive market pressures, the retail devices will be able to offer the latest technologies, including faster CPU speeds, ever-speedier graphics, and interfaces to the newest audio/video peripherals. And happier customers can result: many are more content without the need for the bulk and clutter of the set-top box, as set-top box functions are integrated with the digital television.

This picture is quite clear for services including standard- and high-definition audio/video offered on subscription and impulse-pay-per-view (IPPV) basis. The standardization of the interface to the removable security module, the network (physical cable) interface, and system and service information (and agreements to deliver it) has enabled consumer electronics manufacturers to start designing digital cable-compatible devices for retail sale, starting with digital TVs (DTV).

But what about other services, such as Electronic Program Guides (EPGs), video on demand (VOD), voice over IP (VOIP), or streaming audio and video in formats other than MPEG-2 or Dolby Digital? And what about services not yet conceived? Set-top boxes supplied by the MSO can be built to offer advanced services. How can a device available for retail sale be enabled to do so?

A simplistic view of the world, from the point of view of the cable operator, is that the primary purpose of any cable-ready device to be available at retail should be to generate revenue for that operator. To that end, the retail device would be 100% controlled by the cable operator in terms of everything that is presented for viewing—its "look and feel."

An MSO's dream, therefore, might be that a DTV or other retail cable-ready device, after being brought home and installed by the consumer, would be downloaded with code supplied by the local cable operator. At that point, any access the consumer would attempt to make of any services offered on the cable would be managed through a navigation application supplied by the cable operator.

If a special offer or preview were available, the navigator could make sure to present that information to the user. If new services were offered, the navigator could be set up to notify the user of their existence, and to guide the user towards their access. As an additional source of revenue, advertising or links to commercial sites could be included in the navigator.

Services such as VOD could be offered, because the navigator could support whatever proprietary form of access and control was required by that operator's plant and equipment. New forms of services could be offered when they became available, even if the details of presentation and

decoding are yet unknown. That's because an updated navigator could be provided when the details of the new service are worked out.

An example of such a new service is a data broadcasting service. Multicast data synchronized to video is offered today, but the standardized techniques for transport and the content coding formats are not yet totally settled. When industry acceptance is widespread and the particular cable operator implements the new standard, the navigator can be upgraded to allow all subscribers to have access to the new data enhancement.

The EPG can adapt the presentation based on what services are authorized for viewing in this particular device. For example, if the user has not subscribed to *MovieMax*, the navigator can direct that user to the *MovieMax* preview channel, can notify the user of special sign-up offers, or allow him or her to sign up online (self provisioning).

With a downloaded navigator, the cable operator has direct control over the look and feel of the EPG. They can organize the guide in such a way that the services with the highest profit margin (IPPV perhaps) are given prominence. They can put effort into human factors design to help ensure that the consumer's experience is productive and pleasurable.

FACING REALITY

What's wrong with this dream? A troubling aspect of this code download scenario is the implied notion that every retail device, regardless of manufacturer, make, or model, would behave in exactly same way. In the following sections we first explore this product differentiation problem, and then discuss further difficulties, including:

- Problems with the overall philosophy of cable- operator-supplied downloaded code
- Technical challenges, such as reliability and the difficulty of porting the API
- Challenges related to the magnitude and complexity of the problem

The paper goes on to suggest some resolutions to these problems, and describe example products we would like to be able to manufacture once the software download system design and specifications are complete. We then suggest a way forward in the near-term, bridging between technologies available today and that which we will develop and refine in the next several years.

PRODUCT DIFFERENTIATION

From the point of view of the consumer electronics manufacturer, the idea that every model of every manufacturer's product ultimately runs the same cable operator-supplied code causes real marketing problems.

Competition in the marketplace

Consider a high-end product from manufacturer A compared with a high-end product from manufacturer B: when compared side-by-side on the sales floor, both products will appear to be *identical* once downloaded with the local cable operator's application suite.

Low, middle, and high-end

Commonly in consumer electronics marketing, a manufacturer offers low-end, middle, and high-end products. The middle of the road product offers some features not found on the low-end model, and the high-end product offers bells and whistles not found on the level below.

