



7 Programming

The following programming information pertains to the CN2B Mobile Computer:

- Creating CAB Files (page 126)
- Customization and Lockdown (page 140)
- FTP Server (page 141)
- Kernel I/O Controls (page 150)
- Network Selection APIs (page 161)
- Notifications (page 181)
- Reboot Functions (page 182)
- Remapping the Keypad (page 182)

Creating CAB Files

The Windows CE operating system uses a .CAB file to install an application on a Windows CE-based device. A .CAB file is composed of multiple files that are compressed into one file. Compressing multiple files into one file provides the following benefits:

- All application files are present.
- A partial installation is prevented.
- The application can be installed from several sources, such as a desktop computer or a Web site.

Use the CAB Wizard application (CABWIZ.EXE) to generate a .CAB file for your application.

Creating Device-Specific CAB Files

Do the following to create a device-specific .CAB file for an application, *in the order provided*:

- 1 Create an .INF file with Windows CE-specific modifications (*see below*).
- 2 *Optional* Create a SETUP.DLL file to provide custom control of the installation process (*page 136*).
- 3 Use the CAB Wizard to create the .CAB file, using the .INF file, the optional SETUP.DLL file, and the device-specific application files as parameters (*page 139*).

Creating an .INF File

An .INF file specifies information about an application for the CAB Wizard. Below are the sections of an .INF file:

[Version]

This specifies the file creator, version, and other relevant information.

Required? Yes

- **Signature:** *"signature_name"*
"\$Windows NT\$"
- **Provider:** *"INF_creator"*
The company name of the application, such as "Microsoft."
- **CESignature**
"\$Windows CE\$"

Example

```
[Version]
Signature = "$Windows NT$"
Provider = "Intermec"
CESignature = "$Windows CE$"
```

[CEStrings]

This specifies string substitutions for the application name and the default installation directory.

Required? Yes

- **AppName:** *app_name*
Name of the application. Other instances of %AppName% in the .INF file are replaced with this string value, such as RP32.
- **InstallDir:** *default_install_dir*
Default installation directory on the device. Other instances of %InstallDir% in the .INF file are replaced with this string value. Example:
\\SDMMC_Disk\\%AppName%

Example**[CEStrings]**

```
AppName="Game Pack"
InstallDir=%CE1%\%AppName%
```

[Strings]

This section is optional and defines one or more string keys. A string key represents a string of printable characters.

Required? No

- **string_key:** *value*
String of letters, digits, or other printable characters. Enclose *value* in double quotation marks “” if the corresponding string key is used in an item that requires double quotation marks. No string_keys is okay.

Example**[Strings]**

```
reg_path = Software\Intermec\My Test App
```

[CEDevice]

Describes the targeted application platform. All keys are optional. If a key is nonexistent or has no data, Windows CE does not perform any checking with the exception being *UnsupportedPlatforms*. If the *UnsupportedPlatforms* key exists but no data, the previous value is not overridden.

Required? Yes

- **ProcessorType:** *processor_type*
The value that is returned by SYSTEMINFO.dwProcessorType. For example, the value for the ARM CPU is 2577
- **UnsupportedPlatforms:** *platform_family_name*
This lists known unsupported platform family names. If the name specified in the [CEDevice.xxx] section is different from [CEDevice], both *platform_family_name* values are unsupported for the microprocessor specified by xxx. The unsupported platform family names list is appended to the previous list. Application Manager does not display the

application for an unsupported platform. Also, a user is warned during the setup process if the .CAB file is copied to an unsupported device.

Example

[CEDevice]

UnsupportedPlatforms = pltfrm1 ; pltfrm1 is unsupported

[CEDevice.SH3]

UnsupportedPlatforms = ; pltfrm1 is still unsupported

- **VersionMin:** *minor_version*
Numeric value returned by **OSVERSIONINFO.dwVersionMinor**. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **VersionMin**.
- **VersionMax:** *major_version*
Numeric value returned by **OSVERSIONINFO.dwVersionMajor**. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **VersionMax**.
- **BuildMin:** *build_number*
Numeric value returned by **OSVERSIONINFO.dwBuildNumber**. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **BuildMin**.
- **BuildMax:** *build_number*
Numeric value returned by **OSVERSIONINFO.dwBuildNumber**. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **BuildMax**.

Example

This shows three **[CEDevice]** sections: one giving basic information for any CPU and two specific to the SH3 and the MIPS microprocessors.

[CEDevice] ; A "template" for all platforms

UnsupportedPlatforms = pltfrm1; Does not support *pltfrm1*

; The following specifies version 1.0 devices only.

VersionMin = 1.0

VersionMax = 1.0

[CEDevice.ARM]; Inherits all **[CEDevice]** settings

; This will create a .CAB file specific to ARM devices.

ProcessorType = 2577; ARM .cab file is valid for ARM microprocessors.

UnsupportedPlatforms = ; *pltfrm1* is still unsupported

; The following overrides the version settings so that no version checking is performed.

VersionMin =

VersionMax =

[CEDevice.MIPS] ; Inherits all **[CEDevice]** settings

; This will create a .CAB file specific to "MIPS" devices.

ProcessorType = 4000; MIPS .CAB file is valid for MIPS microprocessor.

UnsupportedPlatforms =pltfrm2; *pltfrm1*, *pltfrm2* unsupported for MIPS .CAB file.



Note: To create the two CPU-specific .CAB files for the SETUP.INF file in the previous example, run the CAB Wizard with the “/cpu arm mips” parameter.

[DefaultInstall]

This describes the default installation of your application. Note that under this section, you will list items expanded upon later in this description.

Required? Yes

- **Copyfiles:** *copyfile_list_section*
Maps to files defined later in the .INF file, such as Files.App, Files.Font, and Files.Bitmaps.
- **AddReg:** *add_registry_section*
Example: RegSettings.All
- **CEShortcuts:** *shortcut_list_section*
String that identifies one more section that defines shortcuts to a file, as defined in the [CEShortcuts] section.
- **CESetupDLL:** *setup_DLL*
Optimal string that specifies a SETUP.DLL file. It is written by the Independent Software Vendor (ISV) and contains customized functions for operations during installation and removal of the application. The file must be specified in the [SourceDisksFiles] section.
- **CESelfRegister:** *self_reg_DLL_filename*
String that identifies files that self-register by exporting the **DllRegisterServer** and **DllUnregisterServer** Component Object Model (COM) functions. Specify these files in the [SourceDiskFiles] section. During installation, if installation on the device fails to call the file's exported **DllRegisterServer** function, the file's exported **DllUnregisterServer** function is not called during removal.

Example

[DefaultInstall]

```
AddReg = RegSettings.All
CEShortcuts = Shortcuts.All
```

[SourceDiskNames]

This section describes the name and path of the disk on which your application resides.

Required? Yes

- **disk_ordinal:** *disk_label,,path*
1=, “App files” , C:\Appsoft\RP32\...
2=, “Font files” , C:\RpTools\...
3=, “CE Tools” , C:\windows ce tools...
- **CESignature:** “\$Windows CE\$”

Example

```
[SourceDisksNames]; Required section
1 = , "Common files", , C:\app\common; Using an absolute path
[SourceDisksNames.SH3]
2 = , "SH3 files", , sh3; Using a relative path
[SourceDisksNames.MIPS]
2 = , "MIPS files", , mips; Using a relative path
```

[SourceDiskFiles]

This describes the name and path of the files in which your application resides.

Required? Yes

- **filename:** *disk_number[,subdir]*
RPM.EXE = 1,c:\appsoft\...
WCESTART.INI = 1
RPMCE212.INI = 1
TAHOMA.TTF = 2



Note: [,subdir] is relative to the location of the INF file.

Example

```
[SourceDisksFiles]; Required section
begin.wav = 1
end.wav = 1
sample.hlp = 1
[SourceDisksFiles.SH3]
sample.exe = 2; Uses the SourceDisksNames.SH3 identification of 2.
[SourceDisksFiles.MIPS]
sample.exe = 2; Uses the SourceDisksNames.MIPS identification of 2.
```

[DestinationDirs]

This describes the names and paths of the destination directories for the application on the target device. *Note Windows CE does not support directory identifiers.*

Required? Yes

- **file_list_section:** *0,subdir*
String that identifies the destination directory. The following list shows the string substitutions supported by Windows CE. Use these only for the beginning of the path. \

%CE1%	\Program Files
%CE2%	\Windows
%CE3%	\My Documents
%CE4%	\Windows\Startup
%CE5%	\My Documents
%CE6%	\Program Files\Accessories
%CE7%	\Program Files\Communication
%CE8%	\Program Files\Games

%CE9% \Program Files\Pocket Outlook
 %CE10% \Program Files\Office
 %CE11% \Windows\Start Menu\Programs
 %CE12% \Windows\Start Menu\Programs\Accessories
 %CE13% \Windows\Start Menu\Programs\Communications
 %CE14% \Windows\Start Menu\Programs\Games
 %CE15% \Windows\Fonts
 %CE16% \Windows\Recent
 %CE17% \Windows\Start Menu

%InstallDir%

Contains the path to the target directory selected during installation. It is declared in the [CEStrings] section

%AppName%

Contains the application name defined in the [CEStrings] section.

Example

[DestinationDirs]

```
Files.Common = 0,%CE1%\My Subdir; \Program Files\My Subdir
Files.Shared = 0,%CE2%; \Windows
```

[CopyFiles]

This section, under the [DefaultInstall] section, describes the default files to copy to the target device. Within the [DefaultInstall] section, files were listed that must be defined elsewhere in the INF file. This section identifies that mapping and may contain flags.

Required? Yes

- **copyfile_list_section:** *destination_filename,[source_filename]*
The *source_filename* parameter is optional if it is the same as *destination_filename*.
- **copyfile_list_section:** *flags*
The numeric value that specifies an action to do while copying files. The following table shows values supported by Windows CE.

Flag	Value	Description
COPYFLG_WARN_IF_SKIP	0x00000001	Warn user if skipping a file is attempted after error.
COPYFLG_NOSKIP	0x00000002	Do not allow a user to skip copying a file.
COPYFLG_NO_OVERWRITE	0x00000010	Do not overwrite files in destination directory.
COPYFLG_REPLACEONLY	0x00000400	Copy the source file to the destination directory only if the file is already in the destination directory.
CE_COPYFLG_NO_DATE_DIALOG	0x20000000	Do not copy files if the target file is newer.
CE_COPYFLG_NODATECHECK	0x40000000	Ignore date while overwriting the target file.
CE_COPYFLG_SHARED	0x80000000	Create a reference when a shared DLL is counted.

Example

```
[DefaultInstall.SH3]
CopyFiles = Files.Common, Files.SH3
[DefaultInstall.MIPS]
CopyFiles = Files.Common, Files.MIPS
```

[AddReg]

This section, under the [DefaultInstall] section, is optional and describes the keys and values that the .CAB file adds to the device registry. Within the [DefaultInstall] section, a reference may have been made to this section, such as “AddReg=RegSettings.All”. This section defines the options for that setting.

Required? No

- **add_registry_section:***registry_root_string*
String that specifies the registry root location. The following list shows the values supported by Windows CE.
 - HKCR Same as HKEY_CLASSES_ROOT
 - HKCU Same as HKEY_CURRENT_USER
 - HKLM Same as HKEY_LOCAL_MACHINE
- **add_registry_section:***value_name*
Registry value name. If empty, the “default” registry value name is used.
- **add_registry_section:***flags*
Numeric value that specifies information about the registry key. The following table shows the values that are supported by Window CE.

Flag	Value	Description
FLG_ADDREG_NOCLOBBER	0x00000002	If the registry key exists, do not overwrite it. Can be used with any of the other flags in this table.
FLG_ADDREG_TYPE_SZ	0x00000000	REG_SZ registry data type.
FLG_ADDREG_TYPE_MULTI_SZ	0x00010000	REG_MULTI_SZ registry data type. Value field that follows can be a list of strings separated by commas.
FLG_ADDREG_TYPE_BINARY	0x00000001	REG_BINARY registry data type. Value field that follows must be a list of numeric values separated by commas, one byte per field, and must not use the 0x hexadecimal prefix.
FLG_ADDREG_TYPE_DWORD	0x00010001	REG_DWORD data type. The noncompatible format in the Win32 Setup .INF documentation is supported.

Example

```
AddReg = RegSettings.All
```

[RegSettings.All]

```
HKLM,%reg_path%,0x00000000,alpha; <default> = "alpha"
HKLM,%reg_path%,test,0x00010001,3; Test = 3
HKLM,%reg_path%\new,another,0x00010001,6; New\another = 6
```

[CEShortcuts]

This section, a Windows CE-specific section under the **[DefaultInstall]** section, is optional and describes the shortcuts that the installation application creates on the device. Within the **[DefaultInstall]** section, a reference may have been made to this section, such as “Shortcuts.All”. This section defines the options for that setting.

Required? No

- **shortcut_list_section:***shortcut_filename*
String that identifies the shortcut name. It does not require the .LNK extension.
- **shortcut_list_section:***shortcut_type_flag*
Numeric value. Zero or empty represents a shortcut to a file; any non-zero numeric value represents a shortcut to a folder.
- **shortcut_list_section:***target_file_path*
String value that specifies the destination location. Use the target file name for a file, such as MyApp.exe, that must be defined in a file copy list. For a path, use a *file_list_section* name defined in the **[DestinationDirs]** section, such as *DefaultDestDir*, or the *%InstallDir%* string.
- **shortcut_list_section:***standard_destination_path*
Optional string value. A standard *%CEX%* path or *%InstallDir%*. If no value is specified, the *shortcut_list_section* name of the current section or the *DefaultDestDir* value from the **[DestinationDirs]** section is used.

Example

```
CEShortcuts = Shortcuts.All
[Shortcuts.All]
Sample App,0,sample.exe; Uses the path in DestinationDirs. Sample App,0,sam-
ple.exe,%InstallDir%; The path is explicitly specified.
```

Sample .INF File

```
[Version]; Required section
Signature = "$Windows NT$"
Provider = "Intermec Technologies Corporation"
CESignature = "$Windows CE$"

;[CEDevice]
;ProcessorType =

[DefaultInstall]; Required section
CopyFiles = Files.App, Files.Fonts, Files.BitMaps, Files.Intl, Files.Tele-
comNcsCE, Files.Windows, Files.Import, Files.Export, Files.Work, Files.Data-
base, Files.WinCE AddReg = RegSettings.All ;CEShortcuts = Shortcuts.All

[SourceDisksNames]; Required section
1 = , "App files" , , c:\appsoft\...
2 = , "Font files" , , c:\WinNT\Fonts
3 = , "CE Tools" , , c:\windows ce tools\wce400\700ie\mfc\lib\x86

[SourceDisksFiles]; Required section
rpm.exe = 1,C:\Appsoft\program\wce400\WCEX86Rel700
```

```

wcestart.ini = 1
rpmce212.ini = 1
intermec.bmp = 1
rpmlogo.bmp = 1
rpmname.bmp = 1
import.bmp = 1
export.bmp = 1
clock.bmp = 1
printer.bmp = 1
filecopy.bmp = 1
readme.txt = 1
lang_eng.bin = 1
rpmdata.dbd = 1,database\wce1
tahoma.ttf = 2
mfcce212.dll = 3
olece212.dll = 3
olece211.dll = 1,c:\windows ce tools\wce400\NMSD61102.11\mfc\lib\x86
rdm45wce.dll = 1,c:\rptools\rdm45wce\4_50\lib\wce400\wcex86rel
picfmt.dll = 1,c:\rptools\picfmt\1_00\wce400\wcex86rel6110
fmtctrl.dll = 1,c:\rptools\fmtctrl\1_00\wce400\wcex86rel6110
ugrid.dll = 1,c:\rptools\ugrid\1_00\wce400\wcex86rel6110
simple.dll = 1,c:\rptools\pspbm0c\1_00\wce400\wcex86rel
psink.dll = 1,c:\rptools\psink\1_00\wce400\WCEX86RelMinDependency
pslpwce.dll = 1,c:\rptools\pslpm0c\1_00\wce400\WCEX86RelMinDependency
npcpport.dll = 1,c:\rptools\cedk\212_03\installable drivers\printer\npcp
;dexcom.dll = 1,c:\rptools\psdxm0c\1_00\x86
ncsce.exe = 1,c:\rptools\ncsce\1_04
nrinet.dll = 1,c:\rptools\ncsce\1_04

```

```

[DestinationDirs]; Required section
;Shortcuts.All = 0,%CE3% ; \Windows\Desktop
Files.App= 0,%InstallDir%
Files.DataBase= 0,%InstallDir%\DataBase
Files.BitMaps= 0,%InstallDir%\Bitmaps
Files.Fonts= 0,%InstallDir%\Fonts
Files.Intl= 0,%InstallDir%\Intl
Files.TelecomNcsCE= 0,%InstallDir%\Telecom\NcsCE
Files.Windows= 0,%InstallDir%\Windows
Files.Import= 0,%InstallDir%\Import
Files.Export= 0,%InstallDir%\Export
Files.Work= 0,%InstallDir%\Work
Files.WinCE= 0,\storage_card\wince

```

```

[CEStrings]; Required section
AppName = Rp32
InstallDir = \storage_card\%AppName%

```

```

[Strings]; Optional section
;[Shortcuts.All]
;Sample App,0,sample.exe; Uses the path in DestinationDirs.
;Sample App,0,sample.exe,%InstallDir%; The path is explicitly specified.

```

```

[Files.App]
rpm.exe,,0
rpm.ini,rpmce212.ini,,0
mfcce212.dll,,0
olece212.dll,,0
olece211.dll,,0
rdm45wce.dll,,0

```

```

picfmt.dll,,,0
fmtctrl.dll,,,0
ugrid.dll,,,0
simple.dll,,,0
psink.dll,,,0
pslpwce.dll,,,0
npcpport.dll,,,0
;dexcom.dll,,,0

[Files.DataBase]
rpmdata.dbd,,,0

[Files.Fonts]
tahoma.ttf,,,0

[Files.BitMaps]
intermec.bmp,,,0
rpmlogo.bmp,,,0
rpmname.bmp,,,0
import.bmp,,,0
export.bmp,,,0
clock.bmp,,,0
printer.bmp,,,0
filecopy.bmp,,,0

[Files.Intl]
lang_eng.bin,,,0

[Files.TelecomNcsCE]
ncsce.exe,,,0
nrinet.dll,,,0

[Files.Windows]
readme.txt,,,0

[Files.Import]
readme.txt,,,0

[Files.Export]
readme.txt,,,0

[Files.Work]
readme.txt,,,0

[Files.WinCE]
wcestart.ini,,,0

[RegSettings.All]
HKLM,"SOFTWARE\Microsoft\Shell\AutoHide",,0x00010001,1; Autohide the taskbar
HKLM,"SOFTWARE\Microsoft\Shell\OnTop",,0x00010001,0; Shell is not on top
HKLM,"SOFTWARE\Microsoft\Clock",SHOW_CLOCK,0x00010001,0
; Clock is not on taskbar

```

Using Installation Functions in SETUP.DLL

SETUP.DLL is an optional file that enables you to perform custom operations during installation and removal of your application. The following list shows the functions that are exported by SETUP.DLL.



Note: Use **[DefaultInstall] > CESSelfRegister** (page 129) in the .INF file to point to SETUP.DLL.

Install_Init	Called before installation begins. Use this function to check the application version when reinstalling an application and to determine if a dependent application is present.
Install_Exit	Called after installation is complete. Use this function to handle errors that occur during application installation.
Uninstall_Init	Called before the removal process begins. Use this function to close the application, if the application is running.
Uninstall_Exit	Called after the removal process is complete. Use this function to save database information to a file and delete the database and to tell the user where the user data files are stored and how to reinstall the application.

After the CAB File Extraction

Cab files that need to cause a warm reset after cab extraction need to create the `__RESETMEPLEASE__.TXT` file in the “\Windows” directory. The preferred method to create this file is within the `DllMain` portion of the `SETUP.DLL` file. It looks like this:

```
#include <windows.h>
#include <Tlhelp32.h>
#include <winioctl.h>
#include <ce_setup.h>    // in the public SDK dir

#define IOCTL_TERMINAL_RESET CTL_CODE (FILE_DEVICE_UNKNOWN, FILE_ANY_ACCESS,
2050, METHOD_NEITHER)

BOOL APIENTRY DllMain( HANDLE h, DWORD reason, LPVOID lpReserved )
{
return TRUE;
} // DllMain

/*****
// $DOCBEGIN$
// BOOL IsProcessRunning( TCHAR * pname );
//
// Description: Get process table snapshot, look for pname running.
//
// Arguments: pname - pointer to name of program to look for.
// for example, app.exe.
//
// Returns: TRUE - process is running.
//          FALSE - process is not running.
// $DOCEND$
*****/
```

```

BOOL IsProcessRunning( TCHAR * pname )
{
HANDLE hProcList;
PROCESSENTRY32 peProcess;
DWORD thDeviceProcessID;
TCHAR lpname[MAX_PATH];

if ( !pname || !*pname ) return FALSE;

_tcscpy( lpname, pname );
_tcslwr( lpname );

hProcList = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );

if ( hProcList == INVALID_HANDLE_VALUE ) {
return FALSE;
} // end if

memset( &peProcess, 0, sizeof(peProcess) );
peProcess.dwSize = sizeof(peProcess);

if ( !Process32First( hProcList, &peProcess ) ) {
CloseToolhelp32Snapshot( hProcList );
return FALSE;
} // end if

thDeviceProcessID = 0;

do {
_tcslwr( peProcess.szExeFile );

if ( _tcscstr( peProcess.szExeFile, lpname ) ) {
thDeviceProcessID = peProcess.th32ProcessID;
break;
} // end if
} while ( Process32Next( hProcList, &peProcess ) );

if ( ( GetLastError() == ERROR_NO_MORE_FILES ) && ( thDeviceProcessID == 0 ) )
{
CloseToolhelp32Snapshot( hProcList );
return FALSE;
} // end if

CloseToolhelp32Snapshot( hProcList );
return TRUE;
} // IsProcessRunning

codeINSTALL_INIT Install_Init(
HWND hwndParent,
BOOL fFirstCall,
BOOL fPreviouslyInstalled,
LPCTSTR pszInstallDir )
{
return codeINSTALL_INIT_CONTINUE;
}

codeINSTALL_EXIT Install_Exit (
HWND hwndParent,
LPCTSTR pszInstallDir,

```

```

WORD cFailedDirs,
WORD cFailedFiles,
WORD cFailedRegKeys,
WORD cFailedRegVals,
WORD cFailedShortcuts )
{
HANDLE h;
TCHAR srcfile[MAX_PATH];
TCHAR dstfile[MAX_PATH];

if (cFailedDirs || cFailedFiles || cFailedRegKeys ||
cFailedRegVals || cFailedShortcuts)
return codeINSTALL_EXIT_UNINSTALL;

if ( IsProcessRunning( L"autocab.exe" ) )
{
h = CreateFile( L"\\Windows\\__resetmeplease__.txt",
(GENERIC_READ | GENERIC_WRITE), 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_HIDDEN, NULL );

if ( h != INVALID_HANDLE_VALUE )
CloseHandle( h );
else
{
// Couldn't create the file. If it failed because the file already exists, it
is not fatal.
// Otherwise, notify user of the inability to reset the device and they will
have to
// perform it manually after all of the installations are complete.
} // end if
}
else
{
DWORD dret;

h = CreateFile( L"SYI1:",
(GENERIC_WRITE | GENERIC_READ), 0, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL );

// Force a warm start NOW.
if ( h != INVALID_HANDLE_VALUE )
{
DeviceIoControl( h, IOCTL_TERMINAL_RESET, NULL, 0, NULL, 0, &dret, NULL);

// Won't return, but we'll show clean up anyway
CloseHandle( h );
}
else
{
// Couldn't access SYSIO. Notify user.
} // end if
} // end if

return codeINSTALL_EXIT_DONE;
}

codeUNINSTALL_INIT
Uninstall_Init(
HWND hwndParent,

```

```

LPCTSTR pszInstallDir ) {

// TODO: Perform the reverse of INSTALL_INIT here
return codeUNINSTALL_INIT_CONTINUE;
}

codeUNINSTALL_EXIT
Uninstall_Exit(HWND hwndParent) {

// TODO: Perform the reverse of INSTALL_EXIT here
return codeUNINSTALL_EXIT_DONE;
}

```

The system software looks for the following directory structure and files on the installed media card whether it be a Secure Digital card or embedded flash file system. No other folders need exist.

```

\2577\autorun.exe
\2577\autorun.dat
\2577\autocab.exe
\2577\autocab.dat
\cabfiles\*.cab

```

Creating CAB Files with CAB Wizard

After you create the .INF file and the optional SETUP.DLL file, use the CAB Wizard to create the .CAB file. The command-line syntax for the CAB Wizard is as follows:

```
cabwiz.exe "inf_file" [/dest dest_directory] [/err error_file] [/cpu cpu_type
[cpu_type]]
```

A batch file, located in <program> directory, with the following commands, works well:

```
cabwiz.exe c:\appsoft\<program>\<inf_file_name>
cd \appsoft\<program>
```

"inf_file"	The SETUP.INF file path.
dest_directory	The destination directory for the .CAB files. If no directory is specified, the .CAB files are created in the "inf_file" directory.
error_file	The file name for a log file that contains all warnings and errors that are encountered when the .CAB files are compiled. If no file name is specified, errors are displayed in message boxes. If a file name is used, the CAB Wizard runs without the user interface (UI); this is useful for automated builds.
cpu_type	Creates a .CAB file for each specified microprocessor tag, which is a label used in the Win32 SETUP.INF file to differentiate between different microprocessor types. The /cpu parameter, followed by multiple <i>cpu_type</i> values, must be the last qualifier in the command line.

