From the July 2002 issue of MSDN Magazine

10/23/2019 • 35 minutes to read

Windows CE.NET

New Version Offers Revamped Platform Builder, Improved Tools, Enhanced API, and Source Code

This article assumes you're familiar with Windows CE, Win32, and .NET

Level of Difficulty 1 2 3

Download the code for this article: WindowsCENET.exe (268KB)

SUMMARY Windows CE .NET, the newest member of the .NET family, includes a number of improvements over previous versions of Windows CE. For example, there are quite a few new APIs and enhancements to security and connectivity, the user interface, the kernel, and the emulator. In addition, DirectX support has been added and C++ in Windows CE .NET now supports C++ exceptions, STL, and runtime type information. In this article the author takes a tour of Windows CE .NET, starting with the New Platform Wizard that allows you to code for your choice of devices. A sample application is included that locates features on portable devices so the reader knows what's available before writing code.

The smallest, most mobile, most configurable, and newest member of the Microsoft® .NET family is Windows® CE .NET. Built on the solid foundation laid by Windows CE 3.0, Windows CE .NET is the starting point for embedded, wireless, distributed, connected (both part-time connected and full-time connected) solutions. Microsoft plans to make Windows CE .NET a client in today's Internet-centric paradigm. There will be, for example, a .NET Compact Framework optimized for devices which will provide a Windows CE-specific version of the .NET Framework supported on desktop Windows.

Perhaps the most important new features of Windows CE .NET are the improvements to the tools used to build custom versions of Windows CE. Platform Builder has been completely revamped, and a slew of new wizards have been added.

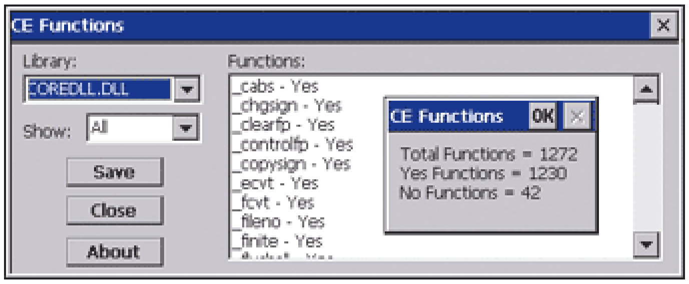
I'll start with a look at the new API in Windows CE .NET and give you a tour of the most important improvements to the Platform Builder toolset. Then I'll cover the new improvements for connectivity, including security features and protocols, kernel and user interface features, and finally I'll wrap up by discussing new C++ enhancements and DirectX® support.

You might be wondering about the recently released Pocket PC 2002. Pocket PC 2002 is based on Windows CE 3.0, not on Windows CE .NET. The Pocket PC 2002 has a few of the features discussed here, including completely reengineered emulation, as well as the MSN® Messenger and the NETUI library. In general, though, Pocket PC programmers can read this article for a taste of what may be available on a future version of the Pocket PC, but not on the currently shipping Pocket PC or Pocket PC 2002 devices.

Before you can really grasp the benefits of Windows CE .NET, you may first need to understand the differences between a Pocket PC and other kinds of devices that you can build applications for using the Platform Builder. The sidebar accompanying this article describes some of these differences.

Of course, Windows CE .NET includes new API functions. To a programmer like me, these are the real operating system features. These hooks let software developers take control and make things happen. To figure out what new functions were being introduced, I compared the DEF files that ship in the Windows CE 3.0 Platform Builder with comparable files from Windows CE .NET. I used the visual file comparison tool, WINDIFF, that ships with Platform Builder to help me find the changes represented by exported functions.

Figure 1 shows my CE Functions tool, CEFUN. This tool, available for download (from the link at the top of this article), checks for the presence of Windows CE-compatible DLLs and functions within each DLL. I built the tool to help me determine what functions are available on different Windows CE devices. Instead of reporting all libraries (some of which are private to specific applications) or all exported functions (some undocumented), CEFUN uses a database of 44 DLLs and 2716 documented functions. I created this database using the Windows CE .NET Platform Builder and the Pocket PC 2002 SDK. I cross-checked function names against published documentation when it was available and against public include files before the documentation was updated.



_cabs - Yes
_chgsign - Yes
_clearfp - Yes
_controlfp - Yes
_copysign - Yes
_ecvt - Yes
_fcvt - Yes
_fileno - Yes
_finite - Yes