Perhaps the low-end product doesn't support software download at all. But let's say the middle and high-end products do support the OpenCable Middleware Solution. Once downloaded with the cable operator's application, when either of these boxes accesses a cable service, the user experience is the same (aside from factors like CRT display size).

In this world, a manufacturer can no longer differentiate one product from another based on a software-related feature.

PHILOSOPHICAL PROBLEMS

Native applications

A native application is one written in or compiled to the machine code of the retail device's

CPU. A cable operator may want to maintain control over native applications, for example by downloading a "Master Application" capable of authenticating them, launching them, and determining their privileges and resource usage. Is this practical, possible, or even reasonable? We think not.

Given that each model of each manufacturer's product would typically have a different set of native code, it is entirely unclear how the cable operator's downloaded application could be afforded such control.

Consider that a cable-ready device offered for retail sale must have *some* level of functionality even before an operator-supplied code download occurs. For example, it will likely provide access to analog and free services on cable, assuming that the user has a basic cable service. It would provide some form of user setup and/or diagnostic functions even without a basic cable service.

This native application cannot and should not be under the control of the Middleware Solution or the cable operator's downloaded application.

In fact, the native application needs to take priority over anything that might be downloaded, for example to allow it flush memory and re-initialize the unit in case of trouble. Or, in case the user moves it to a new city and/or a new cable system.

It is not only impractical but also unwise to say that an application provided by the cable operator should control the native applications in a retail device.

The native application on the right in Figure 1 accesses OS functions in the device directly. As shown, a companion *resident* application is also present. The resident app is written in a platform independent way by the manufacturer, and takes advantage of the middleware layer implementation.

As shown, a cable-operator supplied Application Suite is present, including a "master application." We feel that this master application should be the "master" of the elements of the application suite (EPG, VOD, web browser, as shown), but it cannot and should not be involved with the control of the resident or native applications.

Extensibility

Let's say an API is eventually agreed upon, and some number of compliant platforms are fielded. In a year or two, as history tells us, typical CPU speeds will be doubled, memory prices will be halved, and graphics capabilities will increase by a large factor. In two years it may be cost effective to include video hard disks in most devices.

If the MSO or cable operator does not create a new application suite tailored to the 2nd generation platform, the power of any new available hardware cannot be fully exploited. The size and capabilities of the application are limited by the least common denominator platform.

If the solution is to provide a new application suite to be run on the next-generation boxes, where does this progression stop? The process of defining and standardizing new platforms and API extensions is never-ending as the operator's configuration control and management problem becomes exponentially more difficult.

How many new downloadable applications might any MSO develop over a five-year period? We think the answer is something like one or two at most, given the enormous complexity of the task. A software release for a large cable plant must be rigorously and thoroughly lab-tested before large-scale deployment. Testing such an application would be an unprecedented challenge because of the large (and growing) number of target platforms upon which the application must be validated.

Control over *all* cable-delivered services

A cable operator may want to use the downloaded application to control the look and feel of *all* cable-delivered services, including the DOCSIS cable modem. Clearly, the operator grants or denies access via the cable modem to the Internet. Access to cable modem services is based on whether or not the consumer has paid for a subscription to the cable modem service.