Example

This example creates .CAB files for the ARM and MIPS microprocessors, assuming the Win32 SETUP.INF file contains the ARM and MIPS tags:

```
cabwiz.exe "c:\myfile.inf" /err myfile.err /cpu arm mips
```



Note: CABWIZ.EXE, MAKECAB.EXE, and CABWIZ.DDF (Windows CE files available on the Windows CE Toolkit) must be installed in the same directory on the desktop computer. Call CABWIZ.EXE using its full path for the CAB Wizard application to run correctly.

Troubleshooting the CAB Wizard

To identify and avoid problems that might occur when using the CAB Wizard, follow these guidelines:

- Use %% for a percent sign (%) character when using this character in an .INF file string, as specified in Win32 documentation. This does not work under the [Strings] section.
- Do not use .INF or .CAB files created for Windows CE to install applications on Windows-based desktop platforms.
- Ensure the MAKECAB.EXE and CABWIZ.DDF files, included with Windows CE, are in the same directory as CABWIZ.EXE.
- Use the full path to call CABWIZ.EXE.
- Do not create a .CAB file with the MAKECAB.EXE file included with Windows CE. You must use CABWIZ.EXE, which uses MAKECAB.EXE to generate the .CAB files for Windows CE.
- Do *not* set the read-only attribute for .CAB files.

Customization and Lockdown

Pocket PC (Windows Mobile) is a hardware specification created by Microsoft Corporation. Devices that wish to carry the Pocket PC logo must meet the minimum hardware requirements set in the Pocket PC specification. Manufacturers are free to add extra hardware functionality.

Pocket PC devices also use a specialized version of the CE operating system. This operating system is built from Windows CE 4.2 but contains customizations, most notably the lack of a desktop and the addition of the Today Screen.

To carry the Pocket PC logo, all devices must be tested at an Independent Test Laboratory. The ITL testing is done based on Microsoft requirements. The test lab then reports the findings back to Microsoft Corporation and Intermec Technologies. If the CN2B Computer passed all tests, Intermec is allowed to ship the device with the Pocket PC logo. Each time the operating system is modified, Intermec must resubmit to ITL testing.

This means we cannot change the operating system much and still be a Pocket PC device. For example, if we remove Word from the Start menu, the device would fail ITL testing and we would not be able to ship devices with the Pocket PC logo.

Although many customers want a Pocket PC device, some customers would prefer that their users not have access to all of the Pocket PC features. Intermec cannot customize the operating system in any way but a custom application can:

- Delete items from the Start menu and Programs folder. These items are just shortcuts in the file system so the application is not really being deleted. Cold booting the device brings these items back so the application will need to be run on every cold boot.
- Use the RegFlushKey() API to save a copy of the registry to a storage device. See the CN2B Management Tools portion for more information on how to do this. Saving a copy of the registry restores most system settings in a cold boot situation.
- Use the SHFullScreen() API in conjunction with other APIs to make the application take up the entire display and prevent the start menu from being available.
- Remap keys and disable keys on the keypad.
- Create a custom SIP.
- Make changes to the registry to configure the device.

Should you want your CN2B Computer to display a full screen, keep in mind that your computer is Pocket-PC certified by Microsoft Corporation. Check out resources on programming for the Pocket PC, using the following links. These give full instructions on how to display full screen.

- Instructions on how to create a full screen application for eVC++ applications using an SHFullScreen() API:
support.microsoft.com/support/kb/articles/Q266/2/44.ASP
- Instructions on how to create a full screen application for eVB applications also using the SHFullScreen() API:
support.microsoft.com/support/kb/articles/Q265/4/51.ASP

FTP Server

FTP support is provided through the FTP Server application FTPDCE.EXE (MS Windows CE Versions) which is provided as part the base system.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests the FTP Server also send a “network announcement” to notify prospective clients of server availability.



Note: You should consult the RFC959 specification for proper use of some of these commands at the following URL:

- www.ietf.org/rfc/rfc959.txt for the text version, or
- www.w3.org/Protocols/rfc959/ for an html version.

Do the following to send commands:

- 1 Start an FTP client and connect to the device FTP server.
- 2 Log in with “intermec” as the user name and “cr52401” for the password.
- 3 From the FTP client, send the command.
- 4 Wait for a response.

Synopsis

ftpdce [*options*]

Options

–Aaddr	(where <i>addr</i> is in the form of a.b.c.d) Sets the single target address to which to send the network announcement. <i>Default is broadcast.</i>
–Bbyte	Sets the FTP data block size. Smaller sizes may be useful over slower links. <i>Default is 65536.</i>
–Cname	Sets the device name. Used by Intermec management software.
–Fvalue	Disables the default Intermec account. A value of “0” disables the account. <i>Default is “1”.</i> Note that disabling the default account without providing a working access control list on the server will result in a device that will not accept any FTP connections.
–Hsec	Sets the interval between network announcements in seconds. “0” turns the network announcement off. <i>Default is 30 seconds.</i>
–Iaddr	(where <i>addr</i> is in the form of a.b.c.d) Sets the preferred 6920 Communications Server (<i>optional</i>).
–Llog	(where <i>log</i> is either “0” or “1”) Sets the state of logging. <i>Default is 0 (disabled).</i>
–Nsec	Specifies the number of seconds to wait before initially starting FTP server services.
–Pport	Sets the UDP port on which the network announcement are sent. <i>Default port is 52401.</i>
–Qport	Sets the port on which the FTP Server will listen for connections. <i>Default port is 21.</i>
–Rdir	Sets the FTP mount point to this directory. Default is the root folder of the object store.
–Tscrip	Sets the script name for the 6920 Communications Server to process.
–Uurl	Sets the default URL for this device.
–Z“parms”	Sets extended parameters to be included in the network announcement.

Configurable Parameters Via the Registry Editor

The following parameters receive default values during the installation of the Intermec FTP Server components. A few of the parameters are visible in the registry by default, but most must be created to modify the default behavior of the FTP server.

BlockSize

Setting this parameter configures the Intermec FTP Server to transmit and receive Ethernet packets using the specified data block size. By default, the FTP server transmits and receives data using a 64K data block size. Adjusting this value may be useful in certain wireless TCP/IP installations.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_DWORD - data block size, in bytes.
Valid Range	0x100-0x10000 (256-65536 decimal).
Default	65536

DeviceName

This parameter configures the Intermec FTP Server to include the specified device name in the Intermec Device Network Announcement (IDNA). Adjusting this value may be useful in assigning a symbolic name to this device for asset tracking.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_SZ
Valid Range	None.
Default	None.

DeviceURL

This parameter configures the Intermec FTP Server to transmit the specified URL in the IDNA. This can be used by Intermec management software for asset management.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_SZ
Valid Range	None.
Default	None.

IDNATarget

This parameter configures the Intermec FTP Server to transmit the IDNA to a specific destination instead of a general UDP broadcast. This parameter is useful on networks that do not allow UDP broadcasts to be routed between subnets. The use of this parameter restricts the reception of the IDNA to the target destination only.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_SZ
Valid Range	None.
Default	None.

ManifestName

This parameter configures the Intermec FTP Server to transmit the specified manifest name in the IDNA. This parameter is used by the Intermec 6920 Communications Server for communication transactions. See 6920 Communications Server documentation for proper use of this parameter.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_SZ
Valid Range	None.
Default	iftp.ini

PauseAtStartup

This configures the Intermec FTP Server to sleep for the specified number of seconds before making the FTP service available on the device.

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_DWORD - stored in seconds.
Valid Range	None.
Default	0

Root

This parameter configures the Intermec FTP Server to set the root of the FTP mount point to the specified value. *Note that this must map to an existing directory or you will not be able to log into the FTP Server.*

Key	HKLM\Software\Intermec\IFTP
Value Type	REG_SZ
Valid Range	None.
Default	\

Transferring Files Over TCP/IP Networks

The File Transfer Protocol (FTP) server transfers files over TCP/IP networks. The FTPDCE.EXE program is a version that does not display a window, but can run in the background.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests, the FTP Server also sends a “network announcement” to notify prospective clients of server availability.

The FTP Server currently supports the following FTP requests:

CDUP	Changes to the parent directory of the current working directory.
CWD	Changes working directory.
DELE	Deletes a file.
HELP	Gives help information.
LIST	(This FTP request is the same as the <code>ls -lgA</code> command). Gives list files in a directory.
MKD	Makes a directory.
MODE	<i>(Always Uses Binary)</i> . Specifies data transfer mode.
NLST	<i>(Not supported)</i> Gives a name list of files in directory (this request is the same as the <code>ls</code> command).
NOOP	Does nothing.
PASS	Specifies a password.
PWD	Prints the current working directory.
QUIT	Terminates session.
RETR	Retrieves a file.
RMD	Removes a directory.
RNFR	Specifies rename-from file name.
RNTO	Specifies rename-to file name.
STOR	Stores a file.
SYST	Shows the operating system type of server system.
TYPE	<i>(Binary transfers only.)</i> Specifies the data transfer type with the Type parameter.
USER	Specifies user name.
XCUP	<i>(Not Normally Used)</i> Changes the parent directory of the current working directory.
XCWD	<i>(Not Normally Used)</i> Changes the current directory.
XMKD	<i>(Not Normally Used)</i> Creates a directory.
XPWD	<i>(Not Normally Used)</i> Prints the current working directory.
XRMD	<i>(Not Normally Used)</i> Removes a directory.

SITE	The following extended OEM commands are supported by the SITE request. For Microsoft FTP clients, you can send site commands by preceding the command with “quote” such as “quote site status.”
ATTRIB	Gets or sets the attributes of a given file. (SITE ATTRIB)
	Usage: QUOTE SITE ATTRIB [+R -R] [+A -A] [+S -S] [+H -H] [[<i>path</i>] <i>filename</i>] + Sets an attribute. - Clears an attribute. R Read-only file attribute. A Archive file attribute. S System file attribute. H Hidden file attribute. To retrieve the attributes of a file, only specify the file. The server response will be: <i>200-AD SHRCEIX filename</i> To retrieve the attributes of a file, only specify the file. The server response will be: <i>200-AD SHRCEIX filename</i> If the flag exists in its position shown above, it is set. Also, in addition to the values defined above, there is also defined: C Compressed file attribute. E Encrypted file attribute. I INROM file attribute. X XIP file attribute (execute in ROM, not shadowed in RAM).
BOOT	Reboots the server OS. This will cause the system on which the server is executing to reboot. The FTP Server will shut down cleanly before reboot. All client connections will be terminated. Cold boot is default except for the PocketPC build in which the default is warm boot. (SITE BOOT)
	Usage: QUOTE SITE BOOT [WARM COLD]
COPY	Copies a file from one location to another. (SITE COPY)
	Usage: QUOTE SITE COPY [<i>source</i>] [<i>destination</i>]
	Example: QUOTE SITE COPY '\\Storage Card\one.dat' \\Storage Card\two.dat'
EXIT	Exits the FTP Server. This command will shut down the FTP Server thus terminating all client connections. (SITE EXIT)
	Usage: QUOTE SITE EXIT
HELP	Gives site command help information. (SITE HELP)
	Usage: QUOTE SITE HELP [<i>command</i>]
KILL	Terminates a running program. (SITE KILL)
	Usage: QUOTE SITE KILL [<i>program</i> <i>pid</i>]

LOG	Opens or closes the program log. (SITE LOG) Usage: QUOTE SITE LOG [<i>open [filename] close</i>]
PLIST	Lists the running processes (SITE PLIST) Usage: QUOTE SITE PLIST
RUN	Starts a program running. If the program to run has spaces in path or filename, wrapping the name with single quotes is required. Usage: QUOTE SITE RUN [<i>program</i>] Example: QUOTE SITE RUN '\Storage Card\app.exe'
STATUS	Returns the current settings of the FTP Server. MAC, serial number, model, IP address, network announcement information as well as OS memory usage are returned. (SITE STATUS) Usage: QUOTE SITE STATUS
TIMEOUT	Toggles idle timeout between 120 to 1200 seconds (2 to 20 minutes). If this timer expires with no activity between the client and the server, the client connection will be disconnected. If the optional seconds argument is supplied, the server will set the connection timeout to the number of seconds specified. <i>Default is 120 seconds or 2 minutes.</i> (SITE TIMEOUT) Usage: QUOTE SITE TIMEOUT [<i>seconds</i>]
EKEY	Gives site command electronic key information. (SITE HELP) Usage: QUOTE SITE EKEY [<i>command</i>]
EVAL	Gives site command electronic value information. (SITE HELP) Usage: QUOTE SITE EVAL [<i>command</i>]
GVAL	Gives site command general value information. (SITE HELP) Usage: QUOTE SITE GVAL [<i>command</i>]
PVAL	Gives site command value information. (SITE HELP) Usage: QUOTE SITE PVAL [<i>command</i>]

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The banner returned in the parenthetical portion of its greeting shows the version number of the FTP Server as well as the MAC address, serial number and operating system of the machine hosting the server.

The FTP Server supports browsing from the latest Netscape and Microsoft web browsers. Drag-and-drop capability is available using this environment.

The FTPDCMDS subdirectory contains commands to use from the web browser.

- Click EXITME.BIN to execute a SITE EXIT command.
- Click REBOOTME.BIN to execute SITE BOOT command.
- Use the GET command on these files to have the FTP Server execute these commands.

- **Security:**

A customer configurable access control list may be installed on the CN2B Computer. This list will allow customers to restrict access via the FTP Server to users they wish and is in addition to default Inter-mec accounts that are disabled using the *-F0* option at runtime.

The access control list is named FTPDCE.TXT and is placed in the same directory on the CN2B Computer as the FTPDCE.EXE server. The FTP Server encrypts this file to keep the information safe from unauthorized users. This file is encrypted when the FTP Server is started so a file that is placed onto the CN2B Computer after the FTP Server starts will require a restart of the FTP Server to take effect.

The format of the FTPDCE.TXT is as follows:

```
FTPDCE:user1!passwd1<cr><lf>user2!passwd2<cr><lf>user3!pas  
swd3<cr><lf>...
```



Note: The user accounts and passwords are case-sensitive. Once the access control list is encrypted on the CN2B Computer, the FTP Server hides this file from users. Once an access control list is installed on the CN2B Computer, a new one is not accepted by the FTP Server until the previous one is removed. Encrypted access control lists are not portable between CN2B Computers.

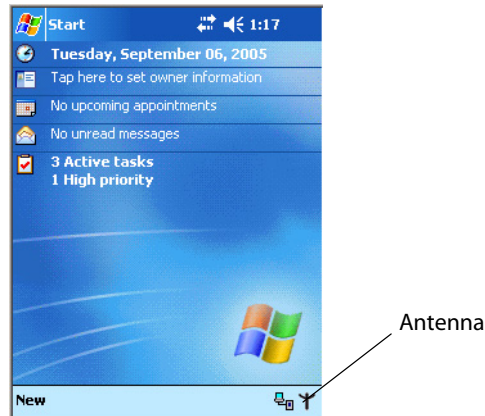
Stopping the FTP Server from Your Application

To allow application programmers the ability to programmatically shut down the FTP Server, the FTP Server periodically tests to see if a named event is signaled. The name for this event is "ITC_IFTP_STOP" (no quotes).

For examples on how to use events, consult the Microsoft Developer Network Library at www.msdn.com. The MSDN Library is an essential resource for developers using Microsoft tools, products, and technologies. It contains a bounty of technical programming information, including sample code, documentation, technical articles, and reference guides.

Autostart FTP

This automatically starts the FTP Server (FTPDCE.EXE) when the CN2B Computer is powered on. This is provided with the NDISTRAY program (the Network Driver Interface Specification tray application), which displays the popup menu that currently allows you to load and unload the network drivers. Tap the antenna icon in the System Tray of the Today screen (a sample antenna icon is shown below) for this pop-up menu.



The default is to start the FTP Server at boot time, unless the following registry entry is defined and set to “0” which disables AutoFTP. “1” enables the AutoFTP. The entry can be set from the NDISTRAY pop-up menu by selecting either **AutoFTP On** or **AutoFTP Off**.

HKEY_LOCAL_MACHINE\Software\Intermec\Ndistray\StartupIFTP

These new entries are located below the selections to load the network drivers. If the StartupIFTP registry key is not defined, the FTP Server is loaded by default, to provide “out-of-the-box” capability for customers who want to begin loading files to the CN2B Computer without prior configuration.

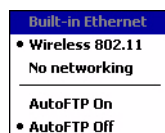


Note: If a network driver is unloaded using the NDISTRAY pop-up menu, and the FTP Server is running, the FTP Server is stopped.

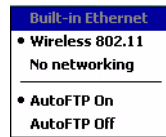
On a resume, if AutoFTP is enabled and the FTP Server is running, it is stopped and restarted. NDISTRAY uses a helper application named RESETIFTP to implement the restart on resume feature.

To do an AutoFTP Installation Check:

- 1 Ensure the FTP Server is running “out-of-the-box” the first time.
- 2 Tap **Start** > **Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP Off** to disable AutoFTP. Perform a warm-boot and confirm the FTP Server is not running.



- 3 Tap **Start > Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP On** to enable AutoFTP, reboot, confirm it is running.



- 4 Unload the network driver when the FTP Server is running and confirm that it is not running any more.
- 5 Load the FTP Server, establish a connection, then suspend and resume. The server should still run, but the FTP connection to the client should be dropped.

Kernel I/O Controls

This describes the `KernelIoControl()` functions available to application programmers. Most C++ applications need to prototype the function as the following to avoid link and compile errors.

```
extern "C" BOOL KernelIoControl(DWORD dwIoControlCode, LPVOID lpInBuf, DWORD
nInBufSize, LPVOID lpOutBuf, DWORD nOutBufSize, LPDWORD lpBytesReturned);
```

IOCTL_HAL_GET_DEVICE_INFO

This IOCTL returns either the platform type or the OEMPLATFORM name based on an input value.

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICE_INFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

<code>lpInBuf</code>	Points to a DWORD containing either the <code>SPI_GETPLATFORMTYPE</code> or <code>SPI_GETOEMINFO</code> value.
<code>lpInBufSize</code>	Must be set to <code>sizeof(DWORD)</code> .
<code>lpOutBuf</code>	Must point to a buffer large enough to hold the return data of the function. If <code>SPI_GETPLATFORMTYPE</code> is specified in <code>lpInBuf</code> , then the "PocketPC\0" Unicode string is returned. If <code>SPI_GETOEMINFO</code> is specified in <code>lpInBuf</code> , then the "Intermec 700\0" Unicode string is returned.
<code>nOutBufSize</code>	The size of <code>lpOutBuf</code> in bytes. Must be large enough to hold the string returned.
<code>lpBytesReturned</code>	The actual number of bytes returned by the function for the data requested.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. `GetLastError()` may be used to get the extended error value.

IOCTL_HAL_ITC_READ_PARM

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_ITC_READ_PARM, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Points to this structure. See “ID Field Values” below. <pre>struct PARMS { BYTE id; BYTE ClassId; };</pre>
nInBufSize	Must be set to the size of the PARMS structure.
lpOutBuf	Must point to a buffer large enough to hold the return data of the function. If this field is set to NULL and <i>nOutBufSize</i> is set to zero when the function is called the function will return the number bytes required by the buffer.
nOutBufSize	The size of <i>lpOutBuf</i> in bytes.
lpBytesReturned	Number of bytes returned by the function for the data requested.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the error value. Either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER may be returned when this function is used to get the error.

ID Field Values

The *id* field of the PARMS structure may be one of the following values:

ID Field Values
ITC_NVPARAM_SERIAL_NUM This IOCTL returns the serial number of the device in BCD format. Six bytes are returned in the buffer pointed to by the <i>lpOutBuffer</i> parameter.
ITC_NVPARAM_MANF_DATE This IOCTL returns the device date of manufacture in the BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the <i>lpOutBuffer</i> parameter.
ITC_NVPARAM_SERVICE_DATE This IOCTL returns the device's date of last service in BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the <i>lpOutBuffer</i> parameter.
ITC_NVPARAM_DISPLAY_TYPE This returns the device's display type. One byte is returned in the buffer pointed to by the <i>lpOutBuffer</i> parameter.

ID Field Values (continued)**ITC_NVPARM_EC�**

This IOCTL returns EC�s applied to the device in a bit array format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.

ITC_NVPARM_CONTRAST

This IOCTL returns the device default contrast setting. Two bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.

ITC_NVPARM_MCODE

This IOCTL returns the manufacturing configuration code for the device. Sixteen bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.

ITC_NVPARM_VERSION_NUMBER

This IOCTL returns the firmware version for various system components. These values for the *ClassId* field of the PARMS structure are allowed when ITC_NVPARM_VERSION_NUMBER is used in the *id* field:

- **VN_CLASS_KBD** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the keypad microprocessor version in the system. Format of the string is *x.xx* with a terminating null character.
- **VN_CLASS_ASIC** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the FPGA firmware in the system. Format of the string is *x.xx* with a terminating null character.
- **VN_CLASS_BOOTSTRAP** Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the Bootstrap Loader firmware in the system. Format of the string is *x.xx* with a terminating null character.

ITC_NVPARM_INTERMEC_SOFTWARE_CONTENT

This IOCTL reads the manufacturing flag bits from the nonvolatile data store that dictates certain software parameters. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates if Intermec Content is enabled in the XIP regions. TRUE indicates that it is enabled. FALSE indicates that it is not enabled.

ITC_NVPARM_WAN_RI

This reads the state of the WAN ring indicator flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates the polarity of the WAN RI signal. TRUE indicates active high. FALSE indicates active low.

ITC_NVPARM_INTERMEC_DATACOLLECTION_SW

This IOCTL reads the state of the data collection software enabled flag. A BOOLEAN DWORD is returned in the buffer pointer to by *lpOutBuffer* that indicates the data collection software is to install at boot time. FALSE indicates the data collection software should not install.

ITC_NVPARM_INTERMEC_DATACOLLECTION_HW

This IOCTL reads the data collection hardware flags. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of data collection hardware installed. The maximum possible value returned is ITC_DEVID_SCANHW_MAX.

- **ITC_DEVID_SCANHW_NONE** No scanner hardware is installed.

- **ITC_DEVID_INTERMEC_EVIO** EVIO linear imager is installed.

The high bit indicates whether the S6 scanning engine is installed. The bit mask for this is ITC_DEVID_S6ENGINE_MASK. A nonzero value indicates that the S6 scanning engine is installed.

ITC_NVPARM_80211_INSTALLED

This IOCTL reads the state of the 802.11b/g radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the 802.11b/g radio is installed. FALSE indicates that no 802.11b/g radio is installed.

ITC_NVPARM_80211_RADIO_TYPE

This IOCTL reads the 802.11b/g radio ID installed by manufacturing. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of 802.11b/g radio hardware installed. The maximum possible value returned is ITC_DEVID_80211RADIO_MAX. The current definitions are:

- **ITC_DEVID_80211RADIO_NONE** No 802.11b/g radio installed.

- **ITC_DEVID_80211RADIO_INTEL_2011B** Intel 2011B radio installed.