Total Functions = 1272
Yes Functions = 1230
No Functions = 42

Figure 1 The CEFUN Tool

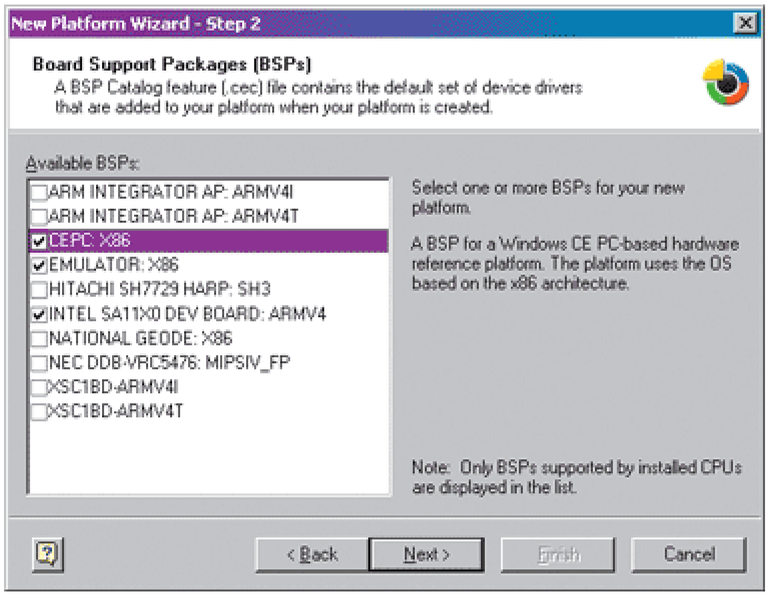
Developers familiar with previous versions of Platform Builder will notice that the Windows CE .NET Platform Builder provides a host of new wizards: a New Platform Wizard, a Component Wizard, and a Board Support Package (BSP) Wizard. Here I'll focus on four new features that will be of interest to most Platform Builder users: the New Platform Wizard, the emulator, the source code included with the Platform Builder, and the improvements in network-based debugging support.

The New Platform Wizard

The New Platform Wizard provides the starting point for anyone creating a platform built on Windows CE .NET. This tool lets you create a wide range of Windows CE configurations with just a few mouse clicks. It walks you through a series of steps, each of which has a property sheet page to show available choices. Later on if you change your mind, it's a simple matter to add items to a platform by dragging components from the Platform Builder catalog. You can configure Windows CE .NET in as few as 4 steps (for Tiny Kernel) or as many as 17 (for a custom configuration).

The two most notable improvements in the Windows CE .NET New Platform Wizard are broader support for different BSPs (Step 2 in the wizard), and a new set of basic operating system configurations (Step 3 in the wizard).

Figure 2 shows the available BSPs that can be selected from Step 2 in the New Platform Wizard. A BSP can be thought of as a motherboard support package. It provides the basic set of device drivers for a particular board and also the OEM Adaptation Layer (OAL). An OAL provides the basic set of services that the Windows CE kernel needs to run. When a Windows CE image is built, the OAL is bound into the kernel (NK.EXE), and provides support for such things as CPU startup code, the clock for the scheduler, definitions for handling interrupts, and debugger support.


New Platform Wizard - Step 2

Board Support Packages (BSPs)
A BSP Catalog feature (.cec) file contains the default set of device drivers that are added to your platform when your platform is created.

Available BSPs:
- ☐ ARM INTEGRATOR AP: ARMV4I
- ☐ ARM INTEGRATOR AP: ARMV4T
- ☑ CEPC: X86
- ☑ EMULATOR: X86
- ☐ HITACHI SH7729 HARP: SH3
- ☑ INTEL SA11X0 DEV BOARD: ARMV4
- ☐ NATIONAL GEODE: X86
- ☐ NEC DDB-VRC5476: MIPSIV_FP
- ☐ XSC1BD-ARMV4I
- ☐ XSC1BD-ARMV4T

Select one or more BSPs for your new platform.

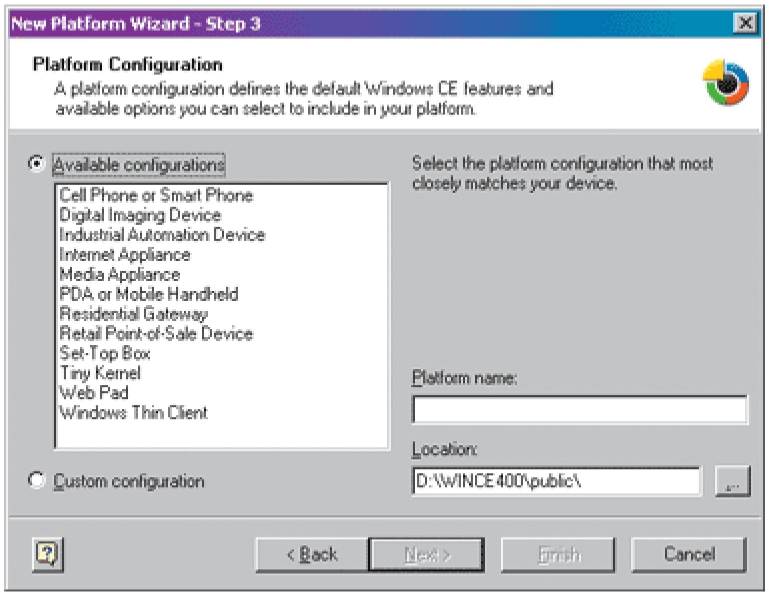
A BSP for a Windows CE PC-based hardware reference platform. The platform uses the OS based on the x86 architecture.

Note: Only BSPs supported by installed CPUs are displayed in the list.

Figure 2 Available BSPs

The Windows CE 3.0 Platform Builder shipped with only two BSPs; the Window CE .NET Platform Builder ships with ten. One is for the Platform Builder emulator discussed in the next section. All the rest are for specific hardware boards that are available from different third parties. Because there is a BSP available, developers of new systems don't need to build a system from scratch. Instead they can license a design and obtain drivers more quickly than could be done without a BSP. Additional BSPs can be added to the Platform Builder so the set of available BSPs is even larger than the list shown in Figure 2. (The Platform Builder provides a wizard to help third parties create new BSPs.)