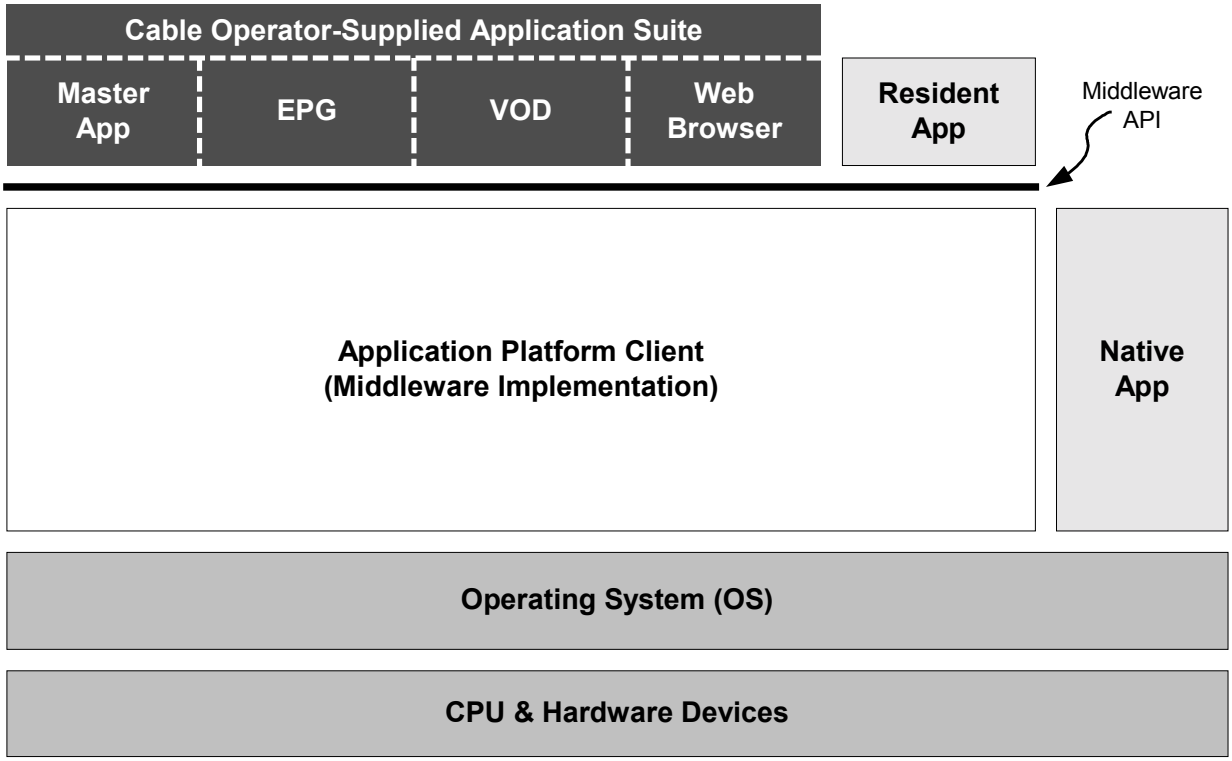

Figure 1. Software Architecture of a Representative OpenCable Retail Box

But if a cable modem service has been paid for, what control over the "look and feel" of it can a cable operator hope to impose? It will be the authors of HTML content on a website, for example, that will determine the appearance (and look and feel) of presentations based on that content.

Perhaps the cable modem in the set-top or DTV is connected by Ethernet to a PC. How could the cable operator have any control or impact of the on-screen look and feel on the PC screen?

One application for *all* OpenCable devices

A cable MSO may want to create a single application for deployment across the full range of OpenCable host devices available at retail. That suggests that the same application that runs in a retail DTV also runs in a D-VHS recorder, retail set-top box, or cable-connected Personal Video Recorder (PVR). Upon reflection, it will be clear that the one-application-for-all-devices goal is unreasonable.

For example, the PVR may be a low-cost device that happens to have a cable tuner but no user interface of its own. It can act as a slave storage peripheral and program source for other devices on a home audio/video bus, but it wouldn't have a use for (let alone the resources to support) a cable-operator-supplied navigator application.

Even if the PVR could accept an application, a PVR is a PVR and not a DTV. The application must account for the functional aspects of the type of host platform it runs on.

Any cable-operator supplied application *must* take into account aspects of the platform upon which it is expected to run. An application downloaded to a retail set-top box must be optimized for a set-top box type of product, just as an application for a DTV must be adapted to the DTV.

Control over *all* resources

A cable MSO may wish to use a downloaded application to control *all* resources available on the host device. We see a difficulty here.

Let's say a new model of consumer device offers wireless connectivity to other devices. Can the cable operator's downloaded application have access to the resources offered by the wireless port? Such access will not be practical until the

Middleware Solution itself is extended to support wireless access. How can the application take advantage of wireless access anyway, given that the capabilities of this particular host device are unknown to it?