ID Field Values (continued)**ITC_NVPARM_BLUETOOTH_INSTALLED**

This IOCTL reads the state of the Bluetooth radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the Bluetooth radio is installed. FALSE indicates that no Bluetooth radio is installed.

ITC_NVPARM_SERIAL2_INSTALLED

This IOCTL reads the state of the serial 2 (COM2) device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the serial 2 device is installed. FALSE indicates that no serial 2 device is installed.

ITC_NVPARM_SIM_PROTECT_HW_INSTALLED

This IOCTL reads the state of the SIM card protection hardware installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection hardware is installed. FALSE indicates that no SIM card protection hardware is installed.

ITC_NVPARM_SIM_PROTECT_SW_INSTALLED

This IOCTL reads the state of the SIM card protection software installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection software is installed. FALSE indicates that no SIM card protection software is installed.

ITC_NVPARM_SIM_PROTECT_SW_INSTALLED

This IOCTL reads the state of the SIM card protection software installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection software is installed. FALSE indicates that no SIM card protection software is installed.

IOCTL_HAL_ITC_WRITE_SYSPARM

Describes and enables the registry save location.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_ITC_WRITE_SYSPARM, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	A single byte that may be one of the <i>id</i> values. See the following “ID Field Values” table.
nInBufSize	Must be set to the size of the <i>lpInBuf</i> in bytes.
lpOutBuf	Must point to a buffer large enough to hold the data to be written to the nonvolatile data store.
nOutBufSize	The size of <i>lpOutBuf</i> in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may get the error value. When this function gets the error, either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER is returned.

ID Field Values

The *id* field of *lpInBuf* may be one of the following values:

ID Field Values

ITC_REGISTRY_SAVE_ENABLE

This function enables or disables the save registry to non-volatile media feature of the RegFlushKey() function. *lpOutBuf* must be set to zero (FALSE) if the feature is to be disabled or one (TRUE) if the feature is to be enabled.

ITC_WAKEUP_MASK

This IOCTL sets a bit mask that represents the mask for the five programmable wakeup keys. The I/O key is not a programmable wakeup key. By default it is always the system resume key and all other keys are set to disable key wakeup. A zero in a bit position masks the wakeup for that key. A one in a bit position enables wakeup for that key. *lpOutBuf* must point to a buffer that contains a byte value of a wakeup mask consisting of the OR'ed constants as defined in OEMIOCTL.H. Only the following keys are programmable as wakeup events.

```
#define SCANNER_TRIGGER1
#define SCANNER_LEFT2
#define SCANNER_RIGHT4
#define GOLD_A18
#define GOLD_A20x10
```

IOCTL_HAL_GET_DEVICEID

This IOCTL returns the device ID. There are two types of device IDs supported, which are differentiated based on the size of the *output* buffer. The UUID is returned if the buffer size is set to *sizeof(UNIQUE_DEVICEID)*, otherwise the oldstyle device ID is returned.

Usage

```
#include "pkfuncs.h"
#include "deviceid.h"
```

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICEID, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL. STRICT_ID settings are not supported.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a UNIQUE_DEVICEID structure as defined by DEVICEID.H if the UUID is to be returned.
nOutBufSize	The size of the UNIQUE_DEVICEID in bytes if the UUID is to be returned. A DEVICE_ID as defined by PKFUNCS.H is returned if the size in bytes is greater than or equal to <i>sizeof(DEVICE_ID)</i> .
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_OAL_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpBytesReturned	Returns <i>sizeof(PVERSIONINFO)</i> .
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values: • cboemverinfo <i>sizeof(tagOemVerInfo)</i>; • verinfover 1 • sig; "ITC\0" • id; 'N' • tgtcustomer "" • tgtplat SeaRay • tgtplatversion Current build version number • tgtcputype[8]; "Intel\0" • tgtcpu "PXA255\0"; • tgtcoreversion "" • date Build time • time Build date
nOutBufSize	The size of VERSIONINFO in bytes.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_BOOTLOADER_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
nInBufSize	Should be set to zero.
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values: • cboemverinfo Sizeof (tagOemVerInfo); • verinfover 1 • sig; "ITC\0" • id; 'B' • tgtcustomer "" • tgtplat SeaRay • tgtplatversion Current build version number of the bootstrap loader • tgtcputype[8]; "Intel\0"; • tgtcpu "PXA255\0" • tgtcoreversion "" • date Build time • time Build date
nOutBufSize	The size of VERSIONINFO in bytes.
lpBytesReturned	The number of bytes returned to <i>lpOutBuf</i> .

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_WARMBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_WARMBOOT, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_COLDBOOT

Causes the system to perform a cold-boot. The object store is cleared.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_COLDBOOT, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_GET_RESET_INFO

This code allows software to check the type of the most recent reset.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_RESET_INFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a HAL_RESET_INFO structure. See sample below.
nOutBufSize	The size of HAL_RESET_INFO in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. May use GetLastError() to get the extended error value.

Sample

```
typedef struct {
    DWORD ResetReason;           // most recent reset type
    DWORD ObjectStoreState;      // state of object store
} HAL_RESET_INFO, * PHAL_RESET_INFO;

// Reset reason types
#define HAL_RESET_TYPE_UNKNOWN 0
#define HAL_RESET_REASON_HARDWARE 1 // cold
#define HAL_RESET_REASON_SOFTWARE 2 // suspend
#define HAL_RESET_REASON_WATCHDOG 4
#define HAL_RESET_BATT_FAULT 8 // power fail
#define HAL_RESET_VDD_FAULT 16 // warm boot

// Object store state flags
#define HAL_OBJECT_STORE_STATE_UNKNOWN 0
#define HAL_OBJECT_STORE_STATE_CLEAR 1
```

IOCTL_HAL_GET_BOOT_DEVICE

This IOCTL code allows software to check which device CE booted from.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_BOOT_DEVICE, LPVOID
    lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
    nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a buffer large enough to hold a DWORD (4 bytes) that contains the boot device. These boot devices are supported: <pre>#define HAL_BOOT_DEVICE_UNKNOWN 0 #define HAL_BOOT_DEVICE_ROM_XIP 1 #define HAL_BOOT_DEVICE_ROM 2 #define HAL_BOOT_DEVICE_PCMCIA_ATA 3 #define HAL_BOOT_DEVICE_PCMCIA_LINEAR 4 #define HAL_BOOT_DEVICE_IDE_ATA 5 #define HAL_BOOT_DEVICE_IDE_ATAPI 6</pre>
nOutBufSize	The size of <i>lpOutBuf</i> in bytes (4).
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. May use GetLastError() to get the extended error value.

IOCTL_HAL_REBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_REBOOT, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_PROCESSOR_INFORMATION

Returns processor information.

Usage

#include "pkfuncs.h"

Syntax

```
BOOL KernelIoControl( IOCTL_PROCESSOR_INFORMATION, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
nInBufSize	Should be set to zero.
lpOutBuf	Should be a pointer to the PROCESSOR_INFO structure. The PROCESSOR_INFO structure stores information that describes the CPU more descriptively. <pre>typedef __PROCESSOR_INFO { WORD wVersion; // Set to value 1 WCHAR szProcessorCore[40]; // "ARM\0" WORD wCoreRevision; // 4 WCHAR szProcessorName[40]; // "PXA255\0" WORD wProcessorRevision; // 0 WCHAR szCatalogNumber[100]; // 0 WCHAR szVendor[100]; // "Intel Corporation\0" DWORD dwInstructionSet; // 0 DWORD dwClockSpeed; // 400 }</pre>
nOutBufSize	Should be set to sizeof(PROCESSOR_INFO) in bytes.
lpBytesReturned	Returns sizeof(PROCESSOR_INFO);

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. May use GetLastError() to get the extended error value.

IOCTL_GET_CPU_ID

Returns Xscale processor ID.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_GET_CPU_ID, LPVOID lpInBuf,
    DWORD nInBufSize, LPVOID lpOutBuf, DWORD nOutBufSize, LPDWORD
    lpBytesReturned );
```

Parameters

lpInBuf	Should point to a CPUIdInfo structure defined in OEMIOCTL.H.
lpInBufSize	Should be <i>sizeof(CPUIdInfo)</i> .
lpOutBuf	Should be NULL.
nOutBufSize	Should be set to 0.
lpBytesReturned	Returns <i>sizeof(PROCESSOR_INFO)</i> ;

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. May use GetLastError() to get the extended error value.

Network Selection APIs

The Network Selection APIs change the network adapter configuration programmatically. Both drivers support the same IOCTL function numbers for loading and unloading the drivers. Loading and unloading of the 802.11b/g driver is performed by the FWL1: device in the system by performing DeviceIOControl() calls to the driver. Loading and unloading of the driver for the built-in Ethernet adapter is performed by the SYI1: device in the system by performing DeviceIOControl() calls to the driver.

- For loading an NDIS driver associated with an adapter, the IOCTL is IOCTL_LOAD_NDIS_MINIPORT.
- For unloading NDIS drivers associated with an adapter the IOCTL is IOCTL_UNLOAD_NDIS_MINIPORT.

Example

```
#include <winioctl.h>
#include "sysio.h"
void DoLoad(int nDevice) {
    LPTSTR devs[] = { _T("SYI1:"), _T("FWL1:") };
    HANDLE hLoaderDev;
    DWORD bytesReturned;
    hLoaderDev = CreateFile(devs[nDevice], GENERIC_READ|GENERIC_WRITE, 0,
        NULL, OPEN_EXISTING, 0, NULL);
    if (hLoaderDev != INVALID_HANDLE_VALUE) {
        if (!DeviceIoControl(hLoaderDev, IOCTL_LOAD_NDIS_MINIPORT, NULL, -1,
            NULL, 0,
            &bytesReturned, NULL)){
            MessageBox(NULL, TEXT("SYSIO IoControl Failed"), TEXT("Network
                loader"), MB_ICONHAND);
            if (hLoaderDev != INVALID_HANDLE_VALUE) CloseHandle(hLoaderDev);
            hLoaderDev = INVALID_HANDLE_VALUE; // bad handle
        }else {
            CloseHandle(hLoaderDev);
        }
    }
}

void DoUnload(int nDevice) {
    LPTSTR devs[] = { _T("SYI1:"), _T("FWL1:") };
    HANDLE hLoaderDev;
    DWORD bytesReturned;
    hLoaderDev = CreateFile(devs[nDevice], GENERIC_READ|GENERIC_WRITE, 0,
        NULL, OPEN_EXISTING, 0, NULL);
    if (hLoaderDev != INVALID_HANDLE_VALUE) {
        if (!DeviceIoControl(hLoaderDev, IOCTL_UNLOAD_NDIS_MINIPORT, NULL, -1,
            NULL, 0,
            &bytesReturned, NULL)){
            MessageBox(NULL, TEXT("SYSIO IoControl Failed"), TEXT("Network
                loader"), MB_ICONHAND);
            if (hLoaderDev != INVALID_HANDLE_VALUE) CloseHandle(hLoaderDev);
            hLoaderDev = INVALID_HANDLE_VALUE; // bad handle
        }else {
            CloseHandle(hLoaderDev);
        }
    }
}
```

The API provided by Intermec Technologies exposes a limited set of routines for a programmer to access and affect the 802.11b/g network interface

card from within their application. The routines provided also reads/writes values to the CE registry pertaining to the 802.11b/g radio driver. By using the provided functions, a programmer can alter the 802.11b/g parameters of Network Name (SSID), WEP keys, infrastructure modes, radio channel, and power management modes. A programmer can also retrieve network connect status and signal strength indication from the RF network card.

The API is contained within the 80211API.DLL file that should be present in any load with the 802.11b/g networking installed.

NETWLAN.DLL PRISMNDS.DLL	This is the 802.11b/g driver. It is present in all CN2B CE loads that use the 802.11b/g network interface card.
80211API.DLL	This file is an Intermec authored file that provides the programmer with a set of API calls to configure or monitor status of the 802.11b/g network.
80211CONF.EXE	This is the “Control Panel” for configuring the 802.11b/g network parameters. Note that it is an EXE file and is actually called by CPL802.CPL (see below).
CPL802.CPL	A control panel application that does nothing but call 80211CONF.EXE.
80211SCAN.EXE	Internally manages the Scan List activity.
802PM.DLL	This handles profile management for radio configurable values.
URODDSV.C	This handles radio configuration and security authentication based on a selected profile.

The Profile Manager supports up to four radio configuration profiles. These profiles are the same as those set by the Wireless Network applet that runs on the Windows CE unit. You can configure different 802.11b/g profiles and switch between them using the 802.11 API. See the `ConfigureProfile()` function on page 176 for more information.

Basic Connect/Disconnect Functions

These functions are available when using the 802.11b/g radio module.

RadioConnect()

Connects to the available radio. Use this function if you plan on using a lot of API calls that talk directly to the radio. Note that the 802.11b/g radio must be enabled via NDISTRAY before you can connect to it.

Syntax	UINT RadioConnect ();
Parameters	None
Return Values	ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED
Remarks	Call this function before calling other functions found within this API. It hunts out and connects to the 802.11b/g radio available on the system. Check extended error codes if anything else is returned.
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_RadioConnect)(); #else UINT RadioConnect(); #endif</pre>

RadioDisconnect()

Call this function when done using the 802.11 API to clean up a connection from a previous RadioConnect() call. If you do not call this function, you may leave memory allocated.

Syntax	UINT RadioDisconnect ();
Parameters	None
Return Values	ERROR_SUCCESS when successful, otherwise ERR_CONNECT_FAILED
Remarks	None
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_RadioDisconnect)(); #else UINT RadioDisconnect(); #endif</pre>

RadioDisassociate()

Call this function to have the 802.11b/g radio disassociate from the current service set. The radio then enters an “off” mode until it is woken again by setting the Service Set Identifier (SSID). Also, the NDIS driver generates an NDIS media disconnect event.

Syntax	UINT RadioDisassociate ();
Parameters	None
Return Values	ERROR_SUCCESS on success, else ERR_CONNECT_FAILED
Remarks	None
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_RadioDisassociate)(); #else UINT RadioDisassociate(); #endif</pre>

Query Information Functions**GetAssociationStatus()**

Call this to obtain the radio’s current association status with a service set.

Syntax	UINT GetAssociationStatus (ULONG <i>ϕ</i>);
Parameters	NDIS_RADIO_ASSOCIATED Indicates the radio is associated with an access point NDIS_RADIO_SCANNING Indicates radio is looking to associate with an access point
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	Data is only valid if the function returns ERROR_SUCCESS. Also, if ERROR_SUCCESS is returned, your ULONG reference is populated by one of the parameters listed above.

Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetAssociationStatus)(ULONG &); #else UINT GetAssociationStatus(ULONG &); #endif</pre>
-------------	--

GetAuthenticationMode()

Call this function to obtain the radio's current authentication mode.

Syntax	UINT GetAuthenticationMode (ULONG &);	
Parameters	NDIS_RADIO_AUTH_MODE_OPEN	802.11 Open Authentication. Indicates that the radio is using an open system.
	NDIS_RADIO_AUTH_MODE_SHARED	802.11 Shared Authentication. Indicates that the radio is using a shared key.
	NDIS_RADIO_AUTH_MODE_AUTO	Auto switch between Open/Shared. Indicates automatic detection is used when available.
	NDIS_RADIO_AUTH_MODE_ERROR	Defined as error value. Indicates authentication mode was not determined or is unknown.
	NDIS_RADIO_AUTH_MODE_WPA	WPA Authentication
	NDIS_RADIO_AUTH_MODE_WPA_PSK	WPA Preshared Key Authentication
	NDIS_RADIO_AUTH_MODE_WPA_NONE	WPA None
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	Data is only valid if ERROR_SUCCESS is returned. Also, if ERROR_SUCCESS is returned, your USHORT reference is populated with one of the parameters listed above.	
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetAuthenticationMode)(ULONG &); #else UINT GetAuthenticationMode(ULONG &); #endif</pre>	

GetBSSID()

Call this function to get the current MAC address (BSSID) of the service set. In ESS mode, this is the MAC address of the access point the radio is associated with. In IBSS mode, this is a randomly generated MAC address, and serves as the ID for the IBSS.

Syntax	UINT GetBSSID (TCHAR *);
Parameters	Pointer to a character array, which is populated with the current BSSID after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the BSSID of the current service set: xx-xx-xx-xx-xx-xx

Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetBSSID)(TCHAR *); #else UINT GetBSSID(TCHAR *); #endif</pre>
-------------	--

GetDiversity()

Call this function to get the current diversity setting of your 802.11b/g radio. This uses an optional NDIS5.1 OID to query the radio, of which a large number of 802.11b/g devices do not support. This may be inaccurate.

Syntax	UINT GetDiversity (USHORT *);
Parameters	None.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your USHORT reference is populated with one of the parameters listed above.
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetDiversity)(USHORT *); #else UINT GetDiversity(USHORT *); #endif</pre>

GetLinkSpeed()

Call this function to get the current link speed of the 802.11b/g radio.

Syntax	UINT GetLinkSpeed (int <i>Ⓢ</i>);
Parameters	This accepts an int reference, and your int is populated with the current link speed, in Mbps, rounded to the nearest whole integer, for example: 1, 2, 5, 11, etc.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	Data returned is valid if ERROR_SUCCESS is returned.
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetLinkSpeed)(int &); #else UINT GetLinkSpeed(int &); #endif</pre>

GetMac()

Call this function to get the MAC address of the 802.11b/g radio.

Syntax	UINT GetMac (TCHAR *);
Parameters	Pointer to a character array, which is populated with the MAC address after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the formatted MAC address of the adapter, as follows: xx-xx-xx-xx-xx-xx
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetMac)(TCHAR *); #else UINT GetMac(TCHAR *); #endif</pre>



Note: Be sure to call RadioConnect() *before* calling this function for this function to work properly.

GetNetworkMode()

Call this function to get the current Network Mode (SSID) for the 802.11b/g radio.

Syntax	UINT GetNetworkMode (ULONG &);												
Parameters	<table> <tr> <td>NDIS_NET_MODE_IBSS</td><td>802.11 Ad-Hoc Mode.</td></tr> <tr> <td>NDIS_NET_MODE_ESS</td><td>802.11 Infrastructure Mode.</td></tr> <tr> <td>NDIS_NET_MODE_UNKNOWN</td><td>Anything Else/Unknown Error</td></tr> <tr> <td>NDIS_NET_AUTO_UNKNOWN</td><td>Automatic Selection. <i>Use of this option is not supported or recommended.</i></td></tr> <tr> <td>NDIS_NET_TYPE_OFDM_5G</td><td>5 Gigahertz 54 Mbps</td></tr> <tr> <td>NDIS_NET_TYPE_OFDM_2_4G</td><td>802.11 2.4 Gigahertz</td></tr> </table>	NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.	NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.	NDIS_NET_MODE_UNKNOWN	Anything Else/Unknown Error	NDIS_NET_AUTO_UNKNOWN	Automatic Selection. <i>Use of this option is not supported or recommended.</i>	NDIS_NET_TYPE_OFDM_5G	5 Gigahertz 54 Mbps	NDIS_NET_TYPE_OFDM_2_4G	802.11 2.4 Gigahertz
NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.												
NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.												
NDIS_NET_MODE_UNKNOWN	Anything Else/Unknown Error												
NDIS_NET_AUTO_UNKNOWN	Automatic Selection. <i>Use of this option is not supported or recommended.</i>												
NDIS_NET_TYPE_OFDM_5G	5 Gigahertz 54 Mbps												
NDIS_NET_TYPE_OFDM_2_4G	802.11 2.4 Gigahertz												
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.												
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.												
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetNetworkMode)(ULONG &); #else UINT GetNetworkMode(ULONG &); #endif</pre>												

GetNetworkType()

Call this function to get the current network type of the radio. Do not confuse this with GetNetworkMode().

Syntax	UINT GetNetworkType (ULONG &);	
Parameters	NDIS_NET_TYPE_FH	Indicates this is a frequency hopping radio.
	NDIS_NET_TYPE_DS	Indicates that this is a direct sequence radio.
	NDIS_NET_TYPE_UNDEFINED	Indicates this radio is unknown or undefined.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetNetworkType)(ULONG &); #else UINT GetNetworkType(ULONG &); #endif</pre>	

GetSSID()

Call this function to get the desired SSID of the 802.11b/g radio.

Syntax	UINT GetSSID (TCHAR *);	
Parameters	Pointer to a character array, which is populated with the current SSID when successful.	
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your TCHAR array is populated with the desired SSID.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetSSID)(TCHAR *); #else UINT GetSSID(TCHAR *); #endif</pre>	



Note: Call RadioConnect() *before* this function for this function to work properly.

GetPowerMode()

Call this function to get the current power savings mode of the radio.

Syntax	UINT GetPowerMode (ULONG &);	
Parameters	NDIS_RADIO_POWER_MODE_CAM	Continuous Access Mode (<i>ie: always on</i>).
	NDIS_RADIO_POWER_MODE_PSP	Power Saving Mode.
	NDIS_RADIO_POWER_UNKNOWN	Unknown power mode.
	NDIS_RADIO_POWER_AUTO	Auto.
	NDIS_RADIO_POWER_MODE_FAST_PSP	Fast PSP, good savings, fast
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of the parameters listed above.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetPowerMode)(ULONG &); #else UINT GetPowerMode(ULONG &); #endif</pre>	



Note: Do not use Automatic Switching mode at this time.

GetRSSI()

Call this function to get the current RSSI (Radio Signal Strength Indicator), in Dbm.

Syntax	UINT GetRSSI (ULONG &);
Parameters	References a ULONG that is populated with the current RSSI after a successful call.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your ULONG reference contains the RSSI. Valid RSSI range is from -100 Dbm to -30 Dbm.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetRSSI)(ULONG &); #else UINT GetRSSI(ULONG &); #endif</pre>

GetTXPower()

Call this function to get the current transmit power of the radio.

Syntax	UINT GetTXPower (ULONG <i>ϕ</i>);	
Parameters	NDIS_POWER_LEVEL_63	63 mW
	NDIS_POWER_LEVEL_30	30 mW
	NDIS_POWER_LEVEL_15	15 mW
	NDIS_POWER_LEVEL_5	5 mW
	NDIS_POWER_LEVEL_1	1 mW
	NDIS_POWER_LEVEL_UNKNOWN	Unknown Value or Error.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with the TX power in milliwatts (mW). Valid ranges are from 5 mW to 100 mW.	
Definitions	<pre> #ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetTXPower)(ULONG &); #else UINT GetTXPower(ULONG &); #endif </pre>	

GetWepStatus()

Call this to get the current state of the radio's WEP and encryption levels.

Syntax	UINT GetWepStatus (ULONG <i>ϕ</i>);	
Parameters	NDIS_ENCRYPTION_1_ENABLED	WEP enabled; TKIP, AES not enabled, and transmit key may or may not be available.
	NDIS_ENCRYPTION_DISABLED	Indicates AES, TKIP, WEP disabled, and transmit key available.
	NDIS_ENCRYPTION_NOT_SUPPORTED	Indicates WEP, TKIP, AES not supported.
	NDIS_ENCRYPTION_1_KEY_ABSENT	Indicates AES, TKIP, WEP disabled, and transmit key not available.
	NDIS_ENCRYPTION_2_ENABLED	Indicates TKIP, WEP enabled; AES not enabled, and transmit key available.
	NDIS_ENCRYPTION_2_KEY_ABSENT	Indicates no transmit keys available for TKIP, WEP, TKIP, WEP enabled; AES not enabled.
	NDIS_ENCRYPTION_3_ENABLED	Indicates AES, TKIP, WEP enabled, and transmit key available.
	NDIS_ENCRYPTION_3_KEY_ABSENT	Indicates no transmit keys available for AES, TKIP, WEP, AES, TKIP, WEP enabled.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	If ERROR_SUCCESS returned, ULONG reference is populated with a parameter listed above.	

Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetWepStatus)(ULONG &); #else UINT GetWepStatus(ULONG &); #endif</pre>
-------------	--

GetRadioIpAddress()

Call this function to obtain a formatted string indicating whether DHCP is enabled, and what is the current adapters IP address.

Syntax	UINT GetRadioIpAddress(TCHAR *);
Parameters	Pointer to a character array that contains the formatted string of the IP address and static/DHCP information.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If ERROR_SUCCESS is returned, your TCHAR array contains a string formatted as follows: IP: DHCP Enabled\nxxx.xxx.xxx.xxx\n or IP: DHCP Disabled\nxxx.xxx.xxx.xxx\n
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetRadioIpAddress)(TCHAR *); #else UINT GetRadioIpAddress(TCHAR *); #endif</pre>

GetCCXStatus()

Call this to get information about the current CCX status of the adapter.