Step 3 of the New Platform Wizard (see Figure 3) lets you pick a base configuration from a set of 12 configurations or put together a custom configuration. This new set replaces the earlier, simpler set of configurations such as MINKERN, MINGDI, and MAXALL that were built more or less around the Microsoft test cycle for Windows CE. The new set is shaped by a much broader vision of the kinds of devices that platform developers are likely to create.


New Platform Wizard - Step 3

Platform Configuration

A platform configuration defines the default Windows CE features and available options you can select to include in your platform.

(●) Available configurations

Cell Phone or Smart Phone
Digital Imaging Device
Industrial Automation Device
Internet Appliance
Media Appliance
PDA or Mobile Handheld
Residential Gateway
Retail Point-of-Sale Device
Set-Top Box
Tiny Kernel
Web Pad
Windows Thin Client

( ) Custom configuration

Select the platform configuration that most closely matches your device.

Platform name:

Location: D:\WINCE400\public\

Figure 3 Available Configurations

When you look at the list of basic configurations, the most obvious theme is support for networked devices. In fact, 11 of the 13 allowable configuration choices have networking hardwired into their definition in the form of anchored features. The only two that don't are the Tiny Kernel and the custom configuration, both of which would allow the addition of network support.

A more important difference is whether a device has a GUI display (known as an IABASE device) or whether it is a non-GUI (known as a headless or HLBASE) device. Figure 4 shows how 12 configurations stack up on this vital question. The final configuration, the custom configuration, allows for either setting. In creating this set of base configurations, the Platform Builder team has provided a more user-oriented set of new platform starting points.

One drawback to this reconfiguration is that some effort is needed to move existing Windows CE 3.0 platforms to Windows CE .NET. There is no automatic migration path.

Emulator

The Windows CE .NET Platform Builder introduces an emulator for embedded system developers. Previously, the only emulation that was supported for Windows CE development was for standard devices like the Handheld PC and the Pocket PC, not for custom Platform Builder-based platforms. This emulator is much improved over previous Windows CE emulators in that all of Windows CE itself is actually running in the emulator. It is not simply a simulation of Windows CE using the support of the underlying Windows 2000 system libraries. So, after you have used Platform Builder to configure a custom version of Windows CE .NET, you can build and run the resulting image on your development machine in Windows 2000 or Windows XP.

The emulator saves development time by alleviating the bottleneck often caused by a shortage of hardware. When you are just getting started with a new custom device, the emulator lets you experiment with different configurations of the operating system without having to wait for the hardware to be built and debugged.

There are some limitations, though. In particular, any development that relies on a specific device that is not standard on a PC (like a bar-code scanner or a digital camera) will require either a dedicated device or a custom extension to the emulator to simulate the presence of the device. Even in such cases, there are often things that can be accomplished without dedicated hardware. For example, UI code or support libraries can be created with the generic, PC-based hardware that the emulator supports.

Device drivers, on the other hand, will likely require the presence of the actual target device. Even here the emulator can help, because you can build and test the basic framework for a new driver in the emulator, moving it to an actual custom platform only when you need to touch the specific device.

The emulator is built on a product from Connectix Corporation of San Mateo, California (http://www.connectix.com) called the Virtual x86 Reference Board. (You can download the Microsoft Windows CE .NET, Emulation Edition here.) To the Platform Builder, it looks just like any other BSP, even having its own directory in the ..\PLATFORM directory in the Windows CE build tree. Like any other device, it has its own OAL, its own device drivers, and of course its own target CPU. In practical terms, this means you can customize any of these elements to more closely resemble the hardware you are developing for. If you have any CPU-specific code such as an ARM floating point library or a MIPS graphics library, you must substitute an x86 version of that code to be able to run under the emulator.

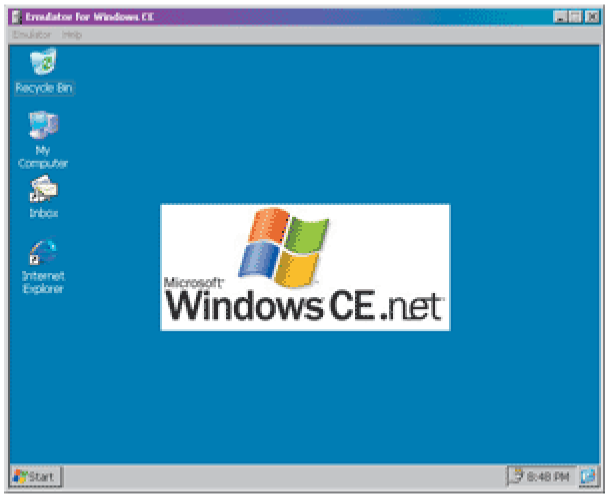


Figure 5 Cool New Emulator

Aside from the new, spiffier look (see Figure 5), the most important new feature of this emulator is a greater similarity between the behavior of the PocketPC 2002 emulator (which was built using the same technology) and an actual, shipping device. An OEM that wants to customize the look of the emulator can create a new UI skin, to make the emulator more closely resemble the actual device being emulated. And while the low-level machine instruction differences—x86 instead of ARM instructions—might mean that you have to rely on device-based testing for issues like misaligned data access, developers will likely find that the new emulation is a very big improvement over what they had to work with in previous SDKs.