Furthermore, as an unending succession of new types of resources are invented and popularized, will the Middleware Solution be continually extended to include each new one? This seems impractical as a road forward.

TECHNICAL CHALLENGES

100% reliability

The reliability of the cable-ready device is very important. Clearly, we cannot allow tomorrow's digital television or set-top box to move anywhere near today's notion of a PC, where program crashes and the need to re-boot the machine are frequent and commonplace.

On the other hand, 100% reliable crash-proof code is impractical with today's technology, especially given the size and complexity of the code involved, and the environment in which it is intended to run.

Even with the Java programming language and its elimination of pointers, some types of programming errors can cause available memory to diminish to zero over time. The program may not "crash" but it can become unusable, non-responsive to RCU keys, and require some form of re-booting or reset.

Another form of crash can occur as two separate software processes, or "threads," both try to access the same data structure in memory. Unintended results can occur, or in some cases a "deadly embrace" can result where both processes are blocked from further operation.

Also, of course, software testing isn't foolproof. By some reports for example, the first release of Windows 2000 included 63,000 known "issues." The count didn't include those bugs not yet discovered of course.

It is possible to write a very stable application that will function reliably for long periods of time, given a stable and well-understood environment in which it can execute. The challenge we face in the OpenCable approach is that the environment is not always well known. There will be many platforms and many implementations.

Lack of standards for code download and authentication

Compared to the issues above, this one is rather easy. Security functions in an OpenCable compatible device are handled within the POD module. Standards are needed to allow a Host device to make use of API calls to the POD module to determine if a particular piece of executable code has come from a reliable source, with no tampering or corruption. Some early proposals have been floated to some standards committees addressing this need.

Porting a complex API definition

When a manufacturer wishes to support the OpenCable Middleware Solution in a product, it may be necessary to port it to the Operating System chosen for that product. If the OS is a common one, it is possible that an implementation of the Middleware Solution standard for that OS can be purchased. Even so, there will always be a significant amount of work involved—the device-level support aspects must be home-grown in almost all cases. These include the aspects of tuning, demodulation, MPEG-2 decoding, SI section filtering and parsing, graphics and display control, audio control, front panel and RCU interface, signal switching and routing, and communications protocol stacks.

Cost and return on investment to support standard API

Once the Middleware Solution is standardized, manufacturers of cable-compatible devices will be asked to provide support for it. The cost to add support for this API will be nontrivial. Many megabytes of RAM and ROM will be required. A significant portion of the cost to develop such a product will be involved with testing the implementation.

The manufacturer is likely to ask: "where is the return on my investment to include support for this API? Will my customer recognize the value of it to the extent that it increases the cost of the product?"

After all, it is the cable operator that will reap the monetary benefits. For manufacturers to embrace the standard API concept, it appears that some form of business arrangement will need to be made with cable operators as an incentive.

THE CERTIFICATION CHALLENGE

Many compliance and interoperability questions are raised by the notion of a standard software API to be used by retail consumer devices. How will a manufacturer certify that a certain implementation of the Middleware Solution is fully compliant? Given the large number of manufacturers involved, and the need to individually test each different product from each manufacturer, the number of products needing testing in a given year is very large.

While CableLabs has undertaken to certify standalone DOCSIS modem implementations, that organization does not appear to be capable of supporting, on its own, certification of all these new consumer devices.

The Middleware Solution specification document is likely to be extremely large and complex. As an example, consider the candidate specifications promoted by ATSC by the DTV Application Software Environment (DASE) group and by Sun Microsystems in the JavaTV effort. The 922-page DASE API specification (version 1.08.01) includes over 350 Java packages, classes, and interfaces.

One might ask, "What's new here—haven't there been other similarly challenging compliance problems?" I think the answer is "no." Let's look for some examples.