Syntax	UINT GetCCXStatus(ULONG &);				
Parameters	<table> <tr> <td>NDIS_NETWORK_EAP_MODE_OFF</td><td>Disable EAP mode.</td></tr> <tr> <td>NDIS_NETWORK_EAP_MODE_ON</td><td>Enable EAP mode.</td></tr> </table>	NDIS_NETWORK_EAP_MODE_OFF	Disable EAP mode.	NDIS_NETWORK_EAP_MODE_ON	Enable EAP mode.
NDIS_NETWORK_EAP_MODE_OFF	Disable EAP mode.				
NDIS_NETWORK_EAP_MODE_ON	Enable EAP mode.				
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.				
Remarks	If ERROR_SUCCESS is returned, your ULONG reference is populated with one of parameters listed above.				
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_GetCCXStatus)(ULONG &); #else UINT GetCCXStatus(ULONG &); #endif</pre>				

Set Information Functions

AddWep()

Call this function to add a WEP key to the radio. Call this function multiple times when adding more than one WEP key. Save the “default” key for last. For example, when adding four keys, and the second key is the default transmit key, add keys 1, 3 and 4 *before* you add key 2.



Note: Add the default transmit key *last*.

Syntax	UINT AddWep (ULONG, BOOL, TCHAR *);	
Parameters	ULONG	Specifies the key index to be set. Valid values are 0-3.
	BOOL	When set to TRUE, specifies that this key is the default transmit key.
	TCHAR	Pointer to a character array that specifies the key data in either HEX (length of 10 or 26) or ASCII (length of 5 or 13). This string must be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	When adding WEP keys to the radio, turn off encryption before you add the keys, then turn encryption back on afterwards. Also, be sure to add the TRANSMIT KEY last.	
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_AddWep)(ULONG, BOOL, TCHAR *); #else UINT AddWep(ULONG, BOOL, TCHAR *); #endif</pre>	

EnableWep()

Enables or disables WEP encryption on the radio (TRUE/FALSE).

Syntax	UINT EnableWep (BOOL);	
Parameters	Set BOOL to TRUE to enable WEP encryption, or FALSE to disable WEP encryption.	
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	Call this function with TRUE as the parameter to enable WEP encryption. Call this function with the FALSE parameter to disable WEP encryption. This call is an alias for EncryptionStatus(). See the following: EnableWEP(TRUE) = EncryptionStatus(NDIS_ENCRYPTION_1_ENABLED) EnableWEP(FALSE) = EncryptionStatus(NDIS_ENCRYPTION_DISABLED)	
Definitions	<pre>#ifndef DYNAMIC_LOADING typedef UINT (*PFN_EnableWep)(BOOL); #else UINT EnableWep(BOOL); #endif</pre>	

EncryptionStatus()

Call this function to set the desired encryption status.

Syntax	UINT EncryptionStatus (UINT <i>mode</i>);	
Parameters	NDIS_ENCRYPTION_1_ENABLED	WEP is enabled; TKIP and AES are not enabled, and a transmit key may or may not be available. (<i>same as NDIS_RADIO_WEP_ENABLED</i>)
	NDIS_ENCRYPTION_DISABLED	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is available. (<i>Same as NDIS_RADIO_WEP_DISABLED</i>)
	NDIS_ENCRYPTION_NOT_SUPPORTED	Indicates that encryption (WEP, TKIP, and AES) is not supported. (<i>Same as NDIS_RADIO_WEP_NOT_SUPPORTED</i>)
	NDIS_ENCRYPTION_1_KEY_ABSENT	Indicates that AES, TKIP, and WEP are disabled, and a transmit key is not available. (<i>Same as NDIS_RADIO_WEP_ABSENT</i>)
	NDIS_ENCRYPTION_2_ENABLED	Indicates that TKIP and WEP are enabled; AES is not enabled, and a transmit key is available.
	NDIS_ENCRYPTION_2_KEY_ABSENT	Indicates no transmit keys available for use by TKIP or WEP (enabled) and AES is not enabled.
	NDIS_ENCRYPTION_3_ENABLED	Indicates that AES, TKIP, and WEP are enabled, and a transmit key is available.
	NDIS_ENCRYPTION_3_KEY_ABSENT	Indicates there are no transmit keys available for use by AES, TKIP, or WEP, which are enabled.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	
Definitions	<pre> #ifdef DYNAMIC_LOADING typedef UINT (*PFN_EncryptionStatus)(UINT mode); #else UINT EncryptionStatus(UINT mode); #endif </pre>	

SetAuthenticationMode()

Call this function to set the desired authentication mode.

Syntax	UINT SetAuthenticationMode (ULONG);	
Parameters	NDIS_RADIO_AUTH_MODE_OPEN	802.11 Open Authentication. Indicates that the radio is using an open system.
	NDIS_RADIO_AUTH_MODE_SHARED	802.11 Shared Authentication. Indicates that the radio is using a shared key.
	NDIS_RADIO_AUTH_MODE_AUTO	Auto switch between Open/Shared. Indicates automatic detection is used when available.
	NDIS_RADIO_AUTH_MODE_ERROR	Defined as error value. Indicates the authentication mode was not determined at this time or is unknown.
	NDIS_RADIO_AUTH_MODE_WPA	WPA Authentication
	NDIS_RADIO_AUTH_MODE_WPA_PSK	WPA Preshared Key Authentication
	NDIS_RADIO_AUTH_MODE_WPA_NONE	WPA None
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetAuthenticationMode)(ULONG); #else UINT SetAuthenticationMode(ULONG); #endif</pre>	

SetChannel()

This function is currently not implemented. Ad-hoc networks automatically select a channel or use the already existing channel.

Syntax	UINT SetChannel (USHORT);
Parameters	USHORT value that should populate with the desired channel (1-14).
Return Values	None.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetChannel)(USHORT); #else UINT SetChannel(USHORT); #endif</pre>

SetNetworkMode()

Call this function to set the desired Network Mode.

Syntax	UINT SetNetworkMode (ULONG);	
Parameters	NDIS_NET_MODE_IBSS	802.11 Ad-Hoc Mode.
	NDIS_NET_MODE_ESS	802.11 Infrastructure Mode.
	NDIS_NET_MODE_UNKNOWN	Anything Else/Unknown Error
	NDIS_NET_AUTO_UNKNOWN	Automatic Selection. <i>Use of this option is not supported or recommended.</i>
	NDIS_NET_TYPE_OFDM_5G	5 Gigahertz 54 Mbps
	NDIS_NET_TYPE_OFDM_2_4G	802.11 2.4 Gigahertz
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetNetworkMode)(ULONG); #else UINT SetNetworkMode(ULONG); #endif</pre>	

SetPowerMode()

Call this function to set the desired power mode.

Syntax	UINT SetPowerMode (ULONG <i>mode</i>);	
Parameters	NDIS_RADIO_POWER_MODE_CAM	Continuous Access Mode (<i>ie: always on</i>).
	NDIS_RADIO_POWER_MODE_PSP	Power Saving Mode.
	NDIS_RADIO_POWER_UNKNOWN	Unknown power mode.
	NDIS_RADIO_POWER_AUTO	Auto.
	NDIS_RADIO_POWER_MODE_FAST_PSP	Fast PSP, good savings, fast
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.	
Remarks	None.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetPowerMode)(ULONG mode); #else UINT SetPowerMode(ULONG mode); #endif</pre>	

SetSSID()

Call this function with a pointer to a null-terminated TCHAR array containing the desired SSID to set the desired SSID of the adapter.

Syntax	UINT SetSSID(TCHAR *);
Parameters	Pointer to a character array that contains the desired SSID. This should be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	If an “ANY” network is desired, pass in _T(“ANY”).
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetSSID)(TCHAR *); #else UINT SetSSID(TCHAR *); #endif</pre>

SetCCXStatus()

Call this function to set the desired CCX / Network EAP status.

Syntax	UINT SetCCXStatus(ULONG);
Parameters	NDIS_NETWORK_EAP_MODE_OFF Disable Network EAP / CCX NDIS_NETWORK_EAP_MODE_ON Enable Network EAP / CCX
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetCCXStatus)(ULONG); #else UINT SetCCXStatus(ULONG); #endif</pre>

SetMixedCellMode()

Call this function to set the desired mixed cell mode.

Syntax	UINT SetMixedCellMode(ULONG);
Parameters	NDIS_MIXED_CELL_OFF Disable Mixed Cell NDIS_MIXED_CELL_ON Enable Mixed Cell
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SetMixedCellMode)(ULONG); #else UINT SetMixedCellMode(ULONG); #endif</pre>

RemoveWep()

Call this with a key index of 0-3 to remove the WEP key at that index.

Syntax	UINT RemoveWep (ULONG);
Parameters	ULONG value that specifies the key index to set. Valid values are 0-3.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when fails, or ERR_CONNECT_FAILED if connection with radio fails.
Remarks	On disassociation with all BSSIDs of the current service set, WEP key is removed by the adapter.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_RemoveWep)(ULONG); #else UINT RemoveWep(ULONG); #endif</pre>

Helper Functions**ConfigureProfile()**

If using the Intermec 802.11 Profile Management system, you can pass a specific profile name to program the API to configure the radio.

Syntax	UINT ConfigureProfile (TCHAR *);
Parameters	Pointer to a character array that contains the profile name. This should be null-terminated.
Return Values	ERROR_SUCCESS when successful, ERR_QUERY_FAILED when the query failed, or ERR_CONNECT_FAILED if a connection with the radio failed.
Remarks	Call this function with a pointer to a null-terminated TCHAR array that contains the name of the profile you wish to configure. This function reads profile data from the profile manager, sets that profile as the default active profile, and configures the radio appropriately. If needed, the supplicant and any other related services are automatically started and stopped.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_ConfigureProfile)(TCHAR *); #else UINT ConfigureProfile(TCHAR *); #endif</pre>

EnableZeroConfig()

This enables or disables the Wireless Zero Configuration Wizard from Microsoft. After calling this function, a warm-boot is required for the change to take effect. *Note that enabling this effectively disables all SET commands in this API.*

Syntax	UINT EnableZeroConfig(USHORT);	
Parameters	TRUE	Enable Wireless Zero Config
	FALSE	Disable Wireless Zero Config
Return Values	ERROR_SUCCESS when successful, ERR_ZERO_CONFIG_CHANGE_FAILED when the query failed.	
Remarks	Call this function to set the desired Zero Config status.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_EnableZeroConfig)(USHORT); #else UINT EnableZeroConfig(USHORT); #endif</pre>	

isZeroConfigEnabled()

Call this function to determine whether Zero Config is currently enabled.

Syntax	UINT isZeroConfigEnabled();	
Parameters	None.	
Return Values	TRUE if ZeroConfig is enabled, and FALSE if it is disabled.	
Remarks	None.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_isZeroConfigEnabled)(); #else UINT isZeroConfigEnabled(); #endif</pre>	

isSupplicantRunning()

Call this function to determine whether the security supplicant is running.

Syntax	UINT isSupplicantRunning();	
Parameters	None.	
Return Values	TRUE if the security supplicant is running, FALSE if not running.	
Remarks	None.	
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_isSupplicantRunning)(); #else UINT isSupplicantRunning(); #endif</pre>	

StartScanList()

If a scan list is configured on the system, this causes the API to begin the process of scanning for an available network. This call can take quite a while to process (*depending upon the length of the scan list and how long it takes to find a valid network*), you may wish to call it from a separate thread.

Syntax	UINT StartScanList();
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	Call this function to start the scan list functionality of the system.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_StartScanList)(); #else UINT StartScanList(); #endif</pre>

StartSupplicant()

Call this to start the supplicant service if it is installed on the system.

Syntax	UINT StartSupplicant();
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_StartSupplicant)(); #else UINT StartSupplicant(); #endif</pre>

StopSupplicant()

Call this function to stop the supplicant service.

Syntax	UINT StopSupplicant();
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_StopSupplicant)(); #else UINT StopSupplicant(); #endif</pre>

isDHCPEnabled()

Call this to determine whether DHCP is enabled on the current adapter.

Syntax	UINT isDHCPEnabled ();
Parameters	None.
Return Values	TRUE if DHCP is enabled, FALSE if it is not.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_isDHCPEnabled)(); #else UINT isDHCPEnabled(); #endif</pre>

RenewDHCP()

Call this to force a DHCP renewal on the current network adapter.

Syntax	UINT RenewDHCP ();
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	You should not have to call this function.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_RenewDHCP)(); #else UINT RenewDHCP(); #endif</pre>

GetCurrentDriverName()

Call this function to populate the TCHAR array with the driver name.

Syntax	UINT GetCurrentDriverName (TCHAR *);
Parameters	Pointer to a TCHAR array which contains the name of the driver when successful.
Return Values	ERROR_SUCCESS when successful.
Remarks	This function is called with a pointer to a TCHAR array that is large enough to hold the name of the driver PLUS the null terminator.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_GetCurrentDriverName)(TCHAR *); #else UINT GetCurrentDriverName(TCHAR *); #endif</pre>

ResetRadioToSystemSave()

Call this function to force the radio to reset to the last desired active profile.

Syntax	UINT ResetRadioToSystemSave ();
Parameters	None.
Return Values	ERROR_SUCCESS when successful.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_ResetRadioToSystemSave)(); #else UINT ResetRadioToSystemSave(); #endif</pre>

EnableSuppLogging()

Call this function to set the desired supplicant logging mode.

Syntax	UINT EnableSuppLogging (ULONG);
Parameters	NDIS_SUPP_LOGGING_ON Supplicant Logging Enabled NDIS_SUPP_LOGGING_OFF Supplicant Logging Disabled
Return Values	ERROR_SUCCESS when successful.
Remarks	None.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_EnableSuppLogging)(ULONG); #else UINT EnableSuppLogging(ULONG); #endif</pre>

SwitchPacketDriver()

Call this function to switch between available packet drivers on the system.

Syntax	UINT SwitchPacketDriver (USHORT);
Parameters	INTERMEC_PACKET_DRIVER Intermec Packet Driver (ZNICZIO) NDISUIO_PACKET_DRIVER Microsoft Packet Driver (NDISUIO)
Return Values	ERROR_SUCCESS when successful.
Remarks	After switching to a new packet driver, perform a warm boot for changes to take effect.
Definitions	<pre>#ifdef DYNAMIC_LOADING typedef UINT (*PFN_SwitchPacketDriver)(USHORT); #else UINT SwitchPacketDriver(USHORT); #endif</pre>

Notifications

Use the following information to programmatically control the vibrator, to write an application to turn on the vibrator when a message is received via the WLAN radio link, and turn it off when the user hits a key.

Vibrator support is implemented in the NLED driver as a false LED. The vibrator is LED 5 and is identified with an CycleAdjust of -1. The vibrate option is only available in the notifications panel when the vibrator is present in the system.

Regarding an applications interface to NLED.DLL, LEDs must be available for use by applications. This is possible via two functions exported by the COREDLL.DLL file. To use the LED functions, declare these as extern “C” as follows:

```
extern "C" BOOL WINAPI NLEDGetDeviceInfo(UINT nInfoId,
void *pOutput) ;
extern "C" BOOL WINAPI NLEDSetDevice( UINT nDeviceId, void
*pInput) ;
```

The LEDs are enumerated for access through the data structures associated with these APIs: Notification LED (0), Alpha Lock LED (2), Scanner LED (3), or Low Battery (4).

NLEDGetDeviceInfo

Usage

#include “nled.h”

Syntax

```
BOOL NLEDGetDeviceInfo ( UINT nInfoId, void *pOutput );
```

Parameters

nInfoId	Integer specifying the information to return. These values are defined:	
	NLED_COUNT_INFO	Indicates the <i>pOutput</i> buffer specifies the number of LEDs on the device.
	NLED_SUPPORTS_INFO_ID	Indicates the <i>pOutput</i> buffer specifies information about the capabilities supported by the LED.
	NLED_SETTINGS_INFO_ID	Indicates the <i>pOutput</i> buffer contains information about the LED current settings.
pOutput	Pointer to the buffer to which the information is returned. The buffer points to various structure types defined in “nled.h”, depending on the value of <i>nId</i> , as detailed in the following table:	
	Value of <i>nId</i>	Structure in <i>pOutput</i>
	NLED_COUNT_INFO	NLED_COUNT_INFO
	NLED_SUPPORTS_INFO	NLED_SUPPORTS_INFO
	NLED_SETTINGS_INFO	NLED_SETTINGS_INFO

NLEDSetDevice

Usage

#include "nled.h"

Syntax

```
BOOL NLEDSetDevice ( UINT nDeviceId, void *pInput );
```

Parameters

nDeviceId	Integer specifying the device identification. The following is defined:
NLED_SETTINGS_INFO_ID	Contains information about the desired LED settings.
pInput	Pointer to the buffer that contains the NLED_SETTINGS_INFO structure.

Reboot Functions

There are several methods, via Kernel I/O Control functions, that an application program can use to force the CN2B Computer to reboot.

IOCTL_HAL_REBOOT

IOCTL_HAL_REBOOT performs a warm-boot. See page 159.

IOCTL_HAL_COLDBOOT

Invoking with this forces a cold reboot, resets the CN2B Computer, reloads Windows CE as if a power-up was performed, and discards the contents of the Windows CE RAM-based object. See page 157.

IOCTL_HAL_WARMBOOT

This function is supported on CN2B Computers. It performs a warm boot of the system, preserving the object store. See page 156.

Remapping the Keypad



Note: Use caution when remapping the keypad. Improper remapping may render the keypad unusable. Data within the CN2B Computer could also be lost, should any problems occur.

Applications have the ability to remap keys on the CN2B keypad. This allows applications to enable keys that would otherwise not be available, such as the [F1] function key. Also, to disable keys that should not be available, such as the alpha key because no alpha entry is required. Use caution when attempting to remap the keypad because improper remapping may cause the keypad to become unusable. This can be corrected by performing a cold-boot on the device that reloads the default keymap.

Note that remapping the keys in this way affects the key mapping for the entire system, not just for the application that does the remapping.

There are three “planes” supported for the CN2B keypad. Keys used in more than one shift plane must be described in each plane.

Unshifted Plane

The unshifted plane contains values from the keypad when not pressed with other keys, such as the following:

Press the Key	To Enter This
[1]	1
[5]	5
[9]	9

Orange Plane

The **orange** plane contains values from the keypad when a key is simultaneously pressed with the **orange** key, such as the following:

Press the Keys	To Enter This
orange [0]	Start menu
orange [6]	A4
orange [9]	PgDn

Alpha (Green) Plane

The alpha plane contains values from the keypad when the keypad has been placed in alpha mode by pressing the green **[Alpha]** key, such as:

Press the Keys	To Enter This
[Alpha] [1]	Caps
[Alpha] [5]	j
[Alpha] [9]	w

Key Values

Key values for each plane are stored in the registry. All units ship with a default key mapping loaded in the registry. Applications that change the default mapping need to read the appropriate key from the registry into an array of words, modify the values required and then write the updated values back into the registry. The registry access can be done with standard

Microsoft API calls, such as `RegOpenKeyEx()`, `RegQueryValueEx()`, and `RegSetValueEx()`. These registry keys contain the plane mappings:

- The unshifted plane mapping can be found in the registry at:
`HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\Vkey`
- The orange plane mapping can be found in the registry at:
`HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyGold`
- The alpha plane mapping can be found in the registry at:
`HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyAlpha`

How Key Values Are Stored in Registry

To know which fields to update in the registry, you must know what Scan Codes are assigned to each physical key (see page 185). The Scan Code is used at the lowest level of the system to let the keypad driver know which physical key is pressed. The keypad driver takes that scan code and looks it up in a table (a copy of the table in the registry) to determine the values to pass to the operating system.

Each registry key is just an array that describes to the keypad driver what value needs to be passed for each physical key. The key values are indexed by the scan code, this is a zero-based index. For example in the unshifted plane, the [4] key has a scan code of 0x06. This means that the seventh word under the “Vkey” registry key has the value for the [4] key. Taking a sample of the “Vkey” registry key shows the following values:

00, 00, 0B, 05, 02, 03, C1, 07, 04, 03, BE, 00, 34, 00, 00, 00, . . .

The value is 34,00. The values are in reverse byte order because that is the way the processor handles data. When writing an application, nothing needs to be done to swap the bytes, as this will happen automatically when the data is read into a byte value. This is something you just need to be aware of when looking at the registry. Knowing this, we can see that the value that the keypad driver will pass to the system is a hex 34. Looking that up on an UNICODE character chart, we see that it maps to a “4”. If you wanted the key, labeled “4”, to output the letter “A” instead, you would need to change the seventh word to “41” (the hexadecimal representation of “A” from the UNICODE chart), then put the key back into the registry.



Note: Do not remap scan codes 0x01, 0x41, 0x42, 0x43, or 0x44, or the CN2B Computer becomes unusable until a cold-boot is performed.

If you wish to disable a certain key, remap its scan code to 0x00.

Change Notification

Just changing the registry keys do not immediately change the key mappings. Signal the “ITC_KEYBOARD_CHANGE” named event using the `CreateEvent()` API to notify the keypad driver the registry was updated.

Advanced Keypad Remapping

It is also possible to map multiple key presses to one button and to map named system events to a button. The multiple key press option could be useful to cut down on the number of keys needed to press in a given situation or to remap which key behaves like the action key. Mapping events to a button could be useful to change which buttons will fire the scanner, control volume, and allow for suspending and resuming the device. If you need help performing one of these advanced topics please contact Intermec Technical Support.

Scan Codes

At the lowest driver level, the CN2B keypad identifies keys as scan codes. These scan codes are sent via the keypad microcontroller, and cannot be changed without modifying the keypad firmware.

Keypad Scan Codes and Meanings

Press this Key	Meaning	Scan Code
	Reserved	0x00
[I/O]	I/O button	0x01
	Scanner Handle Trigger	0x02
	Scanner Left	0x03
	Scanner Right	0x04
[4]	4/GHI/A2	0x06
	None	0x07
	Left arrow/Back Tab	0x08
	None	0x09
[BkSp]	BkSp// (forward slash)	0x0A
	orange key	0x0B
	None	0x0C
[Esc]	Esc/– (minus sign)	0x0D
[v]	Down arrow	0x0E
[1]	1/Caps	0x0F
[7]	7/PQRS/PgUp	0x10
[Alpha]	[Alpha] key	0x11
	None	0x12

Keypad Scan Codes and Meanings (continued)

Press this Key	Meaning	Scan Code
[^]	Up arrow/Volume increase	0x13
	Right arrow/Tab	0x14
[2]	2/ABC	0x15
[8]	8/TUV/* (asterisk)	0x16
[0]	0/Win	0x17
[5]	5/JKL/A3	0x18
	None	0x19
[Action]	Action/+ (plus symbol)	0x1A
[3]	3/DEF/backlight	0x1B
[9]	9/WXYZ/PgDn	0x1C
[ENTER]	Enter/@ (at symbol)	0x1D
[6]	6/MNO/A4	0x1E
	None	0x1F-0x40
	Charge Detect	0x41
	LCD frontlight	0x42
	Ambient light	0x42
	Threshold crossed	0x42
	Headset detected	0x43
	Keypad Backlight	0x44
	Ambient Light	0x44
	Threshold Crossed	0x44

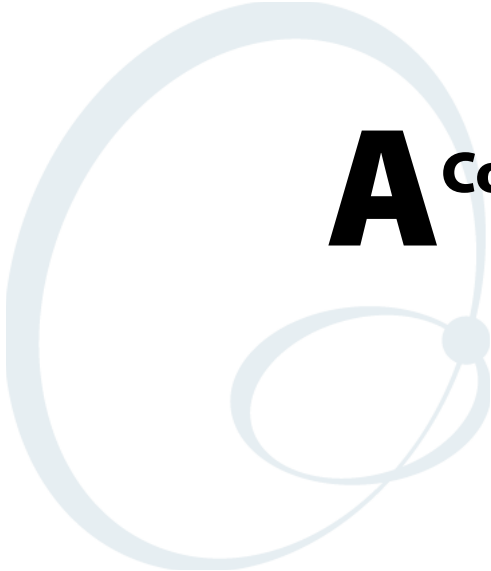
Sample View of Registry Keys

The following is a sample view of the current default key mapping for the CN2B keypad. See the registry on your device for the latest key mappings.

```
[HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD]
"ResumeMask"=dword:7
"Vkey"=hex: 00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
25,00,00,00,08,00,03,02,00,00,1B,00,28,00,31,00,\
37,00,01,02,00,00,26,00,27,00,32,00,38,00,30,00,\
35,00,00,00,01,03,33,00,39,00,0D,00,36,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,07,05,01,05,03,05,02,05

"VkeyGold"=hex:00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
09,01,00,00,BF,00,03,02,00,00,BD,00,75,00,72,00,\
21,00,01,02,00,00,76,00,09,00,73,00,38,01,5B,00,\
35,00,00,00,BB,01,09,05,22,00,32,01,36,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,07,05,01,05,03,05,02,05

"VkeyAlpha"=hex:00,00,0B,05,02,03,C1,07,04,03,BE,00,47,00,00,00,\
25,00,00,00,08,00,03,02,00,00,1B,00,28,00,02,02,\
50,00,01,02,00,00,26,00,27,00,41,00,54,00,20,00,\
4A,00,00,00,01,03,44,00,57,00,0D,00,4D,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
00,00,07,05,01,05,03,05,02,05
```

A Configurable Settings

This appendix contains information about the Intermec Settings, Utilities, and Wireless Network applets that may be on the CN2B Mobile Computer. Information about using reader commands and configuration bar codes to configure some of your settings is also in this appendix.