Windows CE Source Code

An important addition to Platform Builder is a select subset of the source code from Windows CE. The availability of CE source code is not new; Windows CE 3.0 source code has been available since September 2000, but it's noteworthy. Few developers use this resource when it could provide assistance at a low cost. And the scope of available source code has grown. The earlier offering involved about 300 source files, while the new set adds over 1000 source files to bring the total to almost 1400. Of course, a count of files is a crude way to measure value—like counting lines of code to measure programmer productivity. So, perhaps you're wondering why this is so valuable. I will take a moment to describe some of the gems you'll find.

For starters, the source code includes the Windows CE kernel. Have a question on memory allocation, thread scheduling, DLL loading, or interrupt handling? It's all there, spelled out for you not as indecipherable OBJ files, but in languages you know: C and C++ (with tantalizing bits of ARM/SH3/x86/MIPS assembler to whet your appetite for low-level stuff). Are you building a new OAL and wondering just when and why it is called? Simply search for the OEM prefix, and you'll get a screenful (and more) to help you understand the intimate relationship between the CE kernel and your OAL.

Building a display driver? Then you are probably using the Graphics Primitive Engine (GPE) wrapper classes. To help you make better use of this class library, the Platform Builder source code includes the complete source files to the GPE classes. You'll also find a variation of GPE that supports rotating a CE-based display driver's image by 90, 180, and 270 degrees.

Or perhaps your work takes you to more Internet-centric parts of Windows CE. The source files will help you here as well. Want to see how Windows CE supports ASP files in its Web server? You can find the source files to what amounts to a Windows CE-based ISAPI extension that provides ASP support in the CE Web server, HTTPD. Perhaps that's not low-level enough for you? Well, you'll find the source code to HTTPD, the Web server, as well.

Maybe you prefer to work with MSMQ and are having problems with some unexplained behavior. To help you, the source files to various MSMQ components are included, as are source files to the SOAP support that is provided with Windows CE and the Universal Plug-n-Play (UPnP) components.

I personally expect to spend much time over the next few years digging into this rich mountain of data. An hour spent spelunking into the source files will save at least a day of staring at your own misbehaving code. Just remember, if you read the licensing agreement you'll see that the source code is available to help you learn about Windows CE and to help you debug your systems. You cannot build and ship any portion of these source files for commercial purposes, but you can build the sources to help you debug your Windows CE-based systems. You can also build the kernel for noncommercial purposes (for example, in school to learn about embedded systems or as a hobbyist to create a radio-controlled robot).

Connectivity Improvements

Another enhancement in the Windows CE .NET Platform Builder is improved support for debugging over a network. This support was already present in Windows CE 3.0, but if you let Windows CE know about your network adapter (for Windows CE-based Web, FTP, or socket access) you were required to have a second network adapter to support image download and device debugging. With Windows CE .NET, a single network adapter can be used for both purposes.

Although you are able to download an image using the parallel port of your development system, doing so will slow the pace of your development project considerably. If you can participate in the design of your hardware, you ought to suggest—even insist—that it support networking download and debug. The key benefit over the other alternatives is speed. While a parallel port may give you 100KB throughput, a network adapter will give you at least 100 times that speed. So instead of waiting over 10 minutes to download a 10MB image when using a parallel port download, that same image will download in 15 seconds over a 10MB network. You probably have better things to do with your life than wait for binary images to download.

A long-standing strength of Windows CE has been its connectivity. Windows CE .NET continues to push the envelope, adding new device drivers, support for new protocols, and new APIs.

Next, I'll discuss features and protocols available for creating secure wireless connections. I'll discuss a standard interface for a security protocol known as Security Support Provider Interface (SSPI), the Bluetooth protocol, 802.11 (a protocol for wireless device to access an Ethernet-based network), and 802.11 automatic configuration support to simplify the setup of 802.11-based networks. Finally, I'll talk about support for XML and SOAP, support for the Object Exchange (OBEX) protocol, UPnP, real-time communications, and virtual private networks.

SSPI

Windows CE .NET improves security for network-connected devices. Since version 2.0, Windows CE has supported SSL, the Encryption API (in CRYPT32.DLL), and X.509 Digital Certificates. Windows CE .NET adds to these existing features a standard interface for the security protocol known as SSPI. This effectively allows you to create a custom security interface. The role of an SSPI in Windows CE .NET is the same as for Windows 2000: to provide authentication, data integrity, and data privacy. (These are covered in great detail in the *Microsoft Systems Journal* article "Exploring Kerberos, the Protocol for Distributed Security in Windows 2000" by David Chappell, August 1999).

The support that Windows CE .NET provides for SSPI is essentially a subset of what is already available on desktop versions of Windows. While few developers are likely to create custom SSPIs, most are likely to incorporate the two that ship with Windows CE .NET: NTLM and Kerberos. NTLM is an older security package that you'll use to authenticate a Windows CE device with Windows NT® servers. Kerberos is the newer, more flexible, and more secure package that you will likely use for servers running Windows 2000 and Windows XP. An additional advantage of Kerberos is that it is an industry standard, making it a good choice when you plan to connect to servers in a multivendor setting.