In the realm of Microsoft-Intel PC platforms, certainly there are a large number of manufacturers and implementations, but the compliance problems are much simpler. Most importantly, all platforms use the same type of CPU, so all code is "native." Compliance testing has been achieved and many vendors have created clean-room implementations of the Basic I/O System (BIOS). If the BIOS is compliant, any Operating System or application riding on it will work. Unlike the OpenCable Middleware Solution, which will be *very* large and complex, the IBM BIOS is small and very well defined.

In the world of satellite TV, different vendors can license the technology needed to build satellite IRDs compatible with a certain operator's signal. Typically, the number of manufacturers is quite small (three or so). Each manufacturer is free to use a proprietary hardware platform, APIs and native applications. Nevertheless, the compliance testing conducted by a typical operator is quite complex and time consuming.

The essential difference in the case of retail cable-compatible devices is that 1) the API is very large and complex, and 2) there are likely to be a very large number of equipment manufacturers, each with multiple products.

SUMMARY OF ISSUES

In summary, the following issues and difficulties with the Middleware Solution concept proposed for OpenCable have been raised here:

- No provision for or acknowledgment for the need for product differentiation is made
- Authenticating and controlling execution of native applications is impractical and unnecessary
- There is a clear need for a well-understood evolutionary path, or of an approach to realizing extensibility in a practical way
- The concept of "one application for all retail devices" is impractical
- "Crash-proof" code is practically impossible
- The cost of implementation and testing the API is an issue to the CE manufacturer
- Certification and compliance testing will be an unprecedented challenge

PROPOSALS FOR ACHIEVING OUR GOALS

The following sections outline proposals for an expanded Middleware Solution concept that we think can work for consumer electronics

manufacturers as well as cable operators and MSOs. The most fundamental proposed change is that the Middleware Solution must support resident and manufacturer-supplied applications alongside the cable- operator-supplied downloaded code.

We first argue why this change is necessary, and then describe some of the new system requirements that result.

PRESERVATION OF COMPETITIVE URGE

The competitive urge must be preserved. If a Middleware Solution were designed such that product differentiation was not supported, the only challenge for a manufacturer would be to make the most cost-effective implementation. Enhancing the product line with more and better features could not occur. Innovation would be stifled.

Clearly, this scenario is unacceptable. Product differentiation ultimately benefits both the consumer electronics industry and the cable MSO. If we agree that product differentiation must be possible, then the following conclusions result:

- The Middleware Solution must support unrestricted execution of native and resident applications, under control of the consumer (not the cable operator). This was discussed above.
- The Middleware Solution must support native or manufacturer-specific extensions and "hooks."
- The Middleware Solution must allow any specific implementation to fully take advantage of (on its own) hardware and interface features that might be present.
- Communications resources such as the cable modem must be freely available to the native and resident applications (given the proper basic authorization for cable modem service, of course).

Support for API "hooks"

To support certain desirable product features, the resident or native application needs to be aware of certain user actions. In some cases in fact, the resident app needs to be able to take control of the user interface.

The manufacturer's native application needs to be able to "register" with the Middleware Solution so that it can be notified about certain events. For example, it may want to be notified whenever events of the following types occur:

- Any tuning-related API is called
- The user targets a program scheduled for future viewing (for example to set a timer)
- A timer activates or expires
- An Emergency Alert notification is received
- Access is made to web content related to the television program being viewed

The API must support such access to these events, as well as any others deemed useful by the implementation.

Support for hardware features

A Middleware Solution may be implemented on various devices, each with a different set of included hardware and interface features. These might include 3D graphics acceleration, special content decoding formats (MPEG-1, DirecTV transport, etc.), support for specific peripheral devices on the IEEE-1394 serial bus, camera and video input ports, game controller ports, USB, storage peripherals, DVD player/recorders, personal video recording capability, format conversion (camcorder to MPEG, etc.), wireless communications ports and protocols, or countless others.

If any of these hardware-related features are present in the cable-compatible device, the manufacturer-supplied resident and native applications must be afforded access to them. Furthermore, to the extent possible via API hooks and extensions, the native code should be able to integrate them with the cable operator-supplied application suite.

For this philosophy to work, some fundamental aspects of the Middleware Solution API may need to change.