Note: Information about the settings you can configure with the Intermec Settings applet is described in the *Intermec Computer Command Reference Manual* (P/N: 073529). The online manual is available from the Intermec web site at www.intermec.com.

Configuration Parameters

A configuration parameter changes the way the CN2B Computer operates, such as configuring a parameter to have the CN2B Computer emit a very loud beep in a noisy environment. Use either of the following methods to execute configuration parameters:

- Send parameters from an SNMP management station. See “SNMP Configuration on the Mobile Computer” on page 109.
- Scan EasySet bar codes. You can use the EasySet bar code creation software from Intermec Technologies Corporation to print configuration labels. Scan the labels to change the scanner configuration and data transfer settings.

Use the Intermec EasySet software to print configuration labels you can scan to change your configuration settings. For more information, see the EasySet online help. EasySet is available from the Intermec Data Capture web site.

Menus of available parameters for each group are listed. Use the scroll bars to go through the list. Expand each menu (+) to view its parameter settings. Tap a parameter to select, or expand a parameter to view its subparameters.

Note that each parameter or subparameter is shown with its default setting or current setting in (< >) brackets. Tap a parameter or subparameter to select that parameter, then do any of the following to change its setting: Tap **Apply** to apply any changes.

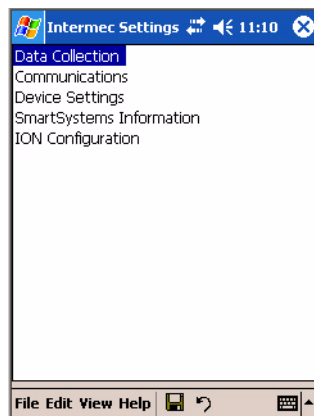
- Typing a new value in an entry field.
- Choosing a new value from the drop-down list.
- Selecting a different option. The selected option contains a bullet.
- Tap **Defaults**, then **Apply** to restore factory-default settings. Tap **Yes** when you are prompted to verify this action.
- Tap **Refresh** to discard changes and start again. Tap **Yes** when you are prompted to verify this action.

Intermec Settings Applet

You may have the Intermec Settings applet. Information about the settings you can configure with this applet is described in the *Intermec Computer Command Reference Manual*. The online manual is available from the Intermec web site at www.intermec.com.

See the Data Collection Resource Kit in the IDL for information about data collection functions. The IDL is available as a download from the Intermec web site at www.intermec.com. Contact your Intermec representative for more information.

To access the settings from the CN2B Computer, tap **Start > Settings > the System tab > Intermec Settings** to access its applet.



Utilities Applet

The Utilities applet examines and modifies settings and operational modes of specific hardware and software on the CN2B Computer, including the registry storage, wakeup mask, and application launch keys.



To access the settings from the CN2B Computer, tap **Start > Settings > the System tab > Utilities** to access its applet. These tabs represent the following groups of settings or parameters:

- Registry Save (*next paragraph*)
- Wakeup Mask (*page 192*)
- App Launch (*page 193*)

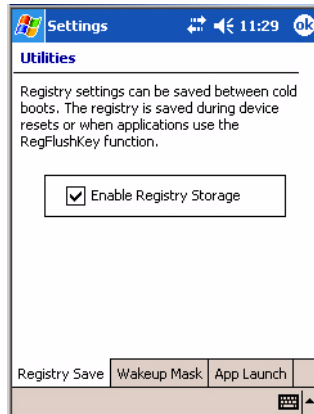
Registry Save



From the CN2B Computer, tap **Start > Settings > the System tab > Utilities > the Registry Save tab** to access the Registry Save page.

For Windows Mobile 2003, the only medium available for saving the registry is the Flash File System (PSM). Registry data is stored in the

“Flash_File_Store\Registry” path. Check **Enable Registry Storage** to enable this function.



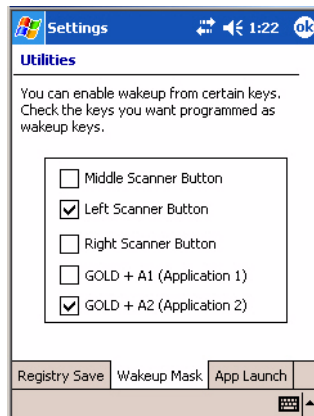
Wakeup Mask



Utilities

From the CN2B Computer, tap **Start** > **Settings** > the **System** tab > **Utilities** > the **Wakeup Mask** tab to access the Wakeup Mask page.

This page programs three scanner buttons and the **A1** and **A2** application keys to be “wakeup” or resume keys. That is, to prompt the CN2B Computer to “wake up” or resume activity after going to “sleep” as a result of being inactive after a length of time. This information remains between warm and cold boots. Check the appropriate box, then tap **ok** to apply.



Based on the setting, do the following to “wake up” the CN2B Computer.

Middle Scanner Button	Squeeze the button on the Scan Handle
Left Scanner Button	Squeeze the left scanner button
Right Scanner Button	Squeeze the right scanner button
GOLD + A1 (Application 1)	Press orange [period]
GOLD + A2 (Application 2)	Press orange [4]

App Launch

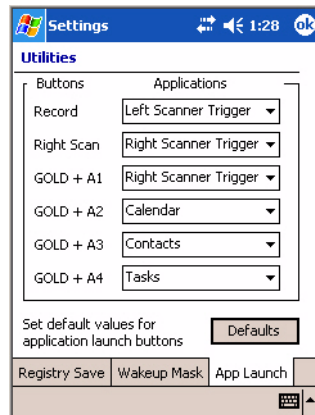


Utilities

From the CN2B Computer, tap **Start > Settings > the System tab > Utilities > the App Launch** tab to access the Application Launch page.

This page programs or maps two scanner buttons and four application keys to start up to six applications. *Note that the left scanner button also acts as the record button.*

For CN2B Computers with an imager,
default mappings are shown in the following illustration.



For CN2B Computers without an imager,
the default maps the Record, Calendar, Contacts, and Tasks applications the top four and the A3 and A4 buttons are “unassigned.”



Note: Record, Calendar, Contacts, and Tasks are Pocket PC applications. See Chapter 2, “Windows Mobile 2003” for information.

- To assign an application to a button, select an application from the applicable drop-down list box.
- To assign a new application, select the “Add new application” option, which brings up an Open File dialog and browse Secure Digital storage cards for new applications.
- To disable or unmap a currently mapped application from a corresponding button, select “unassigned” from the applicable drop-down list.
- Tap **Defaults** in the lower right corner to restore defaults.



Note: You cannot map an application to more than one button. Should you assign the same application to two buttons, a verification prompt appears after the second button to confirm whether you want to remap the application. If you tap **Yes**, the applet changes the first button to “unassigned” and maps the application to the second button.



Note: All changes are activated immediate upon selection.

Wireless Network Applet



Note: See Chapter 4, “Network Support” for information about the 802.11b/g radio module.

About the Wireless Network

Your wireless adapter (network interface card) connects to wireless networks of two types: infrastructure networks and ad-hoc networks.

- Infrastructure networks get you onto your corporate network and the internet. Your CN2B Computer establishes a wireless connection to an access point, which links you to the rest of the network. When you connect to a network via an access point, you are using the 802.11b/g infrastructure mode.
- Ad-hoc networks are private networks shared between two or more clients, even with no access point.

Each wireless network is assigned a name (or Service Set Identifier — SSID) to allow multiple networks to coexist in the same area without infringement.

Intermec Technologies recommends using security measures with wireless networks to prevent unauthorized access to your network and to ensure your privacy of transmitted data. The following are required elements for secure networks:

- Authentication by both the network and the user
- Authentication is cryptographically protected
- Transmitted data

There are many schemes available for implementing these features.

Terminology

Below are terms you may encounter when configuring the wireless network:

- **CKIP** (Cisco Key Integrity Protocol)
This is Cisco’s version of the TKIP protocol, compatible with Cisco Airnet products.
- **EAP** (Extensible Authentication Protocol)
802.11b/g uses this protocol to perform authentication. This is not necessarily an authentication mechanism, but is a common framework for transporting actual authentication protocols. Intermec provides a number of EAP protocols for you to choose the best for the network.
- **TKIP** (Temporal Key Integrity Protocol)
This protocol is part of the IEEE 802.11b/g encryption standard for wireless LANs., which provides per-packet key mixing, a message integ-

rity check and a re-keying mechanism, thus overcoming most of the weak points of WEP. This encryption is more difficult to crack than the standard WEP. Weak points of WEP include:

- No Installation Vector (IV) reuse protection
- Weak keys or no key updates
- No protection against message replay
- No detection of message tampering
- **WEP (Wired Equivalent Privacy) encryption**
With preconfigured WEP, both the client CN2B Computer and access point are assigned the same key, which can encrypt all data between the two devices. WEP keys also authenticate the CN2B Computer to the access point — unless the CN2B Computer can prove it knows the WEP key, it is not allowed onto the network.

WEP keys are only needed if they are expected by your clients. There are two types available: 64-bit (5-character strings, 12345) (default) and 128-bit (13-character strings, 1234567890123). Enter these as either ASCII (12345) or Hex (0x3132333435).

- **WPA (Wi-Fi Protected Access)**
This is an enhanced version of WEP that does not rely on a static, shared key. It encompasses a number of security enhancements over WEP, including improved data encryption via TKIP and 802.11b/g authentication with EAP.

Configuring Your Wireless Network

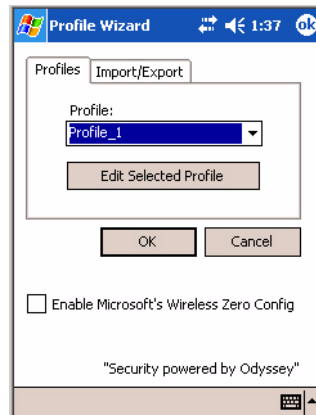


To start 802.11b/g communications on the CN2B Computer, tap **Start > Settings > the System tab > Wireless Network** to access the Profile Wizard for the 802.11b/g radio module.

A profile contains all the information necessary to authenticate you to the network, such as login name, password or certificate, and protocols by which you are authenticated.

You can have up to four profiles for different networks. For example, you may have different login names or passwords on different networks, or you may use a password on one network, and a certificate on another.

Use the Profiles page to select and configure between the networking environments assigned to this 802.11b/g radio.



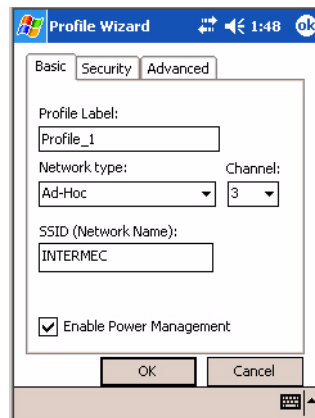
- **Profile:**
Tap the drop-down list to choose between four different profiles assigned to this unit, then tap **Edit Select Profile**, make the changes needed for this profile (*starting on the next page*), then tap **OK** to return to the Profiles page.
- **Enable Microsoft's Wireless Zero Config**
Check this box to enable Microsoft's Wireless Zero Config application. This effectively disables the Intermec software solution for 802.11b/g, including configuration via the Wireless Network applet.

Basic

Use the Basic page to set the network type, name, and manage battery power for this profile. Tap **ok** or **OK** to return to the Profiles page.

- **Profile Label:**
Enter a unique name for your profile.
- **Network type:**
Tap the drop-down list to select either "Infrastructure" if your network uses access points to provide connectivity to the corporate network or internet; or "Ad-Hoc" to set up a private network with one or more participants.
- **Channel:**
If you selected "Ad-Hoc" for the network type, select the channel on which you are communicating with others in your network. There are up to 11 channels available.
- **SSID (Network Name):**
This assumes the profile name *unless another name is entered in this field*. If you want to connect to the next available network or are not familiar with the network name, enter "ANY" in this field. Consult your LAN administrator for network names.

- **Enable Power Management:**
Check this box to conserve battery power (default), or clear this box to disable this feature.



Security

The following are available from the **802.1x Security** drop-down list. *Note that the last four methods are available if you have purchased the security package. Contact your Intermec representative for information.*

- None (*next paragraph*)
- PEAP (*page 198*)
- TLS (*page 201*)
- TTLS (*page 203*)
- LEAP (*page 207*)
- EAP-FAST (*page 209*)

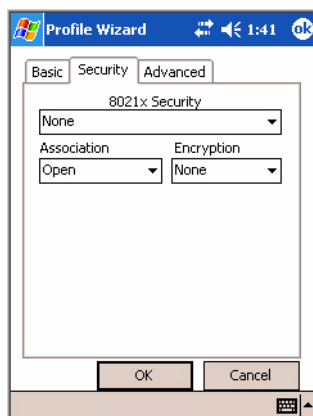
None

Use “None” to disable 802.11b/g Security and enable WEP encryption.

To Disable 802.1x Security

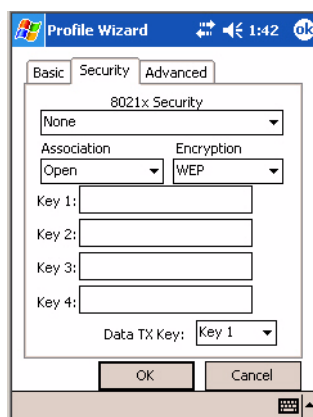
- 1 Set **802.1x Security** as “None.”
- 2 Set **Association** to “Open.”

3 Set Encryption to “None.”



To Enable WEP Encryption

- 1 Set 802.1x Security as “None.”
- 2 Set Association to either “Open” if WEP keys are not required; or “Shared” when WEP keys are required for association.
- 3 Set Encryption to “WEP.” See page 195 for information about WEP encryption.
- 4 If you had set Association to “Shared,” then select a data transmission key from the Data TX Key drop-down list near the bottom of this screen, then enter the encryption key for that data transmission in the appropriate Key # field.



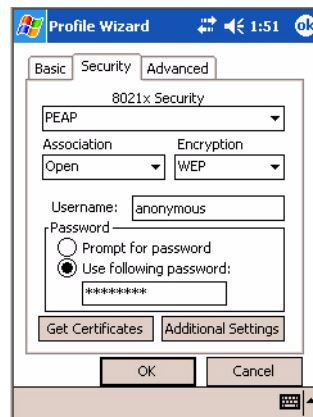
PEAP (Protected EAP)

This protocol is suitable for performing secure authentication against Windows domains and directory services. It is comparable to EAP-TTLS (see page 18), both in its method of operation and its security, though not as flexible. This does not support the range of inside-the-tunnel authentication methods supported by EAP-TTLS. Microsoft and Cisco both support this protocol.

Use “PEAP” to configure the use of PEAP as an authentication protocol and to select “Open,” “WPA,” or “Network EAP” as an association mode.

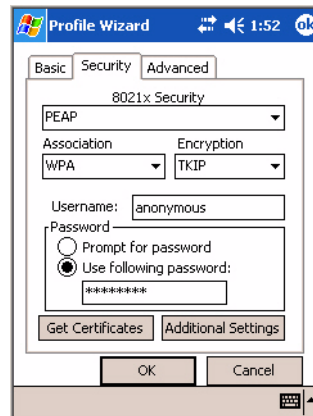
To Enable PEAP with an Open Association

- 1 Set **8021x Security** as “PEAP.”
- 2 Set **Association** to “Open.”
- 3 Skip **Encryption** as it is automatically set to “WEP.” See page 195 for information about WEP encryption.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol; or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 6 Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See page 201 for more information.

**To Enable PEAP with WPA Encryption**

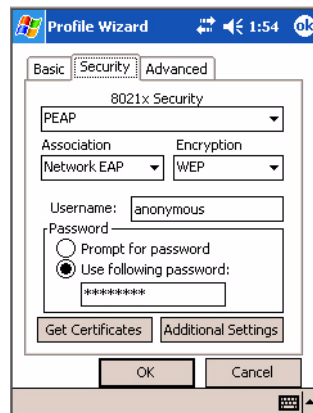
- 1 Set **8021x Security** as “PEAP.”
- 2 Set **Association** to “WPA.” See page 195 for information about WPA encryption.
- 3 Skip **Encryption** as it is automatically set to “TKIP.” See page 194 for more information about TKIP.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.

- 6 Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See page 201 for more information.



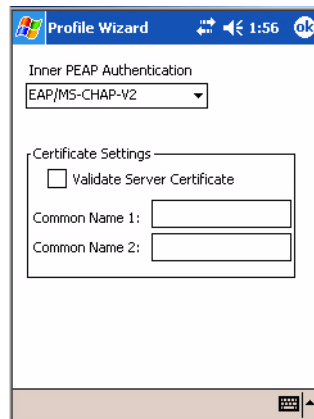
To Enable PEAP with Network EAP

- 1 Set **802.1x Security** as “PEAP.”
- 2 Set **Association** to “Network EAP.” See page 194 for information about EAP.
- 3 Set **Encryption** to either “WEP” or “CKIP.” See page 194 for information about CKIP and page 195 for information about WEP encryption.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 6 Tap **Additional Settings** to assign an inner PEAP authentication and set options for server certificate validation and trust. See below for more information.



Additional Settings

- 1 Select an authentication method from the **Inner PEAP Authentication** drop-down list.
 - **EAP/MS-CHAP-V2**
Authenticates against a Windows Domain Controller and other non-Windows user databases. This is Microsoft's implementation of PEAP.
 - **EAP/Token Card**
Use with token cards. The password value entered is never cached. This is Cisco's implementation of PEAP.
 - **EAP/MD5-Challenge**
Message Digest 5. A secure hashing authentication algorithm.
- 2 Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TTLS or PEAP.
- 3 Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.* Click **ok** to return to the Security page.



TLS (EAP-TLS)

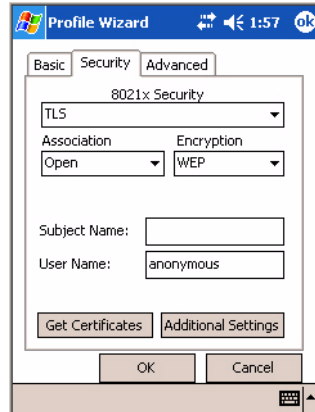
EAP-TLS is a protocol that is based on the TLS (Transport Layer Security) protocol widely used to secure web sites. This requires both the user and authentication server have certificates for mutual authentication. While cryptically strong, this requires corporations that deploy this to maintain a certificate infrastructure for all their users.

Use “TLS” to configure the use of EAP-TLS as an authentication protocol, and to select “Open,” “WPA,” or “Network EAP” as an association mode.

To Enable TLS with an Open Association

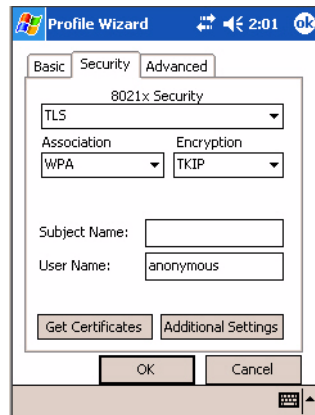
- 1 Set **8021x Security** as “TLS,” and **Association** to “Open.” Skip **Encryption** as it is automatically set to “WEP.” See page 195 for information about WEP encryption.
- 2 Enter your unique **Subject Name** and **User Name** to use this protocol.

- 3 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 4 Tap **Additional Settings** to set options for server certificate validation and trust. See page 203 for more information.



To Enable TLS with WPA Encryption

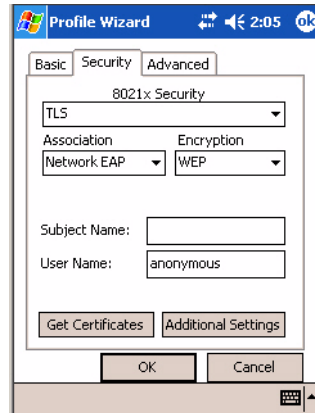
- 1 Set **802.1x Security** as “TLS” and **Association** to “WPA.” See page 195 for information about WPA encryption. Skip **Encryption** as it is automatically set to “TKIP.” See page 194 for more information about TKIP.
- 2 Enter a unique **Subject Name** and **User Name** as credentials.
- 3 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 4 Tap **Additional Settings** to set options for server certificate validation and trust. See page 203 for more information.



To Enable TLS with Network EAP

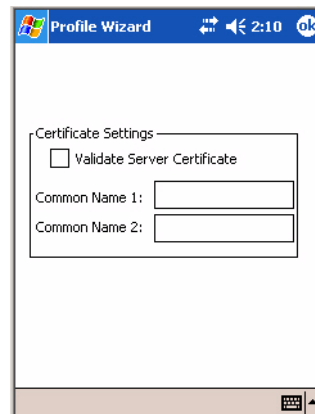
- 1 Set **802.1x Security** as “TLS” and **Association** to “Network EAP.” See page 194 for information about EAP.
- 2 Set **Encryption** to either “WEP” or “CKIP.” See page 194 for information about CKIP and page 195 for information about WEP encryption. Enter a unique **Subject Name** and **User Name** as credentials.

- 3 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information. Tap **Additional Settings** to assign an inner TLS authentication and set options for server certificate validation and trust. See page 203 for more information.



Additional Settings

- 1 Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TLS.
- 2 Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.*
- 3 Click **ok** to return to the Security page.



TTLS (EAP-Tunneled TLS)

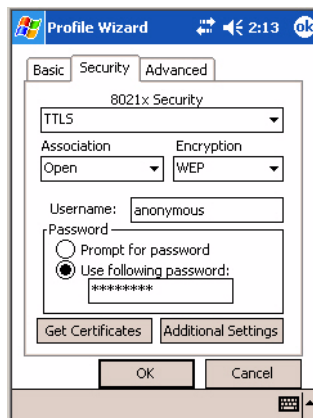
This protocol provides authentication like EAP-TLS (see page 201) but does not require certificates for every user. Instead, authentication servers are issued certificates. User authentication is done using a password or other credentials that are transported in a securely encrypted “tunnel” established using server certificates.

EAP-TTLS works by creating a secure, encrypted tunnel through which you present your credentials to the authentication server. Thus, inside EAP-TTLS there is another *inner authentication protocol* that you must configure via Additional Settings.

Use “TTLS” to configure the use of EAP-TTLS as an authentication protocol, and select “Open,” “WPA,” or “Network EAP” as an association mode.

To Enable TTLS with an Open Association (default configuration)

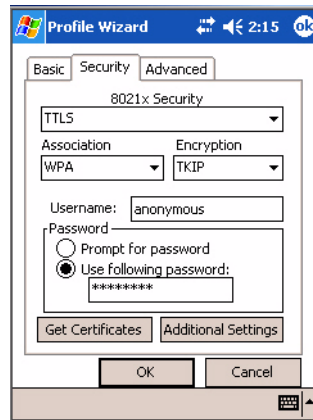
- 1 Set **8021x Security** as “TTLS.”
- 2 Set **Association** to “Open.”
- 3 Skip **Encryption** as it is automatically set to “WEP.” See page 195 for information about WEP encryption.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 6 Tap **Additional Settings** to assign an inner TTLS authentication and an inner EAP, and set options for server certificate validation and trust. See page 206 for more information.