Bluetooth is a protocol for wireless device connectivity. The promise of Bluetooth is the emergence of Personal Area Networks (PAN) in which laptops, desktop computers, cell phones, printers, and so on communicate with one another. The range of basic Bluetooth devices is 10 meters (30 feet), and eight devices can be strung together to form a single network, referred to as a Piconet.

There are two generic mechanisms for applications to use Bluetooth: using the network-oriented Windows Sockets API and using a virtual communications (COMx:) port. These are the same two mechanisms available for accessing infrared ports. The Windows Sockets API is the standard network API that has its origins in Berkeley Sockets. A virtual communications port uses standard Win32 functions (CreateFile, ReadFile, WriteFile, CloseHandle) to open, send, and receive data in a Bluetooth device.

I saw a demonstration in which a Bluetooth-enabled Pocket PC accessed a Bluetooth-enabled cell phone to call out to the Internet. Bluetooth could be used for wireless access in many situations, including for wireless headsets for phones or stereos, and as a replacement for any application that currently uses infrared. A key advantage of Bluetooth over infrared is that you do not need to align two ports to communicate, which is only partially reliable whether you are talking about communicating between two Pocket PCs (to exchange contact or schedule information) or between a TV and its remote. Although the 1MB throughput means that Bluetooth is not likely to replace LAN adapters for point-to-point communication tasks (like synchronizing a mobile device or programming a VCR) it has plenty of bandwidth.

You can get more information at http://www.bluetooth.com.

802.11 and 802.11 Auto Configuration Support

802.11 is a protocol for wireless devices to access an Ethernet-based network. Since devices based on 802.11 first became available, they have become quite widespread, both on corporate and academic campuses. Microsoft, for example, has 802.11 coverage at their Redmond campus that lets roaming devices, both laptops and Pocket PCs, access the corporate network from any room. A number of airports provide 802.11 access points, as do some hotels and Starbucks coffee shops (thanks to MobileStar, http://www.tmobilebroadband.com). I have tried it out equipped with my laptop and its 802.11 card and happily enjoyed accessing both e-mail and the Web. The 11MB throughput provides more than adequate response time when I am away from the office.

802.11 automatic configuration support simplifies the setup of 802.11-based networks. It enables roaming between 802.11 networks by tracking the information required by the driver and operating system and seamlessly configuring the network card as required by the network, according to the user's preferences. Once a network is configured, the OS remembers the configuration and automatically detects when it needs to switch. It also simplifies initial configuration by determining most of the required settings automatically and by letting the user know when it has discovered a wireless network that it can use. Windows XP features provide the same experience to laptop users.

Another feature relating to 802.11 is 802.1x, which enhances wireless security. The IEEE 802.1x standard helps address several issues, including authentication (enabling only authorized people/devices on the wireless network) and Wireless Equivalency Protocol (WEP) key management. WEP keys are limited in their ability to provide secure communications. 802.1x is a mechanism for negotiating and changing WEP keys at predetermined events to help improve the security of 802.11-based wireless networks.

XML and SOAP

Support for XML in Windows CE .NET consists of a subset of the XML version 3.0 (Service Pack 1) parser found in Windows XP. This includes a whole rogue's list of features that XML devotees have come to know and love. The in-memory access to an XML hierarchy is provided in the form of the DOM. A second API for accessing XML data, the Simple API for XML (SAX) is also supported for situations that require a more memory-conserving operation. Windows CE .NET also supports XSL to allow you to convert an XML document written in one dialect to another dialect. You can perform queries on XML hierarchies using XPATH queries, validate XML data using schemas, and access XML data from a Web server using the XMLHTTP object. Just like many other parts of Windows CE, the XML support is componentized so you can include as much or as little as you need.

Every XML component is required for SOAP, an industry standard for communicating between network-based machines using common protocols like HTTP and XML. It is a way to export a COM interface from a Web server. In the new world of .NET programming, this is called a Web Service.

Windows CE .NET-based devices can be configured as either a SOAP Server or a SOAP client. For both roles, a Windows CE .NET device needs to include support for COM. To run as a SOAP server, a Windows CE .NET device must include HTTPD.

To support SOAP on a Windows CE .NET-based device, you need COM, which comes in three flavors. But which is sufficient for SOAP? First consider the three types. Minimal COM, which is only available for headless (HLBASE) systems, supports basic object creation. Midrange COM supports Automation and in-process servers with freethreaded objects. Distributed COM (DCOM) supports full-featured COM.

To support SOAP, you must have either midrange or full DCOM. It's important to note that the samples that ship with the Platform Builder require full DCOM because they do not support freethreaded objects. Although the Platform Builder does not include a sample, it is quite possible to build SOAP support using the midrange COM support, providing that you build only free-threaded objects.

OBEX

Windows CE .NET provides support for the OBEX Protocol, a lightweight, XML-based, HTTP-like protocol for moving files and other data between two devices. OBEX is a client/server protocol that is implemented on Windows CE as a set of COM interfaces. OBEX support in Windows CE .NET is provided in two transports, Infrared and Bluetooth. Presumably, more transports could be added as needed, but the support for these two transports demonstrates why OBEX was created: for data sharing between devices in close proximity.