EXAMPLE PRODUCTS

The following sections explore some possible cable-compatible products to show how the concepts we have presented apply.

Example: Video game console in the box

This product combines a state-of-the-art video game machine with a cable-compatible digital television. A DOCSIS cable modem is included so that interactive games may be played via the Internet.

By way of review, here is a brief list of the issues:

- If the video game platform uses DOCSIS (a cable service), and the Middleware Solution must rule the look-and-feel, how can that work on top of the video game? Clearly each individual game played on the game console has its own look and feel.
- How could the Middleware Solution authenticate each game before it is played? Clearly that's not possible nor is it needed.
- To save cost, eliminate redundancy, and provide proper sharing of resources, the video game application may want to take advantage of API calls within the Middleware Solution. Such sharing of the API appears not to be allowed in the current middleware concept.
- The cable operator's downloaded application is supposed to be offered the ability to interface to any native extensions (in this case, the video game hardware and software support library). This can't work because these extensions are very likely to be proprietary. Even if they were open, the "one application" likely couldn't take advantage of resources available only in a small fraction of implementations.

Example: High-end retail cable-ready box

Consider a high-end cable-compatible device to be offered for retail sale. The manufacturer of this device is aggressively leading-edge and wishes to offer the following product features:

1. Support for a sophisticated file system for personal video recording on a built-in hard disk
2. Support capture of video clips via a CCD camera port or images from a digital camera for attachment to e-mail
3. Support an enhanced navigator function, combining web-hosted databases with data derived from expressed user preferences and observed viewing habits
4. Support a marketing product tie-in service, allowing the user easy access to products and services associated with broadcast television programs
5. Support a feature where the device acts as a home audio/video master control center, capable of routing signals between a/v equipment throughout the home
6. Support a "home gateway" function including a private personal website, through which the homeowner may login from anywhere in the world to access guide data, check on and set recording timers, check home security and access other equipment controlled by the device.

The standard Middleware Solution may or many not include APIs that would support this set of features (it's not likely). Even if it did, while a cable-operator supplied application suite *could* possibly use the standard APIs to include these features, it is extremely improbable.

Even if the operator-supplied application suite did do a few of these features, the user should be able to choose which he or she wishes to use.

CO-EXISTING APPLICATIONS

In an environment that includes a cable-operator supplied application suite alongside the manufacturer's native code, some new challenges arise.

Memory management issues

The cable- operator-supplied application will require or request certain memory resources (RAM, Flash-ROM, even hard disk file space), as will the

resident or native application. The resident and downloaded applications may both request, through API or OS calls, "all the remaining memory."

Once a system of cooperative sharing of memory resources is worked out, both these competing entities can peacefully co-exist.

Another problem relates to management of the persistent storage. A Flash memory file system is an example of a persistent storage system. The file system is often used by applications to store preferences, passwords, "cookie" type data, viewing history, forms data, or any other data that has some kind of lasting value.

The problem arises as the file system inevitably overflows, and no more space is available to create new files or to expand the size of existing ones. Some kind of file space reclamation process must be run. That process must decide which files are good candidates for permanent deletion.

With personal computers, we don't allow such a process to make decisions about what files to delete. We do it manually. Writing a file space reclamation routine will be a challenge, because unexpected results can easily occur when files expected by a certain application turn up missing.

Security policy issues

In a typical Middleware Solution, a *security policy* is an inherent part of the API. Through an enforced security policy, certain applications can be granted or denied access to specific API calls. A Java applet, for example, is not allowed to perform file I/O.

As discussed above, we do not feel it is workable to allow the cable operator's *master application* to dictate and establish all security policies in effect within a cable-compatible device. So the question is open: who will allow whom to do what? Agreements on management of security policies will need to be worked out and documented.

Resource sharing issues

In the scenario we are discussing now, an application suite provided by the cable operator is present alongside a manufacturer supplied resident or native application. If we wish to allow both of these to have actively executing threads, contention for resources can result.