To Enable TTLS with WPA Encryption

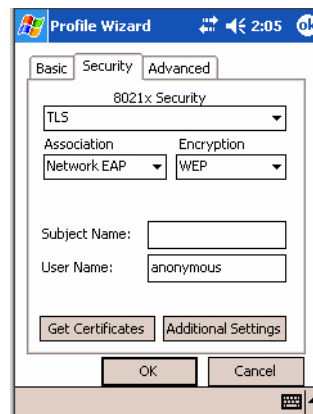
- 1 Set **8021x Security** as “TTLS.”
- 2 Set **Association** to “WPA.” See page 195 for information about WPA encryption.
- 3 Skip **Encryption** as it is automatically set to “TKIP.” See page 194 for more information about TKIP.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.

- 6 Tap **Additional Settings** to assign an inner TTLS authentication and an inner EAP, and set options for server certificate validation and trust. See page 206 for more information.



To Enable TTLS with Network EAP

- 1 Set **8021x Security** as “TTLS.”
- 2 Set **Association** to “Network EAP.” See page 194 for information about EAP.
- 3 Set **Encryption** to either “WEP” or “CKIP.” See page 194 for information about CKIP and page 195 for information about WEP encryption.
- 4 Enter your unique user name and password to use this protocol. Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 5 Tap **Get Certificates** to obtain or import server certificates. See page 207 for more information.
- 6 Tap **Additional Settings** to assign an inner TTLS authentication and set options for server certificate validation and trust. See below for more information.



Additional Settings

- 1 Select an authentication protocol from the **Inner TTLS Authentication** drop-down list:
 - **PAP** (Password Authentication Protocol)
A simple authentication protocol that sends security information in the clear.
 - **CHAP** (Challenge Handshake Authentication Protocol)
Use of Radius to authenticate a terminal without sending security data in the clear. Authenticates against non-Windows user databases. *You cannot use this if authenticating against a Windows NT Domain or Active Directory.*
 - **MS-CHAP, MS-CHAP-V2**
Authenticates against a Windows Domain Controller and other non-Windows user databases.
 - **PAP/Token Card**
Use with token cards. The password value entered is never cached.
 - **EAP** (Extensible Authentication Protocol)
See page 194 for information about EAP.
- 2 If you select “EAP” for the inner authentication protocol, then select an inner EAP protocol from the **Inner EAP** drop-down list.
- 3 Enter the **Common Names** of trusted servers. *Note that if these fields are left blank, the server certificate trust validation is not performed or required.*
- 4 Check **Validate Server Certificate** to verify the identity of the authentication server based on its certificate when using TTLS, PEAP, and TLS.
- 5 Enter the **Anonymous EAP-TTLS Name** as assigned for public usage. Use of this outer identity protects your login name or identity.
- 6 Click **ok** to return to the Security page.

The screenshot shows the 'Profile Wizard' dialog box with the following settings:

- Inner TTLS Authentication:** EAP
- Inner EAP:** EAP/MD5-Challenge
- Certificate Settings:**
 - ☐ Validate Server Certificate
 - Common Name 1:** (empty field)
 - Common Name 2:** (empty field)
- Anonymous EAP-TTLS Name:** anonymous

The dialog box has a title bar with 'Profile Wizard', a help icon, a back icon, a forward icon, a clock showing '2:20', and an 'ok' button. At the bottom right, there is a keyboard icon and a mouse icon.

To Get Certificates

Certificates are pieces of cryptographic data that guarantee a public key is associated with a private key. They contain a public key and the entity name that owns the key. Each certificate is issued by a certificate authority.

Use this page to import a certificate onto the CN2B Computer.

Root Certificates

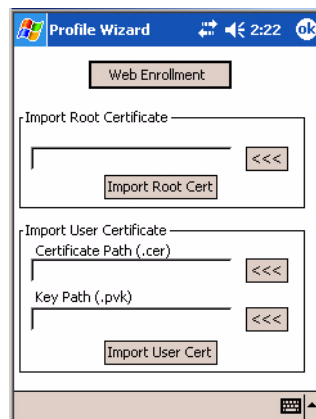
- 1 Tap the <<< button next to the **Import Root Certificate** field to select the root certificate (DER-encoded .CER file) to import.
- 2 Click **Import Root Cert** to install the selected certificate.

User Certificate

- 1 Tap the <<< button next to the **Certificate Path** field to select the user certificate (DER-encoded .CER file without the private key) to import.
- 2 Tap the <<< button next to the **Key Path** field to select the private key (.PVK file) which corresponds to the user certificate chosen in step 1.
- 3 Tap **Import User Cert** to install the selected certificate.

Web Enrollment

Tap **Web Enrollment** to obtain a user certificate over the network from an IAS Server. Tap **ok** to return to the Security page.



LEAP (Cisco Lightweight EAP)

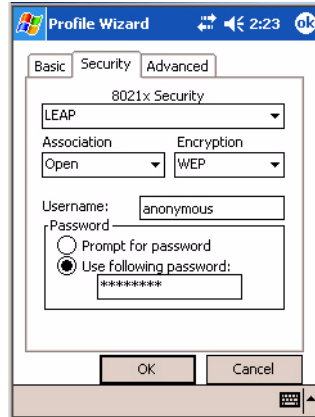
LEAP is the Cisco Lightweight version of EAP. See page 194 for information about EAP.

Use “LEAP” to configure the use of LEAP as an authentication protocol, select “Open,” “WPA,” or “Network EAP” as an association mode, or assign Network EAP. *Note that this defaults to the Network EAP.*

To Enable LEAP with an Open Association

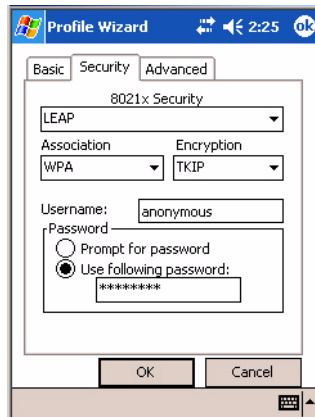
- 1 Set **8021x Security** as “LEAP.”
- 2 Set **Association** to “Open.”
- 3 Skip **Encryption** as it is automatically set to “WEP.” See page 195 for information about WEP encryption.

- 4 Enter your unique **User Name** to use this protocol.
- 5 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.



To Enable LEAP with WPA Encryption

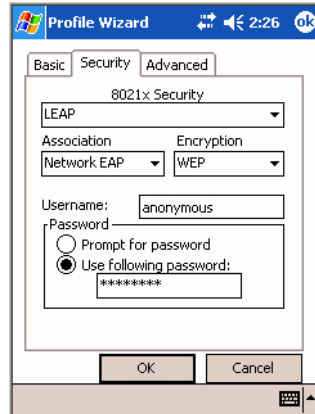
- 1 Set **802.1x Security** as “LEAP.”
- 2 Set **Association** to “WPA.” See page 195 for information about WPA encryption.
- 3 Skip **Encryption** as it is automatically set to “TKIP.” See page 194 for more information about TKIP.
- 4 Enter your unique **User Name** to use this protocol.
- 5 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.



To Enable LEAP with Network EAP

- 1 Set **802.1x Security** as “LEAP” and **Association** to “Network EAP,” an EAP protocol for the network. See page 194 for information about EAP.
- 2 Set **Encryption** to either “WEP” or “CKIP.” See page 194 for information about CKIP and page 195 for information about WEP encryption.

- 3 Enter your unique **User Name** to use this protocol.
- 4 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.



EAP-FAST (EAP-Flexible Authentication via Secured Tunnel)

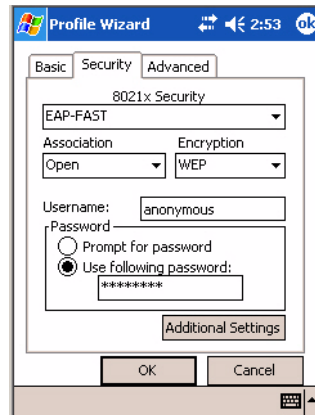
The EAP-FAST protocol is a client-server security architecture that encrypts EAP transactions with a TLS tunnel. While similar to PEAP in this respect, it differs significantly in that EAP-FAST tunnel establishment is based upon strong secrets that are unique to users. These secrets are called Protected Access Credentials (PACs), which CiscoSecure ACS generates using a master key known only to CiscoSecure ACS. Because handshakes based upon shared secrets are intrinsically faster than handshakes based upon PKI, EAP-FAST is the significantly faster of the two solutions that provide encrypted EAP transactions. No certificate management is required to implement EAP-FAST.

Use “EAP-FAST” to configure the use of EAP-FAST as an authentication protocol, select “Open,” “WPA,” or “Network EAP” as an association mode.

To Enable EAP-FAST with an Open Association

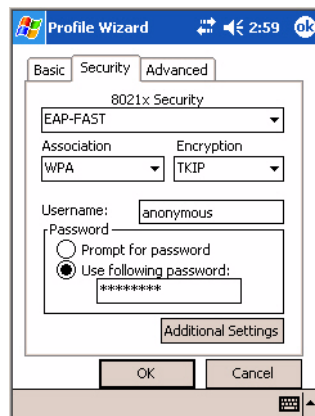
- 1 Set **802.1x Security** as “EAP-FAST.”
- 2 Set **Association** to “Open.”
- 3 Skip **Encryption** as it is automatically set to “WEP.” See page 195 for information about WEP encryption.
- 4 Enter your unique **User Name** to use this protocol.
- 5 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.

- 6 Tap **Additional Settings** to set options for PAC management and assign an anonymous EAP-FAST name. See page 211 for more information.



To Enable EAP-FAST with WPA Encryption

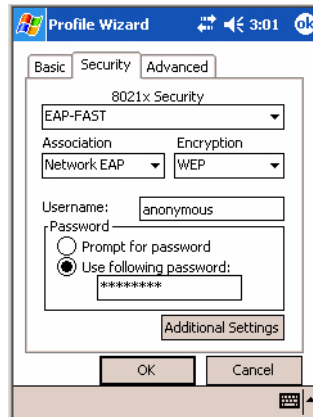
- 1 Set **8021x Security** as “EAP-FAST.”
- 2 Set **Association** to “WPA.” See page 195 for information about WPA encryption.
- 3 Skip **Encryption** as it is automatically set to “TKIP.” See page 194 for more information about TKIP.
- 4 Enter your unique **User Name** to use this protocol.
- 5 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 6 Tap **Additional Settings** to set options for PAC management and assign an anonymous EAP-FAST name. See page 211 for more information.



To Enable EAP-FAST with Network EAP

- 1 Set **8021x Security** as “EAP-FAST.”
- 2 Set **Association** to “Network EAP,” an EAP protocol for the network. See page 194 for information about EAP.

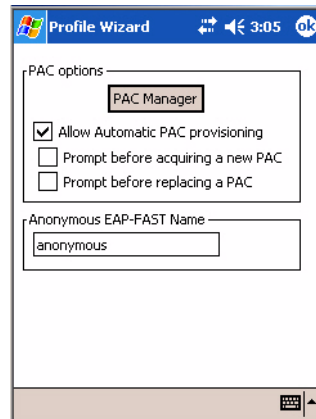
- 3 Set **Encryption** to either “WEP” or “CKIP.” See page 194 for information about CKIP and page 195 for information about WEP encryption.
- 4 Enter your unique **User Name** to use this protocol.
- 5 Select **Prompt for password** to have the user enter this password each time to access the protocol, or leave **Use following password** as selected to automatically use the protocol without entering a password.
- 6 Tap **Additional Settings** to set options for PAC management and assign an anonymous EAP-FAST name. See next page for more information.



Additional Settings

- 1 Tap **PAC Manager** to view the PAC files currently installed on your CN2B. Tap **ok** to return to the Additional Settings screen.
- 2 If you already have a PAC on your CN2B, clear **Allow Automatic PAC provisioning** to avoid receiving additional PACs from the server.
- 3 If **Allow Automatic PAC provisioning** is checked, and you want notification of any incoming PACs, then check **Prompt before acquiring a new PAC**.
- 4 If **Allow Automatic PAC provisioning** is checked, and you want notification whether to replace a current PAC with an incoming PAC, check **Prompt before replacing a PAC**.
- 5 Enter the **Anonymous EAP-FAST Name** as assigned for public usage. Use of this outer identity protects your login name or identity.

6 Click **ok** to return to the Security page.



Advanced

Use this page to configure additional settings for this profile. Tap **ok** to return to the Profiles page.

- **Detect Rogue APs (Access Points):**
Wireless NICs and access points associate based on the SSID configured for the NIC. Given an SSID, the BSSID with the strongest signal is often chosen for association. After association, 802.1x authentication may occur and during authentication credentials to uniquely identify a user — these are passed between the NIC and the access point.

The base 802.1x technology does not protect the network from “rogue APs.” These can mimic a legitimate access point to authentication protocols and user credentials. This provides illegal users ways to mimic legitimate users and steal network resources and compromise security.

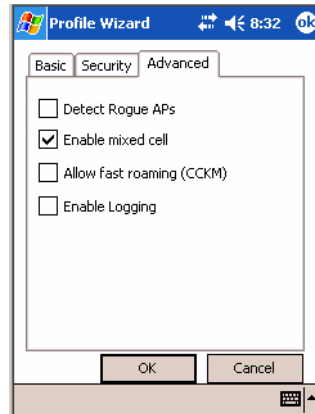
Check this box to detect and report client behavior suspected of being rogue access points. Once detected, your CN2B Computer no longer associates with that access point until you perform a warm-boot.

Clear this box to solve access point connection problems that result when an access point gets put on the rogue access point list due to inadvertant failed authentications and *not* because it is a *real* rouge.
- **Enable mixed cell:**
Mixed cell is a profile-dependent setting. If enabled, you can connect to mixed cell without using WEP, then you can query the cell to determine whether you can use encryption.
- **Allow fast roaming (CCKM):**
When using a wireless LAN that uses Cisco Access Points, a LEAP-enabled client device can roam from one access point to another without involving the authentication (RADIUS) server. If enabled, Cisco Centralized Key Management (CCKM), an access point configured to provide Wireless Domain Services (WDS) takes the place of the RADIUS server (caching credentials of an initial authentication with the RADIUS

server) and authenticates the client without perceptible delay in voice or other time-sensitive applications.

- **Enable Logging:**

Check this box to log what activity incurs for this profile.



Other Configurable Parameters

Configure the following parameters by sending reader commands through the network or from an application. See next page for information.

Reader Command	Description	Option
Audio Volume	Changes the volume of all audio signals.	0 - Off 1 - Very quiet 2 - Quiet 3 - Normal (default) 4 - Loud 5 - Very loud
Automatic Shutoff	Sets the length of time the CN2B Computer remains on with no activity. When you turn on the CN2B Computer, it either resumes exactly where it was when you turned it off or boots and restarts your application.	1 - 1 minute 2 - 2 minutes 3 - 3 minutes (default) 4 - 4 minutes 5 - 5 minutes
Backlight Timeout	Sets the length of time that the display backlight remains on. If you set a longer timeout value, you use the battery power at a faster rate.	10 - 10 seconds 30 - 30 seconds 60 - 1 minute (default) 120 - 2 minutes 180 - 3 minutes 240 - 4 minutes 300 - 5 minutes
Date/Time	Sets the current date and time.	Date Year - 0000–9999 (1999) Month - 1–12 (6) Day - 1–31 (1) Time Hour - 0–23 (0) Minute - 0–59 (00) Second - 0–59 (00)

Reader Command	Description	Option
Key Clicks	Enables or disables the keypad clicks. The CN2B Computer emits a click each time you press a key or decode a row of a two-dimensional symbology.	0 - Disable clicks 1 - Enable soft key clicks 2 - Enable loud key clicks (default)

Using Reader Commands

After the CN2B Computer is connected to your network, you can send the CN2B Computer a reader command from an application to perform a task, such as changing the time and date. Some reader commands temporarily override configuration settings and some change the configuration settings.

Change Configuration

The Change Configuration command must precede any configuration command. If you enter a valid string, the CN2B Computer configuration is modified and the computer emits a high beep. To send the Change Configuration command through the network, use the `$+ [command]` syntax where *command* is the two-letter command syntax for the configuration command followed by the value to be set for that command.

You can also make changes to several different commands by using the `$+ [command] . . . [command n]` syntax. There are seven configuration command settings that you can change in this way. *See each command for information on respective acceptable “data” values.*

Command	Syntax
Audio Volume	BV <i>data</i>
Automatic Shutoff	EZ <i>data</i>
Backlight Timeout	DF <i>data</i>
Key Clicks	KC <i>data</i>
Virtual Wedge Grid	AF <i>data</i>
Virtual Wedge Postamble	AE <i>data</i>
Virtual Wedge Preamble	AD <i>data</i>

Example 1

To change the Beep Volume to Off, you can send this string to the CN2B Computer through the network: `$+BV0`

where:

<code>\$+</code>	Indicates Change Configuration.
<code>BV</code>	Specifies the Audio Volume parameter.
<code>0</code>	Specifies a value of Off.

Example 2

To change the Beep Volume to Very Quiet and the Virtual Wedge Grid to 123:

```
$+BV1AF123:
```

\$+	Indicates Change Configuration
BV1	Specifies Audio Volume, set to Very Quiet (1)
AF123	Specifies Virtual Wedge Grid, set to a value of 123.

Set Time and Date

This command sets the date and time on the CN2B Computer. The default date and time is *June 1, 1999 at 12:00 AM*.

From the network, send the following:

```
/+ yyyymmddhhmmss
```

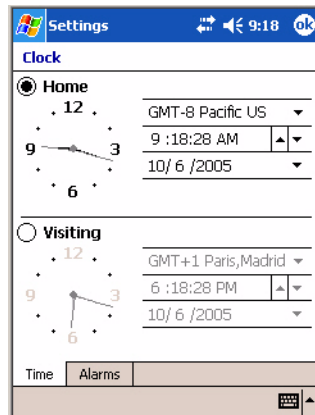
where acceptable values for the date are:

yyyy	0000-9999	Year
mm	01-12	Month of the year
dd	01-31	Day of the month
hh	00-23	Hour
mm	00-59	Minutes
ss	00-59	Seconds



Clock

You can also set the time and date by using the Clock applet in the Settings menu. To access this applet, tap **Start > Settings > the System tab > the Clock icon**.

**Configuration Bar Codes**

You can change some settings on your CN2B Computer by scanning the following Code 39 bar code labels.



Note: When you use a bar code creation utility to make a scannable bar code label, the utility probably adds opening and closing asterisks automatically. Asterisks are included here for translation purposes.

Audio Volume



Note: The Audio Volume parameter information is on page 213.

Turn Audio Off



\$+BV0

Set Audio Volume to very quiet



\$+VB1

Set Audio Volume to quiet



\$+BV2

Set Audio Volume to normal (*default*)



\$+BV3

Set Audio Volume to loud



\$+BV4

Set Audio Volume to very loud



\$+BV5

Automatic Shutoff



Note: The Automatic Shutoff parameter information is on page 213.

Set Automatic Shutoff to 1 minute



\$+EZ1

Set Automatic Shutoff to 2 minutes



\$+EZ2

Set Automatic Shutoff to 3 minutes (*default*)



\$+EZ3

Set Automatic Shutoff to 4 minutes



\$+EZ4

Set Automatic Shutoff to 5 minutes



\$+EZ5

Backlight Timeout



Note: The Backlight Timeout parameter information is on page 213.

Backlight Timeout 10 seconds



\$+DF10

Backlight Timeout 30 seconds



\$+DF30

Backlight Timeout 1 minute (*default*)



\$+DF60

Backlight Timeout 2 minutes



\$+DF120

Backlight Timeout 3 minutes



\$+DF180

Backlight Timeout 4 minutes



\$+DF240

Backlight Timeout 5 minutes



\$+DF300

Key Clicks



Note: The Key Clicks parameter information is on page 214.

Disable key clicks



\$+KC0

Enable soft key clicks



\$+KC1

Enable loud key clicks (*default*)



\$+KC2

Virtual Wedge Grid, Preamble, Postamble

The following parameters are user-configurable strings. Refer to a full ASCII chart for more information.

Grid

For Virtual Wedge Grid, the first part of the bar code would be the following, which can include a string of up to 240 characters.



*\$+AF

Preamble

For Virtual Wedge Preamble, the first part of the bar code would be below, followed by a string of up to 31 characters (*no <NUL>*) and an asterisk.



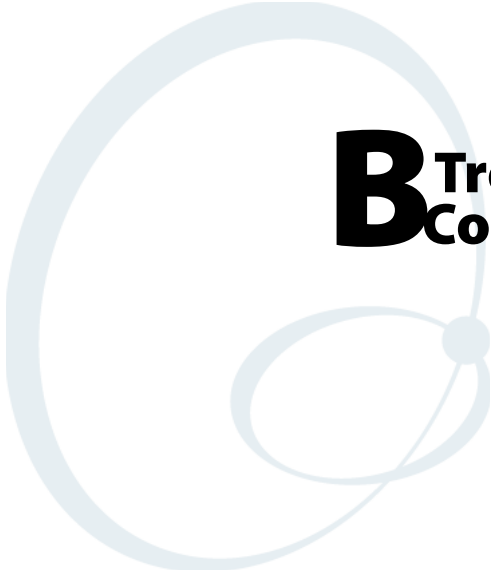
*\$+AD

Postamble

For Virtual Wedge Postamble, the first part of the bar code would be below, followed by a string of up to 31 characters (*no <NUL>*) and an asterisk.



*\$+AE



B Troubleshooting the CN2B Computer

Use this appendix to solve problems you may encounter while using the CN2B Computer.

Problems and Solutions

These tables offer solutions to the problems you may encounter.

Problems While Operating the CN2B Computer

Problem	Solution
You press Power to turn on the CN2B and nothing happens.	Try these possible solutions in order: 1 Make sure the battery door is installed correctly and completely closed. 2 Make sure you have a charged battery installed correctly. 3 The battery may be discharged. Replace the battery with a spare charged battery, or charge the battery and try again. 4 Perform a warm-boot on the CN2B Computer.
You press Power to turn off the CN2B and nothing happens.	To turn off (or suspend) the CN2B Computer, hold the Power key for 2 or 3 seconds, then release it. If the CN2B Computer is processing data, it may not turn off when you press Power . Wait until the CN2B Computer finishes processing. If the CN2B Computer appears to be locked up, perform a warm-boot on the CN2B Computer. If the CN2B Computer does not respond to a warm-boot, perform a cold-boot.
The CN2B Computer is not responding to the stylus.	Press and hold the Power key for 2 to 3 seconds, then release it to turn off the CN2B Computer. Press Power again to turn on the CN2B Computer.
You place the CN2B Computer in the communications dock, and the Battery light turns on and is orange.	• The temperature may not be within the charging range. Make sure that the temperature is from 0°C to 45°C (32°F to 113°F). • The battery may be damaged. Replace the battery.
The CN2B Computer appears locked up and you cannot enter data.	Try these possible solutions in order: 1 Wait at least 10 seconds and try again. If the CN2B Computer is still connecting to the Intermec Application Server or the host, it ignores any input from the keypad or scanner. 2 Press and hold the Power key for 2 to 3 seconds, and then release it to turn off the CN2B Computer. Press Power again to turn on the CN2B Computer. 3 Perform a warm-boot on the CN2B Computer. 4 Perform a cold-boot on the CN2B Computer. 5 Try reloading the firmware. 6 If the CN2B Computer does not boot or reset, contact your local Intermec service representative for help.

Problems While Configuring Security

Problem	Solution
The CN2B Computer does not appear to be authenticating.	The CN2B Computer may not be communicating with your access point. Make sure the network name on the CN2B Computer is the same as the network name (SSID) of the access point that you are trying to communicate with. The default network name is “INTERMEC.” The 802.1x security network may not be active. Make sure that the server software is properly loaded and configured on the server PC. For help, see the documentation that shipped with your server software. The access point that you are trying to communicate with may not be communicating with the server. Make sure your access point is turned on, properly configured, and has 802.1x security enabled.
The CN2B Computer indicates that it is not authenticated.	Make sure that: • The User Name and Password parameters on your CN2B Computer match the user name and password on your authentication server. You may need to re-enter the password on both your CN2B Computer and the authentication server. • On your authentication server, the user and group are allowed and the group policy is allowed to log in to the server. For help, see the documentation for your authentication server software. • The IP address and secret key for your access point must match the IP address and secret key on your authentication server. You may need to re-enter the IP address and secret key on both your access point and authentication server. • The authentication server software is running on the server PC.
The CN2B Computer indicates that it is authenticated, but it does not communicate with the host.	Make sure that the CN2B IP address, host IP address, subnet mask, and default router are properly configured for your network.
You are setting up multiple access points in a network, with different SSIDs, and the connection fails.	The CN2B Computer does not save WEP key values when you change the SSID. Re-enter the WEP key value after you change the SSID and save your changes. You should now be able to connect to the different access points.

