

Embedded Application Tools Bring Internet Appliances to Market Quickly

Today, many original equipment manufacturers (OEMs) are developing dedicated "Internet appliances," which provide application-specific access to the Internet. Examples include screen phones, smart hand-held devices, and set-top boxes, to name a few. There is enormous pressure to create these products quickly, or in "Internet time," with development cycles of less than a year from initial product definition to first shipment. Off-the-shelf solutions can help speed time to market by helping OEMs configure, develop, debug, and deploy firmware and software for this new class of embedded products.

INTRO

There's no question about it—more and more people are using the Internet from home to communicate with friends, search for entertainment ideas, and to purchase goods and services. They're connecting to an estimated 9.5 million Web sites all over the world, say analysts who also predict 26 million home users will install dedicated high-performance broadband networks and purchase two or more Internet-access devices in the next few years¹.

The market is wide open for OEMs to sell low-cost Internet-access devices, or "Internet appliances," such as Web terminals, screen phones, or wireless hand-held devices, to home users who want quick and easy access to the Web and its abundant resources. However, developers face some daunting challenges as they begin to capitalize on this opportunity.

End users will expect the devices themselves to be self-evident and easy to use. Web service and content providers will want to access the devices to manage data remotely, track users' buying habits, or upgrade

software easily and cost effectively. On top of that, OEMs have precious little development time, or else run the risk of being late to market. Challenges notwithstanding, designing software for an Internet appliance is a complex undertaking in and of itself.

Today, turnkey software components and tools are available that make it possible to develop "embedded" applications for Internet appliances quickly, economically, and with minimal risk. Instead of re-inventing the wheel every time, developers can take advantage of software that handles standard functions so that they can concentrate their efforts on differentiating their products from the competition. They need an embedded software bundle for the development, deployment, and management of Internet appliances. They also need a Java™-technology-based client/server framework that allows them to manage devices remotely, deploy new services, or modify existing services.

PLUG-IN ARCHITECTURE

Internet appliances are essentially intelligent devices requiring browser capabilities, and, as such, many of

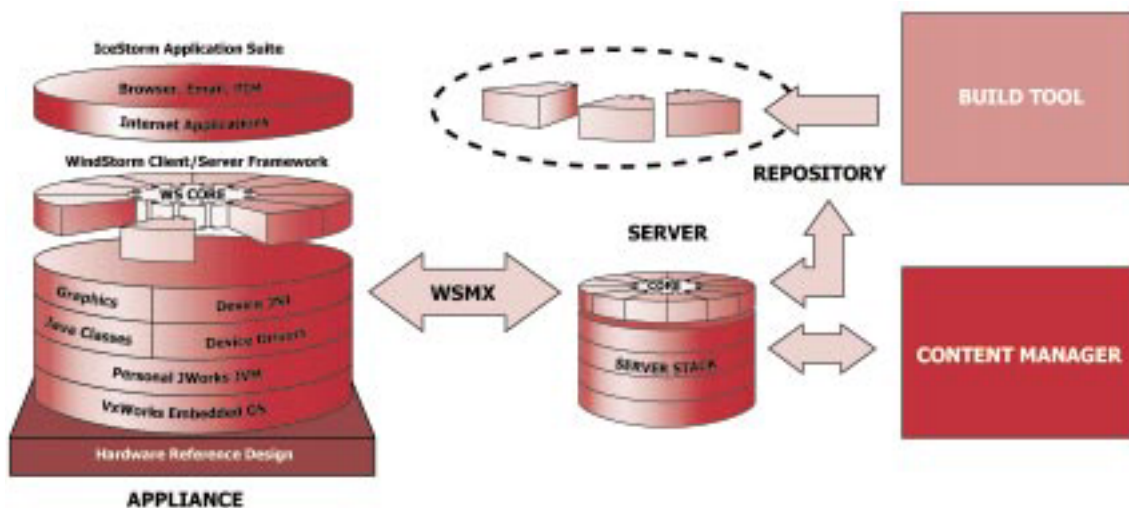


Figure 1. Internet appliance development and deployment architecture.

the components of the devices are fairly standard (embedded http servers, Web browsers, and e-mail clients/servers). Developers can speed time to market for Internet appliances using a modular architecture that allows them to "plug in" standard software components.

The plug-ins - concrete implementations of services and applications that make up an Internet appliance - can have a wide range of functionality from a simple "timer service" plug-in to a fully functional "Web-browser" plug-in. A core Java-technology-based client/server framework manages the plug-ins.

Configuring the software load for a device becomes a straightforward matter of selecting the desired service and application plug-ins to include with the core framework. Plug-ins can also be dynamically installed or removed from a remote location by using any network connection to the Internet.

A MODULAR DESIGN

When using a framework based on a pluggable component architecture, developers need to adopt a component-based viewpoint for the design of their applications and services. Breaking applications into plug-ins makes it easier to define and isolate work tasks and allows several developers to work on the system at the same time. The plug-in approach also forces developers to define clear responsibilities for each plug-in and interface, making testing much easier.

Developers can implement an application in several ways-as a single monolithic plug-in, or several smaller plug-ins. An alarm clock application, for example, could be implemented as a single "application plug-in," but it would only work properly if it remained the active application for the entire time. Once the alarm goes off, the application plug-in's "stop" method is called and any threads used to track the time will be suspended.