For example, both applications may wish to make use of the tuning API. If one has control of tuning, the other will need to deal gracefully with the unavailability of that resource. This simple approach may result in some undesirable behavior, where even a high-importance request can go ungranted. Such problems suggest that a priority scheme should be adopted for resource management.

Clearly, a new level of complexity is introduced. Such priority mechanisms have been discussed in some groups, but the problem has so far not been brought to a clean solution.

PROPOSAL FOR STEPWISE EVOLUTION

Given the scope and magnitude of the unresolved issues, we feel that we are perhaps two to three years away from a workable solution to general-purpose code download. So, in light of these difficulties we present here a proposal for a practical way forward that can benefit both the cable operator and retail device manufacturer.

We start with the definition of a cable-compatible device that can be built today, using currently published SCTE and OpenCable standards. The stepwise evolution therefore starts exactly where we are today.

Level 0: the "Watch TV" box

The features of the level 0 box include the following:

- Performs the basic "watch TV" function.
- Conforms to the OCI-N network interface, tunes/demodulates 64- and 256-QAM, etc.
- Supports all video formats defined in EIA-818 and OpenCable, either by down-converting to NTSC, or passing compressed HD video via 1394 to DTV.
- Navigation based on SI/EPG tables as standardized by SCTE DVS.

- Adheres to OpenCable requirements for diagnostics, audio and video performance.
- Hosts OpenCable POD module for access to premium services, pay-per-view.
- Provides copy protection on all analog or digital outputs.
- If an upstream transmitter is provided, supports impulse pay-per-view (IPPV).

We expect level 0 DTVs to reach the market sometime in late 2001. No further standards work is needed aside from finalizing some of the details of the POD interface and POD copy protection (this work is underway now).

Level 1: the "Web Browser" box

We now define a cable-compatible device with features and capabilities going beyond those offered by the Level 0 box. The Level 1 cable-compatible device provides all the features of Level 0, plus it:

1. Supports a DOCSIS cable modem
2. Supports a TCP/IP protocol stack plus HTTP, SNMP
3. Is capable of interpreting and displaying HTML-based content (for example, HTML 4.0, ECMA Script, DOM1, CSS1)
4. Supports HTML extensions for a standardized TV-based URI scheme so that HTML pages can include links to TV channels (per ATVEF, for example)
5. Supports a scheme whereby the retail box knows the "home page" offered by the cable operator. By this means, the cable operator can operate a *portal* and offer links to services and promotional offerings.
6. Supports access to world-wide web as a pay service (otherwise, box can only access the operator's intranet).
7. For set-top devices: supports OpenCable HDNI specification for pass-through of compressed HD video to DTV.

Except for one detail (#5), we have standards available to allow us to formally define the level 1 device today.

We need to make sure all the standards are in place to allow a cable-ready device purchased at retail to be brought home by the consumer, plugged into the cable, and become operational without the need for the cable operator to roll a truck.

For this discussion we can assume that before even buying the new device, a subscription to basic or premium cable service has already been established. When the new box is first plugged in, it will be able to access unscrambled analog or digital services.

The customer then calls the cable operator to indicate a desire for cable modem and premium movie services, and requests that they send an access card. A POD module arrives in a day or two. The customer plugs it in.

The presence of the POD module triggers an initialization sequence in which the device communicates with the cable headend, registers itself in the network, and is provisioned for access to cable modem and the requested premium channels.

At this time, with a suitable new protocol we need to standardize, the POD module can give the Host a URL to be used to access the cable operator's local home page. In Denver, for an example cable operator named *XYZ Cable*, this URL might be http://stb.xyz-cable.den.com. Now, an RCU button labeled **HOME** could trigger opening a browser window using this URL, and *XYZ Cable*'s portal would pop up.

From the *XYZ Cable* home page, the user can access the following types of functions, at the discretion of this cable operator:

- Links to information on the channel lineup
- Links to Electronic Program Guide data presented in HTML format. A search function can be supported; listings can be by time or genre, etc.
- Hyperlinks to account information pertinent to this particular customer
- Hyperlinks to allow self provisioning (for example, to allow one to sign up for new premium services, special events, etc.)
- Search functions related to program offerings or FAQs

- Telephone numbers to call for service problems or account information
- Hyperlinks to local businesses. The portal can of course include advertising banners.