Problems with Wireless Connectivity

Problem	Solution
A Network Connection icon appears on the status bar, but the host computer is not receiving any data from the CN2B Computer.	In a UDP Plus network, there may be a problem with the connection between the Intermec Application Server and the host computer. Check with your network administrator or see the user’s manual for the Intermec Application Server. In a TCP/IP network, there may be a problem with the connection between the access point and the host computer. Check with your network administrator or use your access point user’s manual.
When you turn on the CN2B Computer after it was suspended for a while (10-15 minutes or longer), it can no longer send or receive messages over the network.	The CN2B Computer may not recognize the network card. Turn off the CN2B Computer, and then turn it on again. The host may have deactivated or lost your current terminal emulation session. In a TCP/IP direct connect network, you need to turn off the “Keep Alive” message (if possible) from the host so that the TCP session is maintained while a CN2B Computer is suspended.

Problems with Wireless Connectivity (continued)

Problem	Solution
The CN2B Computer is connected to the Intermec Application Server or host computer and you move to a new site to collect data. A Network Connection icon was visible but now the no network connection icon is visible.	You may have gone out of range of an access point. Try moving closer to an access point or to a different location to re-establish communications. Once you are in range again, the network connection icon appears again. Any data you collected while out of range is transmitted over the network.
The no network connection icon appears on the status bar.#	The no network connection icon appears in three situations: • The CN2B Computer may not have an IP address. You must configure an IP address for the CN2B Computer or make sure that DHCP assigned an address. Use Intermec Settings and select the radio tab to make sure an IP address is assigned. • The CN2B Computer may not be connected to the access point. Try these possible solutions in order: • Make sure the access point is turned on and operating. • Make sure you are not using the CN2B Computer out of range of an access point. Try moving closer to an access point to re-establish communications. • Make sure the CN2B Computer is configured correctly for your network. The radio parameters on the CN2B Computer must match the values set for all access points the CN2B Computer may communicate with. • If you have an 802.11b/g radio, the radio initialization process may have failed. Try resetting the CN2B Computer. • If you have tried these possible solutions and the no network connection icon still appears, you may have a defective radio card. For help, contact your local Intermec representative.

Problems While Scanning Bar Codes

Problem	Solution
You cannot see a red beam of light from the scanner when you press the Scan button or one of the Side Scan buttons and aim the scanner at a bar code label.	There are three possible problems: • You may be too far away from the bar code label. Try moving closer to the bar code label and scan it again. • You may be scanning the bar code label “straight on.” Change the scanning angle and try again. • The PSM files may not be correctly installed. You can test the effective range of the scanner. Move within 61 cm (2 ft) of a wall and test the scanner. You need to be within the scanning range to scan bar code labels.
When you release the Scan button or Side Scan button, the Good Read light does not turn off.	The Good Read light remains on if you configure the CN2B Computer to use continuous/edge triggering. If you configure the CN2B Computer for level triggering and the Good Read light remains on, there may be a problem. Press the Scan button or one of the Side Scan buttons again without scanning a bar code label. If the light is still on, contact your local Intermec service representative.

Problems While Scanning Bar Codes (continued)

Problem	Solution
The scanner will not read the bar code label.	Make sure you aim the scanner beam so it crosses the entire bar code label in one pass. The angle you are scanning the bar code label may not be working well, or you may be scanning the label “straight on.” Try scanning the bar code label again, but vary the scanning angle. The bar code label print quality may be poor or unreadable. To check the quality of the bar code label, try scanning a bar code label that you know will scan. Compare the two bar code labels to see if the bar code quality is too low. You may need to replace the label that you cannot scan. Make sure the bar code symbology you are scanning is enabled. Use Intermec Settings to check the symbologies. If your bar code symbology is disabled, enable it and then try scanning the bar code label again. Note: If you restored the CN2B Computer to factory default settings, some of the symbologies may be disabled. Make sure that the application you are running on the computer is expecting input from a bar code. You may need to use the input panel to enter this information instead of scanning it.
The scanner does not read the bar code labels quickly, or the scanning beam seems faint or obscured.	The scanner window may be dirty. Clean the window with a solution of ammonia and water. Wipe dry. Do not allow abrasive material to touch the window.
You scan a valid bar code label to enter data for your application. The data decoded by the CN2B Computer does not match the data encoded in the bar code label.	The computer may have decoded the bar code label in a symbology other than the label’s actual symbology. Try scanning the bar code label again. Make sure you scan the entire label. To operate the computer quickly and efficiently, you should only enable the bar code symbologies that you are going to scan.

Sending the CN2B Computer to Intermec for Service

If you send the CN2B Computer in for service, it is your responsibility to save the computer data and configuration. Intermec is responsible only for ensuring that the keypad and other hardware features match the original configuration when repairing or replacing your computer.

For help understanding your warranty and finding help, see “Global Services and Support” on page xi.

You may be asked for the version of the operating system running on your CN2B Computer. For help finding this information, see “Software Build Version” on page 14

Cleaning the Scanner Window and the Touch Screen

To keep the computer in good working order, you may need to clean the scanner window and touch screen with a solution of ammonia and water.

You can clean the scanner window and the touch screen as often as needed for the environment in which you are using the computer. You can help keep the touch screen clean by using the stylus, instead of your fingertip, to tap the screen.



There are no user-serviceable parts inside the CN2B Computer. Opening the unit voids the warranty and may cause damage to the internal components.

To clean the scanner window and touch screen

- 1 Press and hold the **Power** key for 2 to 3 seconds, and then release it to turn off the CN2B Computer.
- 2 Dip a clean towel or rag in the ammonia solution and wring out the excess. Wipe off the scanner window and touch screen. Do not allow any abrasive material to touch these surfaces.
- 3 Wipe dry.

To clean the scanner window

- You can use the Screen Cleaner Kit (P/N 346-065-101) to clean the scanner window.



Symbols

__RESETMEPLEASE__.TXT, 136

Numerics

6820 printers

 NPCP driver, 114

 printer support, 113

6920 Communications Server

 ManifestName parameter, 144

802.11

 API, 161

 channel, 196

 communications setup, 94, 195

 configuration profiles, 162

 EAP-FAST

 network EAP, 210

 WPA encryption, 210

 LEAP

 network EAP, 208

 WPA encryption, 208

 network type, 196

 PEAP

 network EAP, 200

 WPA encryption, 199

 profile label, 196

 profile security information

 WEP encryption, 198

 profiles, 196

 advanced settings, 212

 basic information, 196

 security information, 197

 SSID (network name), 196

 TLS

 network EAP, 202

 TTLS

 network EAP, 205

 WPA encryption, 204

 WPA authentication with pre-shared key

 Zero Configuration, 98

 zero configuration

 WEP encryption, 97

802.11b/g communications, 93

80211API.DLL, 162

80211CONF.EXE, 162

80211SCAN.EXE, 162

802PM.DLL, 162

A

abrasive material, avoiding, 224

Abstract Syntax Notation.1 See ASN.1

Accessory list, 15

Accounts

 via Inbox, 58

ActiveSync

 adding programs, 33

 Folder behavior connected to email server, 56

 installing applications, 77

 Microsoft Reader, 69

 Pocket Internet Explorer

 favorite links, 71

 mobile favorites, 72

 Mobile Favorites folder, 71

 replicating registry settings, 79

 Start menu icon, 22

 URL, 35

 Windows Mobile, 35

add_registry_section

 AddReg

 flags, 132

 registry_root_string, 132

 value_name, 132

Adding programs

 ActiveSync, 33

 to the Start menu, 34

 via File Explorer, 34

 Windows Mobile, 32

AddReg

 add_registry_section

 flags, 132

 registry_root_string, 132

 value_name, 132

 DefaultInstall, 129

AddWep(), 171

Adjusting settings

 Windows Mobile, 32

AllDay events

 Calendar

 creating, 40

ammonia and water for cleaning, 224

Annotations index

 Microsoft Reader, 70

APIs

 802.11, 161

Applets

 clock, 32, 215

 intemec settings

 beeper volume, 9, 109

 intermec settings, 191

 smartsystems, 9, 109

 menu, 32

 owner information, 32

 password, 32

 power, 32

 battery status, 3

 RAM maintenance, 4

 system

 wireless network, 94, 195

- today, 32
- utilities, 191
 - app launch, 193
 - registry save, 191
 - wakeup mask, 192
 - wireless network, 194
- Application keys
 - utilities applet, 192, 193
- Application launch applet, 193
- AppName
 - CEStrings, 127
- Appointments
 - Calendar
 - assigning to a category, 43
 - changing, 39
 - creating, 39
 - setting a reminder, 41
 - viewing, 39
- ASCII
 - printing, 114
 - printing to a port
 - port print method, 114
 - raw text to printer, 114
- ASN.1, 109
- Asset management
 - DeviceURL parameter, 143
- Attaching notes to text
 - Microsoft Reader, 70
- Audio files
 - Windows Media Player, 68
- Audio system
 - external headset jack, 3
 - microphone, 2
 - speaker, 2
- AutoCab
 - command line syntax, 83
- AutoFTP, 149
- AutoIP, 100
- Automatic Private IP See AutoIP
- Automatic shutoff
 - bar code configuration, 213, 216
 - configuration parameter, 213
- Autostart FTP, 149
- AUTOUSER.DAT, 78
- AvantGo channels
 - Pocket Internet Explorer, 73
- B**
- Backlight timeout
 - bar code configuration, 213, 217
 - configuration parameter, 213
- Bar code configuration
 - audio volume, 213
 - automatic shutoff, 213
 - backlight timeout, 213
 - key clicks, 214
- Bar codes
 - configuration
 - audio volume, 216
 - automatic shutoff, 216
 - backlight timeout, 217
 - Code 39, 215
 - key clicks, 218
 - internal scanner supported symbologies, 124
 - scanning labels, 215
- Basic connect/disconnect functions, 162
- Battery
 - capacity, 17
 - low battery conditions, 3
 - RAM maintenance, 4
 - specifications, 17
 - status, 3
- Beeper
 - silencing the volume, 8
 - volume
 - turning it on, 8
- BlockSize
 - FTP Server, 143
- Bluetooth
 - accessing, 86
 - activating, 86
 - connecting with remote devices, 89
 - WPport, 87
- Bluetooth compatibility
 - network support, 86
- Books
 - Microsoft Reader
 - adding bookmarks, 70
 - adding drawings, 70
 - annotations index, 70
 - attaching notes, 70
 - copying, 70
 - highlighting, 70
 - reading, 70
 - removing, 71
 - searching, 70
- Browsing the Internet
 - Pocket Internet Explorer, 74
- Build information
 - software, 14
- BuildMax
 - CEDevice, 128
- BuildMin
 - CEDevice, 128

C

- CAB files
 - after the extraction, 136
 - creating, 126
 - creating INF files, 126
 - creating with CAB Wizard, 139
 - installation functions
 - SETUP.DLL, 136
- Cabinet Wizard
 - creating CAB files, 139
 - troubleshooting, 140
 - using the application, 126
- CABWIZ.EXE, 126
- Calendar
 - all day events
 - creating, 40
 - appointments
 - assigning to a category, 43
 - changing, 39
 - creating, 39
 - setting a reminder, 41
 - viewing, 39
 - categories, 37
 - options
 - changing, 45
 - recurrence pattern, 38
 - Start menu icon, 22
 - synchronizing, 37
- Capacitor, internal super, 3
- Card support
 - radios, 15
- Categories
 - calendar, 37
- CEDevice
 - BuildMax, 128
 - BuildMin, 128
 - ProcessorType, 127
 - UnsupportedPlatforms, 127
 - VersionMax, 128
 - VersionMin, 128
- CESelfRegister
 - DefaultInstall, 129
- CESetupDLL
 - DefaultInstall, 129
- CEShortcuts
 - DefaultInstall, 129
 - shortcut_list_section
 - shortcut_filename, 133
 - shortcut_type_flag, 133
 - target_file_path, 133
- CESignature
 - SourceDiskNames, 129
 - Version, 126
- CEStrings
 - AppName, 127
 - InstallDir, 127
- Channel
 - 802.11 radio module, 196
- ClassID field values
 - VN_CLASS_ASIC, 152
 - VN_CLASS_BOOTSTRAP, 152
 - VN_CLASS_KBD, 152
- cleaning the scanner window and touch screen, 224
- Clock applet
 - setting date and time, 215
 - Windows Mobile settings, 32
- CloseHandle()
 - DTR printing, 119, 120
 - IrDA printing, 114
 - NPCP printing, 114, 116
- Closing drivers
 - NPCP, 116
- Cold boot
 - IOCTL_HAL_COLDBOOT, 157
- Cold boot, performing, 13
- COM1
 - NPCP parameter, 115
- COM1 port, 114
- Command line syntax
 - AutoCab, 83
- Communications
 - DTR, 120
 - NPCP, 117
- CompactFlash cards
 - installing applications, 78
- Computer shutdown, 3
- Configuration parameters
 - automatic shutoff, 213
 - backlight timeout, 213
 - date/time, 213
 - key clicks, 214
 - volume, 213
- ConfigureProfile(), 176
- Connecting to
 - an ISP, 101
 - email server, 109
- Connecting to a mail server
 - via Inbox, 57
- Connections
 - directly to email server, 109
 - ending, 108
 - to an ISP, 101
 - via modem, 101
 - via modem
 - to an ISP, 101
- Connections See Getting connected, 101

- Contacts
 - adding to speed dial, 51
 - changing options, 51
 - copying, 49
 - creating, 46
 - MSN Messenger
 - managing, 67
 - sending messages, 68
 - working with, 67
 - sending a message, 49
 - Start menu icon, 22
 - synchronizing, 47
 - viewing, 47
- Control panel applets
 - intermec settings, 191
- Converting writing to text, 28
- CopyFiles
 - file_list_section
 - destination_filename, 131
 - flags, 131
 - source_filename, 131
- Copyfiles
 - DefaultInstall, 129
- Copying
 - contacts, 49
- COREDLL.DLL, 181
- CPL802.CPL, 162
- CreateEvent(), 184
- CreateFile()
 - DTR printing, 119, 120
 - IrDA printing, 114
 - NPCP printing, 114, 115
- D**
- Date, setting, 215
- Date/Time
 - configuration parameter, 213
- DefaultInstall
 - AddReg, 129
 - CESelfRegister, 129
 - CESetupDLL, 129
 - CEShortcuts, 129
 - Copyfiles, 129
- DeregisterDevice(), 115
 - DTR printing, 119
- DestinationDirs
 - file_list_section, 130
- Detect rogue APs, 212
- DEVICEID.H, 154
- DeviceIOControl(), 161
 - DTR printing, 119
 - NPCP printing, 114
- DeviceIoControl()
 - NPCP printing, 116
- DeviceName
 - FTP Server, 143
- DeviceURL
 - FTP Server, 143
- DHCP, 100
 - replicating registry settings, 79
- disk_ordinal
 - SourceDiskNames, 129
- Display specifications, 16
- DllRegisterServer, 129
- DllUnregisterServer, 129
- Documents
 - creating via Pocket Word, 60
- DRAM
 - low battery shutdown, 4
 - maintenance, 4
- Drawing
 - creating, 29
- Drawing mode
 - Pocket Word, 63
- Drawing on the screen
 - Pocket Word, 63
- Drawings
 - adding via Microsoft Reader, 70
- Drivers
 - DTR
 - communications, 120
 - installing, 119
 - opening, 120
 - removing, 119
 - writing to, 120
 - NPCP
 - closing, 116
 - communications, 117
 - I/O controls, 116
 - installing, 115
 - opening, 115
 - reading from, 116
 - removing, 115
 - writing to, 116
- DTR printing, 119
 - closing driver, 120
 - communications, 120
 - opening driver, 120
 - removing driver, 119
 - writing to driver, 120
- E**
- EAP-FAST
 - 802.11 radio module
 - network EAP, 210
 - WPA encryption, 210
 - profile security information, 209
 - WEP encryption, 209

- Editing a profile, 196
- EnableSuppLogging(), 180
- EnableWep(), 171
- EnableZeroConfig(), 177
- EncryptionStatus(), 172
- Ending a connection, 108
- Environmental specifications, 16
- Epson Escape Sequences, 114
- ERROR_INSUFFICIENT_BUFFER
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
- ERROR_INVALID_PARAMETER
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
- EXITME.BIN, 148
- Expansion slot specifications, 16
- F**
- Favorite links
 - Pocket Internet Explorer, 71
- File Explorer
 - adding programs to Start menu, 34
- File Transfer Protocol See FTP
- file_list_section
 - CopyFiles
 - destination_filename, 131
 - flags, 131
 - source_filename, 131
 - DestinationDirs, 130
- filename
 - SourceDiskFiles, 130
- Find feature
 - Windows Mobile, 31
- fingerprints, cleaning, 224
- Flash File Store
 - packaging an application, 76
- Flash file system, utilities applet, 191
- Folder behavior connected to email server
 - ActiveSync, 56
 - IMAP4, 57
 - POP3, 57
 - SMS, 56
- FRAME_NOT_ACKED, 117
- FTP
 - client, 145
 - configurable parameters, 143
 - BlockSize, 143
 - DeviceName, 143
 - DeviceURL, 143
 - IDNATarget, 144
 - ManifestName, 144
 - PauseAtStartup, 144
 - Root, 144
 - FTPD CMDs subdirectory, 148
 - heartbeat, 145
 - RTC 959, 147
 - server, 145
 - installing applications, 78
 - server requests
 - CDUP, 145
 - CWD, 145
 - DELE, 145
 - HELP, 145
 - LIST, 145
 - MKD, 145
 - MODE, 145
 - NLST, 145
 - NOOP, 145
 - PASS, 145
 - PWD, 145
 - QUIT, 145
 - RETR, 145
 - RMD, 145
 - RNFR, 145
 - RNTO, 145
 - SITE, 146
 - SITE ATTRIB, 146
 - SITE BOOT, 146
 - SITE COPY, 146
 - SITE EKEY, 147
 - SITE EVAL, 147
 - SITE EXIT, 146
 - SITE GVAL, 147
 - SITE HELP, 146
 - SITE KILL, 146
 - SITE LOG, 147
 - SITE PLIST, 147
 - SITE PVAL, 147
 - SITE RUN, 147
 - SITE STATUS, 147
 - SITE TIMEOUT, 147
 - STOR, 145
 - SYST, 145
 - TYPE, 145
 - USER, 145
 - XCUP, 145
 - XCWD, 145
 - XMKD, 145
 - XPWD, 145
 - XRMD, 145
 - stopping server from application, 148
 - support, 145
 - web browsers, 147
- FTPDCE.EXE, 145, 148
 - AutoFTP, 149
 - FTP Server, 141
- FTPDCE.TXT, 148

FTPDCMDS subdirectory
FTP support, 148

G

GDI approach, 114
GetAssociationStatus(), 163
GetAuthenticationMode(), 164
GetBSSID(), 164
GetCCXStatus(), 170
GetCurrentDriverName(), 179
GetDiversity(), 165
GetLinkSpeed(), 165
GetMac(), 166
GetNetworkMode(), 166
GetNetworkType(), 167
GetPowerMode(), 168
GetRadioIpAddress(), 170
GetRSSI(), 168
GetSSID(), 167
Getting connected
ISP, 101
to an ISP, 101
creating a modem connection, 101
Windows Mobile, 101
GetTXPower(), 169
GetWepStatus(), 169
Good Read light
troubleshooting, 222

H

HAL
version of Pocket PC
IOCTL_HAL_GET_BOOTLOADER_VE
RINFO, 155
IOCTL_HAL_GET_OAL_VERINFO, 155
Headset jack, external, 3
Helper functions, 176
Highlighting text
Microsoft Reader, 70
host computer not receiving data, troubleshooting,
221

I

I/O controls
NPCP driver, 116
ID field values
IOCTL_HAL_ITC_READ_PARM
ITC_NVPARM_80211_INSTALLED, 152
ITC_NVPARM_80211_RADIOTYPE,
152
ITC_NVPARM_BLUETOOTH_INSTAL
LED, 153
ITC_NVPARM_CONTRAST, 152
ITC_NVPARM_DISPLAY_TYPE, 151

ITC_NVPARM_ECN, 152
ITC_NVPARM_INTERMEC_DATACOL
LECTION_HW, 152
ITC_NVPARM_INTERMEC_DATACOL
LECTION_SW, 152
ITC_NVPARM_INTERMEC_SOFTWARE
CONTENT, 152
ITC_NVPARM_MANF_DATE, 151
ITC_NVPARM_MCODE, 152
ITC_NVPARM_SERIAL_NUM, 151
ITC_NVPARM_SERIAL2_INSTALLED,
153
ITC_NVPARM_SERVICE_DATE, 151
ITC_NVPARM_SIM_PROTECT_HW_I
NSTALLED, 153
ITC_NVPARM_SIM_PROTECT_SW_IN
STALLED, 153
ITC_NVPARM_VERSION_NUMBER,
152
ITC_NVPARM_WAN_RI, 152
IOCTL_HAL_ITC_WRITE_SYSPARM
ITC_WAKEUP_MASK, 154
ITC_REGISTRY_SAVE_ENABLE, 154
IDNA
DeviceName, 143
DeviceURL, 143
IDNATarget, 144
ManifestName, 144
IDNATarget
FTP Server, 144
IMAP4
Folder behavior connected to email server, 57
Inbox
accounts, 58
connecting to a mail server, 57
downloading messages from server, 58
getting connected, 101
managing email messages and folders, 56
Start menu icon, 22
synchronizing email messages, 56
using My Text, 31
INF files
creating, 126
Input panel
letter recognizer, 26
Pocket Word, 61
selecting typed text, 26
Windows Mobile, 22
Installation functions
SETUP.DLL, 136
InstallDir
CEStrings, 127

Index

- Installing applications
 - using a storage card, 78
 - using Secure Digital cards, 78
 - with ActiveSync, 77
 - with FTP Server, 78
 - Installing drivers
 - DTR, 119
 - NPCP, 115
 - Instant messaging, 65
 - Integrated scanners See Internal scanners
 - Intermec Device Network Announcement See IDNA
 - Intermec part numbers, 15
 - Intermec settings, 191
 - beeper volume, 9, 109
 - Intermec settings applet
 - smartsystems, 9, 109
 - INTERMEC.MIB, 110
 - INTERMEC_PACKET_DRIVER
 - SwitchPacketDriver(), 180
 - Internal scanners
 - configuring, 123
 - specifications, 16
 - supported symbolologies, 124
 - Internet explorer
 - software build version, 14
 - Internet Service Provider See ISP, 101
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157, 182
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERINFO, 155
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159, 182
 - IOCTL_HAL_WARMBOOT, 156, 182
 - IOCTL_LOAD_NDIS_MINIPORT, 161
 - IOCTL_NPCP_BIND, 117
 - IOCTL_NPCP_CANCEL, 117
 - IOCTL_NPCP_CLOSE, 117
 - IOCTL_NPCP_ERROR, 117
 - IOCTL_NPCP_FLUSH, 117
 - IOCTL_PROCESSOR_INFORMATION, 159
 - IOCTL_UNLOAD_NDIS_MINIPORT, 161
 - IP address
 - replicating registry settings, 79
 - isDHCPEnabled(), 179
 - ISP
 - connecting to via Windows Mobile, 101
 - creating
 - a modem connection, 101
 - Pocket Internet Explorer, 71
 - Windows Mobile, 101
 - isSupplicantRunning(), 177
 - isZeroConfigEnabled(), 177
 - ITC_WAKEUP_MASK, 154
 - ITC_DEVID_80211RADIO_INTEL_2011B, 152
 - ITC_DEVID_80211RADIO_MAX values
 - ITC_DEVID_80211RADIO_INTEL_2011B, 152
 - ITC_DEVID_80211RADIO_NONE, 152
 - ITC_DEVID_80211RADIO_NONE, 152
 - ITC_DEVID_INTERMEC_EVIO, 152
 - ITC_DEVID_SCANHW_MAX values
 - ITC_DEVID_INTERMEC_EVIO, 152
 - ITC_DEVID_SCANHW_NONE, 152
 - ITC_DEVID_SCANHW_NONE, 152
 - ITC_IFTP_STOP, 148
 - ITC_KEYBOARD_CHANGE
 - CreateEvent(), 184
 - ITC_NVARM_80211_INSTALLED, 152
 - ITC_NVARM_80211_RADIOTYPE, 152
 - ITC_NVARM_BLUETOOTH_INSTALLED, 153
 - ITC_NVARM_CONTRAST, 152
 - ITC_NVARM_DISPLAY_TYPE, 151
 - ITC_NVARM_ECN, 152
 - ITC_NVARM_INTERMEC_DATACOLLECTION_HW, 152
 - ITC_NVARM_INTERMEC_DATACOLLECTION_SW, 152
 - ITC_NVARM_INTERMEC_SOFTWARE_CONTENT, 152
 - ITC_NVARM_MANF_DATE, 151
 - ITC_NVARM_MCODE, 152
 - ITC_NVARM_SERIAL_NUM, 151
 - ITC_NVARM_SERIAL2_INSTALLED, 153
 - ITC_NVARM_SERVICE_DATE, 151
 - ITC_NVARM_SIM_PROTECT_HW_INSTALLED, 153
 - ITC_NVARM_SIM_PROTECT_SW_INSTALLED, 153
 - ITC_NVARM_VERSION_NUMBER, 152
 - ITC_NVARM_WAN_RI, 152
 - ITC_REGISTRY_SAVE_ENABLE, 154
 - ITCADC.MIB, 110
 - ITCSNMP.MIB, 110
 - ITCTERMIAL.MIB, 110
- K**
- Keep Alive message, 221