A better solution would be to implement the alarm clock as two plug-ins instead of one: a timer "service plug-in" to track the time-till-alarm and an alarm clock "application plug-in" for everything else. The timer function can reference the alarm clock function and it can continue to track the correct time remaining until the alarm goes off, even when the alarm clock application is not the active application.

As illustrated in the example, determining the optimal breakdown of functionality into various service and application plug-ins is a critical step when designing modular applications.

TOOLS

Developers need a powerful set of tools to work within a framework to create projects, add plug-ins, compile and archive plug-ins, and build an executable load. They need only specify core requirements, a list of plug-ins to include, and the end target platform. The tools produce a suitable executable load according to the specifications by compiling and preparing only the necessary classes, libraries, and resources, following a rigid and documented procedure that enables developers to control the process completely.

WANGO GAMING DEVICE

First Real-world Application of Tornado for Internet Appliances

To meet the stringent market demands of the Internet age, RIGEL Technology Corporation, under contract to Lambus Enterprises, Inc., selected Wind River Systems, Inc.'s Tornado for Internet Appliances real-time operating environment and development platform for Lambus' WANGO™ wireless game terminal. This is one of the first real-world applications of the Tornado WindStorm™ architecture.

WANGO is a concept for the delivery of interactive real-time games over a regulated virtual private network. WANGO provides streaming video images to a wireless handheld terminal, enabling players to log on to a particular game through Internet protocols and to compete with other players across town or across the world.

As a Java™ technology-based application, an integrated embedded software bundle is essential to WANGO's deployment. WindStorm is a critical runtime software component of this product, enabling RIGEL to begin development and testing of a graphical interface for various games and game environments. RIGEL is also utilizing WindStorm's simulation environment, which allows them to begin testing applications for the gaming industry prior to actually manufacturing the physical game terminal. Implementing this embedded technology in the WANGO gaming device is significant for both Wind River and RIGEL, allowing each party to test the wireless links for transmitting text and video to many terminals in a very confined area.

The availability of a comprehensive software integration suite has enabled RIGEL and Lambus to focus on their core time-to-market demands and value-add. WANGO is the first of many smart, connected devices that will benefit from the targeted tools and services available through Wind River.

REMOTE MANAGEMENT

Management extensions, based on an implementation of lightweight Java Management Extensions (JMX) and optimized for embedded devices, can allow developers to view, install, and uninstall plug-ins for a specific device connected to the Internet. The extensions can be used during deployment to deliver remote device configuration. After deployment, JMX management applications can be used for device monitoring and software management.

This allows OEMs to design features into their products to upgrade software or track the usage of devices that are already deployed. This is handled dynamically so that end users do not have to reboot their systems when they upgrade or install a new application. A key feature, dynamic remote management, ensures that Internet appliances are stable, flexible, and easy to use.

DISTRIBUTED SERVICES

Java Management Extensions (JMX) also allow OEMs to distribute applications across devices and servers to offload CPU-intensive functions once an application has been partitioned appropriately. A "white-board" application, for example, could be developed with the graphical user interface (GUI) residing on the device with the computational rendering plug-in offloaded to a more powerful server. The ability to distribute services is an especially important feature for Internet Service Providers (ISPs) and Application Service Providers (ASPs).

SIMULATING THE ENVIRONMENT

By simulating the look of a target device, the Java applications developed for a device, and any Java Native Interface (JNI) interactions, developers can build, execute, and test device software in parallel with hardware.

Developers can use a simulation environment to customize the host with a conceptual image of the device, providing complete support to test prototypes. The "simulation skin" can look like a real product. It may contain images of hard keys, indicators, or scroll wheels that invoke changes in their corresponding drivers. The simulator control panel can also provide a GUI to interact with emulate drivers.

A simulation environment also helps developers spend less time integrating software and hardware and speeds time to market.

SUMMARY

A complete client/server software framework that offers device management, content delivery, a suite of pre-built device-centric and services plug-ins, and integrated, host development tools, can be a boon to developers.

Wind River's Tomado® for Internet Appliances gives an integrated embedded software bundle for the development, deployment, and in-field device management of embedded Internet appliances. It includes PersonalJWorks™ a complete PersonalJava execution environment, which runs on the company's VxWorks real-time operating system. At the heart of the software bundle is WindStorm™ a modular, lightweight, embedded software layer and development tool based on Java technology. The product also comes with an integrated Internet applications suite (browser, email, personal information manager), and a hardware reference design complete with board support package and detailed schematics.

Integrated software, hardware, and tools can help with all aspects of network-based applications, speeding the deployment of next-generation "smart" Internet appliances ■

Robert Lepack is a marketing manager for Internet Appliances within the Consumer Business Unit at Wind River, the leader in commercial off-the-self embedded technology. While Wind River's development tools and embedded operating systems are used in a wide variety of embedded applications, Lepack is focused on driving Wind River's Internet appliance solutions that are used to develop a wide array of Internet access devices such as Web-pads, screenphones, net-top boxes, kiosks, and point-of-sale devices. Prior to his current position, Robert worked in product management for AudeSi Technologies, Inc. and in Nortel Networks' Consumer and Wireless Business Units. His experience also includes industrial and user interface design for Internet devices. Lepack holds a BID from Carleton University in Ottawa, Ontario, Canada.



Let us know about your events to be included in Dedicated Systems Magazine and the on-line Dedicated Systems Encyclopaedia!
Contact Nico Van Wijmeersch at +32-2-520.55.77 or info@dedicated-systems.com