OBEX is one of the few areas where the Pocket PC 2002 incorporates Windows CE .NET technology, although there are differences that I have yet to resolve between the library names and include files. I suspect that these are differences of implementation rather than of the underlying protocol, since two devices with a common protocol should always be able to communicate, whether or not the details of how that protocol is supported are the same.

Universal Plug-n-Play

In the category of new protocols, Windows CE .NET provides support for UPnP. This TCP/IP-based network protocol allows a network device to attach to a network, advertise its services, discover other network devices, and cleanly disconnect from the network. The UPnP architecture divides the world into control points and devices, and a Windows CE .NET device can play either role. A control point is a user-driven computer, like a laptop computer or a Pocket PC. A device can be standalone like a digital camera, a cluster of devices like an all-in-one printer/fax/copier, a service such as a print spooler, or any combination of these.

UPnP was introduced in 1999. It relies on an alphabet soup of standard Internet technologies, including DHCP, ARP, HTTP, SOAP, and XML. The presence of UPnP support in the Windows CE .NET Platform Builder means that developers of new custom devices for Windows CE can build in UPnP support to allow them to interconnect and interact with other network-based devices.

For more details, contact the UPnP Forum (http://www.upnp.org). This industry group, which has about 400 members, promotes interoperability between UPnP devices as its primary role.

Real-time Communications

Real-time communications (RTC) is a feature of Windows CE .NET that supports the creation of devices for communicating between people using new stacks that sit on top of TCP/IP. This is the third type of real-time support in Windows CE. The first has to do with interrupt and thread scheduling, and the second with reporting the current time and date (the real-time clock) in the OAL. Communications is the key to understanding RTC, which has little to do with the other two types of real-time support.

The Windows CE .NET RTC allows messaging between people in a variety of forms, including file transfer, voice, and text. The three types of RTC stacks that have been added are accessible through the new RTC API. The RTC support in Windows CE .NET builds on what you find in the Pocket PC 2002 in the MSN Messenger application. The Pocket PC 2002 features two RTC clients that work with the Internet-based .NET Messenger Service, which are validated using the .NET Passport Service. What has been added to Windows CE .NET is support for the standard Session Initiation Protocol (SIP), so that a Windows CE .NET-based device can communicate with SIP phones from other vendors like Cisco and Siemens. An enterprise-based intranet messenger service can be set up with Exchange Server 2000. The Windows CE .NET RTC does not support transfer of video or the application sharing or whiteboarding features available with the MSN Messenger on desktop versions of Windows.

The most basic type of RTC support is text-only messaging, sometimes called instant messaging. The MSN Messenger Service lets you create a contact list and will tell you when their messenger clients are connected. You can set up a watchers list to let people know when you are connected. At such times, you can connect to the messenger clients associated with your friends and type messages which are immediately displayed in their messenger client. When they respond, you see the text in your messenger client.

The instant messaging contact list knows when your friend or family member is connected. An emerging enhancement is presence information, which identifies which device from a list of registered devices to use to contact a person, whether by instant messaging, by cell phone, or by pager.

Voice communication is known as Voice over IP (VoIP), which is enabled through a Telephony Service Provider (TSP) that gets called by the Telephony API library, TAPI.DLL. A VoIP-enabled device can communicate to other devices running Windows CE .NET, to a Windows XP machine, to a regular telephone, or to Public Switched Telephone Network (PSTN).

The Web site http://messenger.msn.com provides links to a number of MSN Messenger clients, including one for the Macintosh, Pocket PC, and Microsoft TV.

Virtual Private Network

VPN provides a mechanism for a networked device to connect to a public Internet connection (for instance, at a client's location or via dial-up modem) and establish a secure connection to a server—an invaluable capability that lets you access e-mail, file servers, and other intranet resources on the road.

Windows CE .NET supports VPN through either a dial-in connection to an ISP or a hardwired connection to a network. In this scenario, the Windows CE .NET device is the VPN client; it does not accept incoming connections but can call out to other network servers using VPN.

New Kernel Features

The core of Windows CE is the kernel, which provides services such as memory management, file I/O, module loading, scheduling, and device driver support. There are kernel drivers and non-kernel drivers; non-kernel drivers belong to the Graphics, Windowing, and Events subsystem (GWES)—(the UI system), and include mouse, keyboard, display, and printer drivers.

The last major overhaul to the Windows CE kernel was for version 3.0 of Windows CE. At that time, a number of real-time enhancements were added. (For details, see "Windows CE 3.0: Enhanced Real-Time Features Provide Sophisticated Thread Handling," in the November 2000 issue.) Aside from real-time refinements, the basic operation of the Windows CE kernel has been rather static, it is a multithreaded kernel with support for up to 32 processes, each of which has an address space that cannot exceed 32MB. A shared memory area holds objects that will be accessed by two or more processes simultaneously and large objects that cannot fit inside the 32MB process address space limitation. To this kernel, Windows CE .NET adds refinements in a number of areas. I will focus on three here: the addition of support for fibers, the new point-to-point message queues, and enhancements for device drivers.