It should be clear that the level 1 device is a very powerful and flexible platform, offering the cable operator a "look and feel" presence in the retail box and a rich set of operator-supplied features. Importantly, this can be done with existing standards (with the small exception noted).

Level 2: Web browser plus Java

The Level 2 device provides all the features of Level 1 plus:

- Adds a Java Virtual Machine to support Java applets associated with web pages.
- A specified set of Java class libraries is resident to support the desired applet capabilities (for example, it may be a subset of Personal Java and JavaTV).
- Minimum RAM requirements are specified for downloaded applets, cache.
- Minimum graphics resolution and performance requirements are specified.
- Support for a prescribed set of content and mime types is specified (graphics formats, audio formats, streaming audio/video formats and plug-ins, etc.)

The level 2 device can respond to web pages enhanced to include applet-based applications. The level 2 retail device can be built with today's technology, with a few exceptions. At this writing the Java components are not yet finalized. Completion of these is expected sometime in 2000.

Also, to be effective, some form of memory management would have to be standardized so that important applets would be cached in RAM for quick retrieval when needed. It wouldn't be practical to have to wait for an applet download whenever a channel was changed, for example.

With the addition of this basic level of Java support, the cable operator can now offer fully-featured Electronic Programming Guides with improved presentation and better look and feel, Video-On-Demand, enhanced broadcasting, enhanced e-commerce applications, and networked games.

Since only one application is assumed to be executing at any given time (the one associated with the web page in current view), many of the complexities associated with resource sharing and application lifetime are avoided. In spite of this limitation, an unlimited array of new services, applications, and features can be supported.

Level 3: Cooperative downloaded applications

At level 3, which we think is realistically three years away, the technical challenges identified in this paper have been successfully met. This allows the co-existence in one retail cable-compatible device of:

- A resident application suite present in ROM or Flash at the time the product is purchased
- An application suite downloaded upon consumer installation of the device

Furthermore, either or both of these two basic applications can be upgraded (or replaced entirely) via a download update mechanism.

CONCLUSION

This paper began by describing the MSO's vision of the capabilities and benefits of the Middleware Solution for support of software downloads to retail devices. It then explored those aspects that are felt to be impractical or unreasonable.

The paper discussed some of the technical challenges that must be met before a general solution to the code download problem can be reached. The argument was made that the preservation of product differentiation in the marketplace is essential.

The paper concluded with a proposal for "stepwise evolution, where HTML and applet-based approaches would be used in the interim, until the challenges of the downloaded application approach can be fully addressed.

ACRONYMS

API	Application Program Interface
ATSC	Advanced Television Systems Committee
ATVEF	Advanced Television Enhancement Forum
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DASE	DTV Application Software Environment
DOCSIS	Data over Cable Service Interface Specification
DTV	Digital Television
D-VHS	Digital VHS
DVD	Digital Versatile Disk
EIA	Electronic Industry Association
EPG	Electronic Program Guide
FAQ	Frequently Asked Question
HD	High Definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IP	Internet Protocol
IPPV	Impulse Pay-per-View
IRD	Integrated Receiver-Decoder
MPEG	Motion Picture Experts Group
MSO	Multiple System Operator
OS	Operating System
PC	Personal Computer
POD	Point of Deployment
PVR	Personal Video Recorder
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RCU	Remote Control Unit
RFP	Request for Proposals
SCTE	Society of Cable Telecommunications Engineers
SI	Service Information
VOD	Video on Demand
VOIP	Voice over Internet Protocol

* * *

About the author

Mark Eyer has worked in the field of satellite and digital television systems for over eighteen years. He has contributed to various standards-making activities in the ATSC, EIA, and SCTE. He is currently employed as the Director of Systems for Sony Electronics, Digital Media of America in San Diego.

Mark Eyer
Director, Systems
Digital Media of America
Sony Electronics
San Diego, CA 92127
(858) 942-7130
mark.eyer@am.sony.com