- KernelIoControl
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157, 182
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERI
NFO, 155
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159, 182
 - IOCTL_HAL_WARMBOOT, 156, 182
 - IOCTL_PROCESSOR_INFORMATION,
159
- KernelIoControl(), 150
- Key clicks
 - bar code configuration, 214, 218
 - configuration parameter, 214
- Key sequences
 - alpha (green) keys, 10
 - orange keys, 9
- Keypad
 - advanced remapping, 185
 - alpha (green) key sequences, 10
 - alpha plane, 183
 - change notification, 184
 - driver registry settings, 184
 - orange key sequences, 9
 - orange plane, 183
 - planes, 183
 - registry settings
 - alpha plane, 184
 - orange plane, 184
 - unshifted plane, 184
 - remapping, 182
 - sample registry keys, 187
 - scan codes, 185
 - specifications, 16
- L**
- LEAP
 - 802.11 radio module
 - network EAP, 208
 - WPA encryption, 208
 - profile security information, 207
 - WEP encryption, 207
- LEAP security
 - fast roaming (CCKM), 212
- LED status, 10
- Letter recognizer
 - Windows Mobile input panel, 26
- Library
 - Microsoft Reader, 69
- Line printing, 114
- lpBytesReturned
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERI
NFO, 156
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_PROCESSOR_INFORMATION,
160
- lpInBuf
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERI
NFO, 156
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156
 - IOCTL_PROCESSOR_INFORMATION,
160
- lpInBufSize
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156

- lpOutBuf
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERINFO, 156
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156
 - IOCTL_PROCESSOR_INFORMATION, 160
- LPT9 printer device, 115
- M**
- Managing email messages and folders
 - via Inbox, 56
- ManifestName
 - FTP Server, 144
- Memory and storage
 - specifications, 17
- Menu applet
 - Windows Mobile settings, 32
- Messages
 - sending to
 - contacts, 49
 - via Inbox
 - downloading from server, 58
- MIBs
 - ASN.1, 109
 - files, 109
 - object identifier, 110
 - OIDs, 110
- Microphone, 2
- Microprocessor
 - specifications, 17
- Microsoft Developer Network Library See MSDN library
- Microsoft Exchange email account, 65
- Microsoft Passport account, 65
- Microsoft Reader
 - adding bookmarks, 70
 - adding drawings, 70
 - annotations index, 70
 - attaching notes, 70
 - books
 - reading, 70
 - removing, 71
 - features, 70
 - highlighting text, 70
 - searching for text, 70
 - using the library, 69
 - Windows Mobile, 69
- Microsoft security
 - allow fast roaming (CCKM), 212
 - detect rogue APs, 212
 - enable mixed cell, 212
- Migrating from another computer, 82
- Mixed cell
 - enable via Microsoft security, 212
- Mobile Favorites
 - Pocket Internet Explorer, 72
- Modems
 - creating a connection
 - to an ISP, 101
- MSDN library, 148
- MSDN Windows CE documentation, 100
- MSN Messenger
 - about, 65
 - accounts
 - Microsoft Exchange email, 65
 - Microsoft Passport, 65
 - contacts
 - managing, 67
 - sending messages, 68
 - working with, 67
 - using My Text, 31
- N**
- nDeviceId
 - NLEDGetDeviceInfo, 182
- NDIS_ENCRYPTION_1_ENABLED
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_1_KEY_ABSENT
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_2_ENABLED
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_2_KEY_ABSENT
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_3_ENABLED
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_3_KEY_ABSENT
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_ENCRYPTION_DISABLED
 - EncryptionStatus(), 172
 - GetWepStatus(), 169

- NDIS_ENCRYPTION_NOT_SUPPORTED
 - EncryptionStatus(), 172
 - GetWepStatus(), 169
- NDIS_MIXED_CELL_OFF
 - SetMixedCellMode(), 175
- NDIS_MIXED_CELL_ON
 - SetMixedCellMode(), 175
- NDIS_NET_AUTO_UNKNOWN
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_MODE_ESS
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_MODE_IBSS
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_MODE_UNKNOWN
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_TYPE_DS
 - GetNetworkType(), 167
- NDIS_NET_TYPE_FH
 - GetNetworkType(), 167
- NDIS_NET_TYPE_OFDM_2_4G
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_TYPE_OFDM_5G
 - GetNetworkMode(), 166
 - SetNetworkMode(), 174
- NDIS_NET_TYPE_UNDEFINED
 - GetNetworkType(), 167
- NDIS_NETWORK_EAP_MODE_OFF
 - GetCCXStatus(), 170
 - SetCCXStatus(), 175
- NDIS_NETWORK_EAP_MODE_ON
 - GetCCXStatus(), 170
 - SetCCXStatus(), 175
- NDIS_POWER_LEVEL_1
 - GetTXPower(), 169
- NDIS_POWER_LEVEL_15
 - GetTXPower(), 169
- NDIS_POWER_LEVEL_30
 - GetTXPower(), 169
- NDIS_POWER_LEVEL_5
 - GetTXPower(), 169
- NDIS_POWER_LEVEL_63
 - GetTXPower(), 169
- NDIS_POWER_LEVEL_UNKNOWN
 - GetTXPower(), 169
- NDIS_RADIO_ASSOCIATED
 - GetAssociationStatus(), 163
- NDIS_RADIO_AUTH_MODE_AUTO
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_ERROR
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_OPEN
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_SHARED
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_WPA
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_WPA_NONE
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_AUTH_MODE_WPA_PSK
 - GetAuthenticationMode(), 164
 - SetAuthenticationMode(), 173
- NDIS_RADIO_POWER_AUTO
 - GetPowerMode(), 168
 - SetPowerMode(), 174
- NDIS_RADIO_POWER_MODE_CAM
 - GetPowerMode(), 168
 - SetPowerMode(), 174
- NDIS_RADIO_POWER_MODE_FAST_PSP
 - GetPowerMode(), 168
 - SetPowerMode(), 174
- NDIS_RADIO_POWER_MODE_PSP
 - GetPowerMode(), 168
 - SetPowerMode(), 174
- NDIS_RADIO_POWER_UNKNOWN
 - GetPowerMode(), 168
 - SetPowerMode(), 174
- NDIS_RADIO_SCANNING
 - GetAssociationStatus(), 163
- NDIS_SUPP_LOGGING_OFF
 - EnableSuppLogging(), 180
- NDIS_SUPP_LOGGING_ON
 - EnableSuppLogging(), 180
- NDISUIO_PACKET_DRIVER
 - SwitchPacketDriver(), 180
- NETWLAN.DLL, 162
- Network adapters
 - no networking, 94
 - wireless 802.11, 94
 - wireless printing, 86
- Network Connection icon, 221, 222

- Network EAP
 - EAP-FAST security method, 210
 - LEAP security method, 208
 - PEAP security method, 200
 - TLS security method, 202
 - TTLS security method, 205
- Network type
 - 802.11 radio module, 196
- nInBufSize
 - IOCTL_HAL_GET_BOOTLOADER_VERIFY, 156
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_PROCESSOR_INFORMATION, 160
- nInfoId
 - NLEDGetDeviceInfo, 181
- NLED driver
 - vibrator, 181
- NLED.H, 181, 182
 - NLEDGetDeviceInfo, 181
 - NLEDSetDevice, 182
- NLED_COUNT_INFO
 - NLEDGetDeviceInfo, 181
- NLED_SETTINGS_INFO_ID
 - NLEDGetDeviceInfo, 181
 - NLEDSetDevice, 182
- NLED_SUPPORTS_INFO_ID
 - NLEDGetDeviceInfo, 181
- NLEDGetDeviceInfo, 181
- NLEDSetDevice, 182
- Notes
 - drawing on the screen, 29
 - creating a drawing, 29
 - recording a message, 30
 - Start menu icon, 22
 - synchronizing notes, 54
 - writing on the screen, 27
 - alternate writing, 28
 - converting writing to text, 28
 - tips for good recognition, 28
- nOutBufSize
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERIFY, 156
 - IOCTL_HAL_GET_DEVICE_INFO, 150
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156
- IOCTL_HAL_ITC_WRITE_SYSPARM, 153
- IOCTL_HAL_REBOOT, 159
- IOCTL_HAL_WARMBOOT, 156
- IOCTL_PROCESSOR_INFORMATION, 160
- NPCP printing, 114, 115
 - closing driver, 116
 - COM1 parameters, 115
 - communications, 117
 - driver I/O controls, 116
 - installation, 115
 - LPT9, 115
 - opening driver, 115
 - reading from driver, 116
 - removal, 115
 - sample code, 118
 - writing to driver, 116
- NPCPPORT.DLL, 114
- O**
- Object Store
 - packaging an application, 76
- Object store
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156
- OEMIOCTL.H
 - IOCTL_GET_CPU_ID, 160
 - IOCTL_HAL_COLDBOOT, 157
 - IOCTL_HAL_GET_BOOT_DEVICE, 158
 - IOCTL_HAL_GET_BOOTLOADER_VERIFY, 155
 - IOCTL_HAL_GET_OAL_VERINFO, 155
 - IOCTL_HAL_GET_RESET_INFO, 157
 - IOCTL_HAL_ITC_READ_PARM, 151
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156
- Oldstyle device ID, 154
- ONeil printing
 - installing driver, 119
- ONEIL.DLL, 119
- Opening drivers
 - DTR, 120
 - NPCP, 115
- Operating system
 - specifications, 17
- OSVERSIONINFO.dwBuildNumber, 128
- OSVERSIONINFO.dwVersionMajor, 128
- OSVERSIONINFO.dwVersionMinor, 128
- Owner Information applet
 - Windows Mobile settings, 32

P

- Packaging an application
 - Flash File Store, 76
 - Object Store, 76
 - Persistent Storage Manager, 76
 - Secure Digital storage cards, 76
- Page format printing, 114
- Password applet
 - Windows Mobile settings, 32
- Patent information, v
- PauseAtStartup
 - FTP Server, 144
- PB20 printers
 - DTR driver, 119
 - printer support, 113
- PB40 printers
 - DTR driver, 119
 - printer support, 113
- PB42 printers
 - DTR driver, 119
 - printer support, 113
- PEAP
 - 802.11 radio module
 - network EAP, 200
 - WPA encryption, 199
 - profile security information, 198
 - WEP encryption, 199
- Phone application
 - CDMA radios
 - adding contact to speed dial, 51
- Physical dimensions
 - specifications, 17
- pInput
 - NLEDSetDevice, 182
- PKFUNCS.H
 - IOCTL_HAL_GET_DEVICEID, 154
 - IOCTL_PROCESSOR_INFORMATION, 159
- Planes, keypad, 183
- Pocket Excel
 - about, 63
 - creating a workbook, 64
- Pocket Internet Explorer
 - about, 71
 - AvantGo channels, 73
 - browsing the Internet, 74
 - favorite links, 71
 - getting connected, 101
 - mobile favorites, 72
 - Mobile Favorites folder, 71
 - software build, 14
 - Start menu icon, 22
 - viewing mobile favorites and channels, 74

- Pocket Outlook, 36
- Pocket PC
 - IOCTL_HAL_GET_BOOTLOADER_VERINFO, 155
 - IOCTL_HAL_GET_OAL_VERINFO, 155
- Pocket Word
 - about, 60
 - creating a document, 60
 - drawing mode, 63
 - synchronizing, 63
 - typing mode, 61
 - writing mode, 62
- POP3
 - Folder behavior connected to email server, 57
- pOutput
 - NLEDGetDeviceInfo, 181
- Power applet
 - battery status, 3
 - RAM maintenance, 4
 - Windows Mobile settings, 32
- Printer support, 114
 - NPCP printer driver, 114
 - ONeil printer driver, 119
- PRISMNDS.DLL, 162
- problems, finding and solving, 223
- Processor information
 - IOCTL_PROCESSOR_INFORMATION, 159
- ProcessorType
 - CEDevice, 127
- Profile label
 - 802.11 radio module, 196
- Profiles
 - 802.11 radio module, 196
 - advanced settings, 212
 - basic information, 196
 - security information, 197
 - editing, 196
- Programs, adding or removing
 - Windows Mobile, 32
- Provider
 - Version, 126
- PSM
 - determining build version, 12
 - packaging an application, 76

Q

- Query Information functions, 163

R

- RadioConnect(), 162
- RadioDisassociate(), 163
- RadioDisconnect(), 163

- Radios
 - card support, 15
- Reader commands, 214
 - configuration change, 214
 - date and time settings, 215
- ReadFile()
 - NPCP printing, 114
- Reading from drivers
 - NPCP, 116
- Reboot methods
 - IOCTL_HAL_COLDBOOT, 182
 - IOCTL_HAL_REBOOT, 182
 - IOCTL_HAL_WARMBOOT, 182
- REBOOTME.BIN, 148
- Recharging time, 17
- Record button
 - recording a message, 30
- Recording
 - via Notes, 30
 - via Pocket Word, 62
- Recovery CD
 - AutoCab method, 83
 - RegFlushKey() API, 141
 - updating the system software, 82
- Recurrence pattern
 - Calendar, 38
- REGFLUSH.CAB, 79
- RegFlushKey(), 141
- RegisterDevice(), 115
 - DTR printing, 119
- Registry
 - confirm the new registry file, 80
 - delete the old registry save, 79, 80
 - FTP Server parameters, 143
 - keypad remapping, 184
 - load the application, 80
 - replicating settings, 79
 - sample view of key mapping, 187
 - save location
 - IOCTL_HAL_ITC_WRITE_SYSPARM, 153
 - update other computers, 81
- Registry Save applet, 191
- Registry settings
 - AutoCfg, 100
 - AutoFTP, 149
 - AutoInterval, 100
 - AutoIP/DHCP, 100
 - DhcpMaxRetry, 100
 - DhcpRetryDialogue, 100
 - keypad driver, 184
 - keypad planes
 - alpha, 184
 - orange, 184
 - unshifted, 184
- RegOpenKeyEx(), 183
- RegQueryValueEx(), 183
- RegSetValueEx(), 183
- Regulatory approvals
 - specifications, 17
- RemoveWep(), 176
- Removing drivers
 - DTR, 119
 - NPCP, 115
- Removing programs
 - Windows Mobile, 32
- RenewDHCP(), 179
- Replicating registry settings, 79
- Reset button, 13
- ResetRadioToSystemSave(), 180
- RFC 959, 147
- Roaming
 - Microsoft security, 212
- Root
 - FTP Server, 144
- RPM.EXE, 130
- RPMCE212.INI, 130
- S**
- Sample code
 - NPCP printing, 118
- Scan codes, 185
- Scanner
 - beeper volume
 - turning it off, 8
 - turning it on, 8
 - specifications, 16
 - unit configuration parameters
 - automatic shutoff, 213
 - backlight timeout, 213
 - date/time, 213
 - key clicks, 214
 - volume, 213
 - utilities configuration
 - button wakeup mask, 192
- scanner
 - cleaning the window, 224
 - window, illustrated, 224
- screen
 - cleaning, 224
- Secure Digital cards
 - installing applications, 78
 - packaging an application, 76
 - specifications, 16
- Set information functions, 171
- SetAuthenticationMode(), 173
- SetCCXStatus(), 175

- SetChannel(), 173
- SetMixedCellMode(), 175
- SetNetworkMode(), 174
- SetPowerMode(), 174
- SetSSID(), 175
- Setting date and time, 215
- SETUP.DLL, 129, 136
 - DllMain, 136
 - installation functions, 136
- SHFullScreen(), 141
- shortcut_list_section
 - CEShortcuts
 - shortcut_filename, 133
 - shortcut_type_flag, 133
 - target_file_path, 133
- Signature
 - Version, 126
- SIM cards
 - protection hardware, 153
 - protection software, 153
 - software installed, 153
- Simple Network Management Protocol See SNMP
- SmartSystems, 9, 109
- SMS
 - Folder behavior connected to email server, 56
- SNMP, 109
- Software versions, 14
- SourceDiskFiles
 - filename, 130
- SourceDiskNames
 - CESignature, 129
 - disk_ordinal, 129
- SourceDisksNames.MIPS, 130
- SourceDisksNames.SH3, 130
- Speaker, 2
- Specifications, 16
 - battery, 17
 - display, 16
 - environmental, 16
 - expansion slots, 16
 - integrated scanner options, 16
 - integrated wireless, 16
 - keypad options, 16
 - memory and storage, 17
 - microprocessor, 17
 - operating system, 17
 - physical dimensions, 17
 - regulatory approvals, 17
 - standard communications, 17
- SSID (network name)
 - 802.11 radio module, 196
- Standard communications
 - specifications, 17
- Start Menu
 - adding programs, 34
 - via File Explorer, 34
- StartScanList(), 178
- StartSupplicant(), 178
- Static IP
 - replicating registry settings, 79
- StopSupplicant(), 178
- Storage media, 15
 - specifications, 16
- Stream device driver
 - NPCPPORT.DLL, 114
 - ONEIL.DLL, 119
- string_key
 - Strings, 127
- Strings
 - string_key, 127
- SwitchPacketDriver(), 180
- Symbologies
 - internal scanner supported symbologies, 124
 - scanning labels, 215
- symbologies
 - disabled incorrectly, 223
- Synchronizing
 - AvantGo channels, 73
 - Calendar, 37
 - contacts, 47
 - email messages, 56
 - favorite links, 71
 - mobile favorites, 72
 - notes, 54
 - Pocket Word, 63
- SYSTEMINFO.dwProcessorType, 127
- T**
- TAHOMA.TTF, 130
- Tasks
 - Start menu icon, 22
- TCP/IP, 93
- TCP/IP client
 - DHCP server, 100
- technical support
 - sending CN2 for repair, 223
- Text messages
 - Windows Mobile, 31
- Time, setting, 215

Index

- TLS
 - 802.11 profile
 - certificates, 207
 - WPA encryption, 202
 - 802.11 radio module
 - network EAP, 202
 - profile security information
 - WEP encryption, 201
 - WPA encryption, 202
- Today applet
 - Windows Mobile settings, 32
- Today screen
 - Windows Mobile, 21
- Tools CD
 - CAB files, 77
 - MIB files, 110
 - sample NPCP code, 118
- touch screen
 - cleaning, 224
 - illustrated, 224
- Troubleshooting
 - CAB Wizard, 140
- troubleshooting
 - bar code symbologies, 223
 - lost network connection after suspend, 221
 - Network Connection icon, 221
- TTLS
 - 802.11 radio module
 - network EAP, 205
 - WPA encryption, 204
 - profile security information
 - WEP encryption, 203, 204
- Typing mode
 - Pocket Word, 61
- U**
- UDP
 - FTPDCE, 145
- UDP broadcasts
 - IDNATarget parameter, 144
- UDP Plus, 93
- Unit
 - configuration parameters
 - automatic shutoff, 213
 - backlight timeout, 213
 - date/time, 213
 - key clicks, 214
 - volume, 213
- Unit Manager
 - date/time, 213
- Unshifted plane on keypad, 183
- UnsupportedPlatforms
 - CEDevice, 127
- Updating
 - bootloader, 77
- URLs
 - ActiveSync, 35
 - full screen display, 141
 - MIBs, 110
 - Microsoft Exchange email account, 65
 - Microsoft Passport account, 65
 - Microsoft support, 20
 - MSDN library, 148
 - MSDN Windows CE documentation, 100
 - Windows Mobile, 20
 - Windows Mobile support, 20
- URODDSVCS.EXE, 162
- USB communications, 92
- Utilities applet
 - app launch, 193
 - registry save, 191
 - wakeup mask, 192
- UUID, 154
- V**
- Version
 - CESignature, 126
 - Provider, 126
 - Signature, 126
- VersionMax
 - CEDevice, 128
- VersionMin
 - CEDevice, 128
- Vibrator
 - programming, 181
- Video files
 - Windows Media Player, 68
- Viewing mobile favorites and channels
 - Pocket Internet Explorer, 74
- Virtual wedge
 - bar code configuration
 - grid, 218
 - postamble, 218
 - preamble, 218
- VN_CLASS_ASIC, 152
- VN_CLASS_BOOTSTRAP, 152
- VN_CLASS_KBD, 152
- Volume
 - bar code configuration, 213, 216
 - configuration parameter, 213
- W**
- Wakeup mask applet, 192
- WAP pages, 71
 - connecting to an ISP, 101
- Warm boot
 - IOCTL_HAL_REBOOT, 159
 - IOCTL_HAL_WARMBOOT, 156

- Warm boot, performing, 13
 - WAV files, 62
 - WCESTART.INI, 130
 - Web browsers
 - FTP support, 147
 - Web pages, 71
 - connecting to an ISP, 101
 - WEP encryption
 - EAP-FAST security method, 209
 - LEAP security method, 207
 - PEAP security method, 199
 - profile security information, 197, 198
 - TLS security method, 201
 - TTLS security method, 203, 204
 - zero configuration, 97
 - Windows CE documentation (MSDN), 100
 - Windows Media Player
 - Start menu icon, 22
 - Windows Mobile, 68
 - Windows Mobile
 - ActiveSync, 35
 - basic skills, 21
 - command bar, 22
 - getting connected, 101
 - MSN Messenger, 65
 - navigation bar, 22
 - notifications, 23
 - Pocket Excel, 63
 - Pocket Word, 60
 - popup menus, 23
 - programs, 21
 - support URLs, 20
 - Today screen, 21
 - where to find information, 20
 - Windows Media Player, 68
 - writing on the screen, 27
 - Wireless network, 94, 195
 - specifications, 16
 - Wireless printing
 - Bluetooth compatible module, 86
 - Wireless TCP/IP installations
 - BlockSize parameter, 143
 - Workbook
 - creating via Pocket Excel, 64
 - WPA authentication
 - with pre-shared key
 - Zero Configuration, 98
 - WPA encryption
 - EAP-FAST security method, 210
 - LEAP security method, 208
 - PEAP security method, 199
 - TLS security method, 202
 - TTLS security method, 204
 - WPport, 87
 - WriteFile()
 - DTR printing, 119, 120
 - IrDA printing, 114
 - NPCP printing, 114, 116
 - Writing mode
 - Pocket Word, 62
 - Writing on the screen
 - Pocket Word, 62
 - Writing on the screen See Notes, 27
 - Writing to drivers
 - DTR, 120
 - NPCP, 116
- x**
- Xscale processor ID
 - IOCTL_GET_CPU_ID, 160
- z**
- Zero Configuration
 - enabling, 196
 - enabling WPA authentication, 98



Corporate Headquarters
6001 36th Avenue West
Everett, Washington 98203
U.S.A.

tel 425.348.2600

fax 425.355.9551

www.intermec.com

CN2B Mobile Computer User's Manual



P/N 935-001-001