Fibers

Windows CE .NET adds support for fibers. To make sense of fibers, you must first understand threads. Threads, which are owned by processes, are the entities which are scheduled. Processes are not scheduled. A single process always has at least one thread, but it can have many more. Threads allow multiple, simultaneous activities, including waiting for input from a network port, sending output to a printer, and waiting for user input. The operating system decides when to schedule a given thread based on thread state: waiting, ready-to-run, and so on, and on that thread's priority relative to other ready-to-run threads.

Fibers represent a refinement to threads. A single thread can run two or more fibers. When two fibers run on the same thread, the operating system still decides when the thread runs. A fiber always runs during its thread's scheduled time. It is up to the fiber owner (the application or the DLL), not the operating system to determine which specific fiber is to run. Interestingly, this is the same type of cooperative, nonpreemptive scheduling that 16-bit Windows applications used in every version of Windows from version 1.01 up to the time Windows NT® 3.1 shipped, which was when Windows got a preemptive scheduler and support for threads.

The primary benefit of fibers is lower overhead. Multiple fibers let you maintain multiple execution contexts without the overhead of threads. And when properly designed, multiple fibers can even run faster than an equal number of threads. The reason is that synchronization can be largely avoided. It's not needed because multiple fibers on the same thread are self-synchronizing. That is, you do not need synchronization objects—critical sections, mutexes, and the like—to prevent two fibers on the same thread from accessing the same shared object. As long as the fiber scheduling mechanism is designed to switch fiber execution contexts only when all shared objects are in a stable state, you are free to write thread-safe multithreaded code without the overhead that's inherent in synchronization.

Point-to-point Message Queues

In introducing point-to-point message queues, Windows CE .NET adds a third type of message queue to Win32. The first type is used to receive user events and communicate them to window procedures, and it is associated with functions like SendMessage and GetMessage.

The second type is associated with the MSMQ API, which provides a mechanism for sending datagrams between networked computers. With this second type of message queue, data gets stored when the network is unavailable and forwarded when it becomes available. MSMQ function names start with an "MQ" prefix and include functions like MQCreateQueue, MQBeginTransaction, and MQSendMessage. Both types of message queues are supported on Windows CE .NET.

Point-to-point message queues represent a unique contribution by Windows CE to the Win32 API; these functions are not available in desktop versions of Windows. This type of message queue serves up an interprocess communication (IPC) technique designed to send data from one process to another. Technically speaking, this is not a kernel feature. But you get the same type of IPC that in other operating systems is provided by pipes, which are traditionally implemented by an operating system's kernel.

Figure 6 summarizes the six functions that make up support for point-to-point message queues. This is a fairly straightforward set of functions which will be helpful in many different places.

Device Driver Enhancements

Windows CE .NET adds a standard mechanism for an installable Interrupt Service Routines (ISRs). On previous versions of Windows CE, all interrupt handling was built into the OAL at system build time. Once a device was built, there was no standard way to assign new interrupts to devices that hadn't been included in the original build.

On Windows CE .NET, the only requirement is that unassigned interrupt requests (IRQs) need to be clearly identified. The drivers for new devices register the new ISRs when the devices are installed by calling the system's LoadIntChainHandler function to hook the IRQ handler function to its list of IRQ functions. The device driver DLL that implements the IRQ handler function must also implement and export two additional helper functions named CreateInstance and IOControl. An ISR that was installed with LoadIntChainHandler can also be detached with a call to FreeIntChainHandler.

New UI Features

Perhaps the most obvious new Windows CE .NET features are the UI enhancements. Some of these are cosmetic, including new icons and support for animated icons. The new icons are borrowed from Windows XP for consistency. Figure 7 shows the Control Panel from Windows CE .NET.

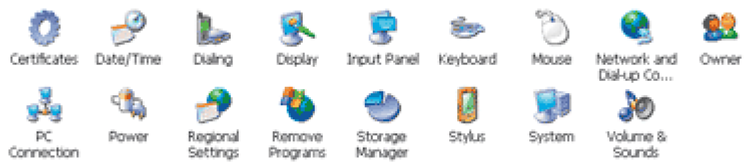


Figure 7 Windows CE .NET Control Panel

One improvement for the Windows CE .NET user interface is the ability to create a high level of customization of the UI, a feature that is referred to as a skin (and which is sometimes referred to as a skinnable UI). Windows CE .NET ships with two skins and the ability to create your own custom skins. The two included skins look like Windows 95 (the default) and Windows XP.

There has always been a high degree of UI customization possible. Prior to Windows CE .NET, it was easy to replace one set of icons with another. The Control Panel, for example, can easily be given a new look for any version of Windows CE. Because Windows CE .NET actually uses many icons from Windows XP, it is a bit hard to see the differences between the two sample skins. But when you dig into the sample code, you see the extent of control available for fine-tuning the UI.

To anyone who has written code for the Windows owner-draw controls, this is familiar. But instead of redrawing the code for individual controls, when you customize a skin you change the appearance of every control. Interestingly, though, there is no overlap between what you can customize using owner-draw controls and what you can change with skins. This means that applications with owner-draw controls can coexist peacefully on devices that have custom skins. Figure 8 summarizes the various types of controls, indicating which can be modified as owner-draw controls and which in user-interface skins.

The real power of this feature comes into play when you want to a change things like the default look of buttons and edit controls. In fact, you can change any of the standard controls (edit, list box, static, button, combobox) as well as some of the common controls. In addition, you can change the system colors and the tiny bitmaps called widgets that are used for scrollbar buttons, as well as checkboxes, radio buttons, the OK button, and the go-away button that appear in the upper-right corner of windows.

Font Picker Common Dialog

There are eight common dialogs on desktop versions of Windows. Previous versions of Windows CE supported five of these: File | Open, File | Save, Print, Page Setup, and Color Picker. Windows CE .NET adds a new common dialog, the Font Picker (see Figure 9). The key advantage of the common dialogs is that a minimum amount of coding is needed to get the benefit of a standard UI. When Microsoft ships a new OS version that provides refinements to common dialogs, applications that use them get an immediate upgrade. (One difference between common dialogs in Windows and Windows CE is that two of the common dialogs, File | Open and File | Save, do not reside in COMMDLG.DLL; instead they are part of COREDLL.DLL.)

Coding for common dialogs is straightforward and follows a standard pattern—fill a data structure, then call a support function. Each data structure contains a member that is the size of the structure, and if you fail to initialize this value, the call to the function fails. Figure 9 summarizes the six supported common dialog functions and their associated data structures.

As useful as the common dialogs are, they are not the only dialog boxes built into the OS. Windows CE .NET adds a browser for file folders that has been on Windows for many years. The name of the function for accessing this dialog is SHBrowseForFolder.

There is also a library named NETUI.DLL that has a number of user-interface elements that can be useful for configuring and controlling network connections. These functions, which are summarized in Figure 10, are also available on the Pocket PC 2002.

C++ Enhancements

For C++ programmers, Windows CE .NET adds support for three key features: C++ exceptions, Runtime Type Information (RTTI), and the Standard Template Library (STL).

C++ exceptions are new in Windows CE .NET. Windows CE has always supported Win32 exceptions, represented by compiler keywords like __try, __except, __finally, and __leave. (The leading underscores are a hint that these are not ANSI/ISO standard keywords, but instead represent Microsoft language extensions.) While the two are conceptually similar, C++ exceptions are a type-based exception mechanism, and Win32 exceptions support 32-bit integer exception codes. The C++ keywords are similar—try, catch, throw—but they do not have the leading underscores. Developers who want to port existing code to Windows CE will welcome the inclusion of C++ exceptions.

C++ RTTI support extends the type-safety mechanism of C++ in several important ways. First, they provide a standard, programmable mechanism for examining object types. You can query for the type of the object being referenced by a pointer using the typeid operator, and then select how you will treat that object based on its type. If a software company had different categories of customers, premier versus regular, then the software that handled customers would easily be able to distinguish between, say a CRegularCustomer and a CPremierCustomer.

A second benefit of RTTI is improved casting. The developer of the C++ language, Bjarne Stroustrup, calls casts "...one of the most error-prone facilities in C++...also one of the ugliest syntactically." The dynamic_cast operator lets you perform a cast in a type-safe way. While previously C/C++ casts let you get away with just about any cast, a dynamic_cast will not let you cast pointers to objects in ways that break the rules of inheritance. By examining the type of object references, a dynamic cast can allow casting to a base class, but block casting when the base object is not actually referenced by the pointer.

The third C++ enhancement for Windows CE .NET is support for STL. This support has been available from third parties, but Windows CE .NET is the first version of Windows CE for which STL support has been provided with the operating system itself. This library, called the Standard C++ Library in the MSDN® documentation, is built around templates, a C++ feature for creating new, parameterized data types.

New DirectX Support

DirectX is the name for a set of multimedia services that Microsoft created for Windows 95 to allow low-level access to a variety of devices, with the primary goal of allowing game developers to build high-performance games. It is, in essence, a standard low-level interface into a wide range of devices including, on Windows CE, the display (DirectDraw® and Direct3D®), sound devices (DirectSound® and DirectMusic®), game controllers (DirectInput®), streaming media (DirectShow®), and multi-player game access (DirectPlay®). Figure 11 compares DirectX support on various versions of Windows.

Conclusion

Windows CE .NET presents embedded systems developers with a bevy of new capabilities. I have touched on the most important features, including the enhancements to the tools, a broader set of connectivity support, an upgraded kernel, a more customizable UI, and an increased set of DirectX multimedia features. But I have only skimmed the surface of all that you'll find. Fortunately, you can try out Windows CE .NET yourself with a trial version of the Windows CE .NET Platform Builder. This version will let you build and download images to the x86-based emulator, read the documentation, study the source code of available drivers, look at the source code for the Windows CE .NET kernel (and other components), and get a more specific idea of how this new embedded operating system can fit into your future.

For related articles see:
Embedded Operating System Development
Windows CE .NET Help
About Platform Builder
Windows CE .NET Overview

__Paul Yao__ is a Windows programming guru, Windows CE expert, and coauthor of the first book published on Windows programming. His company specializes in training programmers in Windows and Windows-related technologies. Visit his Web site at http://www.paulyao.com.