Pocket PC FAQ

Undocumented ActiveSync Keys

Terence Goggin, Information Appliance Associates

At last June's CE DevCon, Microsoft had two-dozen "Sync Stations" set up at various locations. A Sync Station consisted of a PC running CEServices/ ActiveSync that allowed all of the conference attendees to download the latest conference information directly to their CE devices.

There was, however, one very striking difference between these Sync Stations and the PC you're currently using to synchronize your CE device: the Sync Stations never prompted us to create a device-to-desktop partnership! In other words, every device that connected did so as a Guest without user intervention.

About the Author

Terence "Dr. CE" Goggin, author of "The Windows CE Developer's Handbook", a recognized Windows CE expert, has written numerous books and articles on Windows programming and the Internet. He is a San Diego, California-based consultant specializing in Windows CE solutions. He can be reached at terenceg@doctorce.com

How is this possible? Clearly, you've seen that every time you connect a new device –even if you want to connect in Guest mode– you're always prompted. Well, it turns out that what made this magic possible is an undocumented, unsupported ActiveSync 3.0 registry setting.

Even more interesting, it turns out that there are lots of hidden and undocumented registry settings for ActiveSync 3.x! In this article, we'll look at how to turn any PC into our very own Sync Station. Then, we'll look at a case study showing real-world applications for these settings.

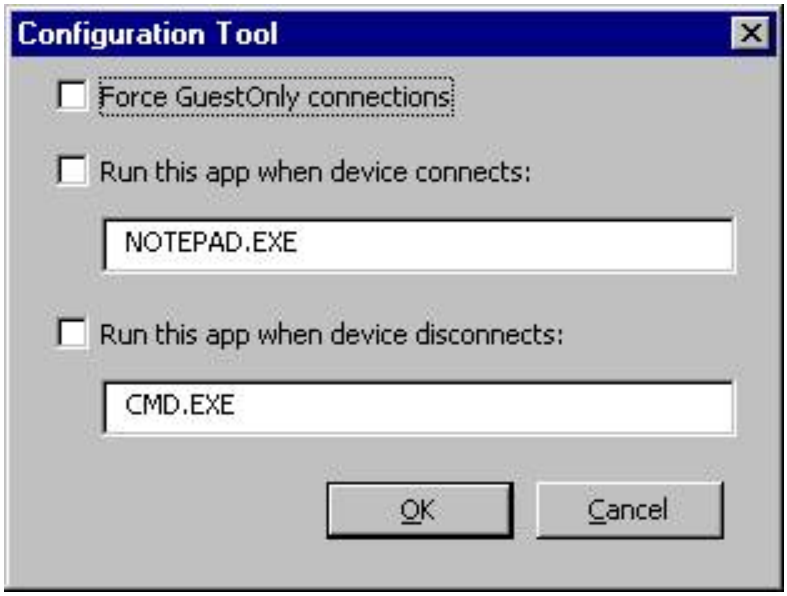
Build your own Sync Station

The good news is that it's possible to create your very own Sync Station using just three of these undocumented registry keys…

1. `GuestOnly`
2. `AutoStartOnConnect`
3. `AutoStartOnDisconnect`

Let's illustrate how each of these work and how to use them by looking at some C source code for a sample application (available for download from the URL specified at the end of this article). The app itself is rather simple, consisting of a dialog with several check boxes and two edit controls, as shown in Figure 1.

Figure 1: Registry Configuration Tool



GuestOnly

`GuestOnly` is the real magic in our personal Sync Stations; it's responsible for allowing all new devices (i.e., those that don't already have a partnership with the PC) to connect as guests without being prompted to establish a partnership. (Note that all devices that have an existing partnership with the PC will still connect and synchronize as they did before.)

`GuestOnly` is a `DWORD` value that resides at `HKLM\Software\Microsoft\Windows CE Services`. Of course, if you use regedit to look at the other values under that key, you'll notice that `GuestOnly` is missing. In order to test it, then, you must manually create the value and then set its value to 1. Or, in code…

```
        dwEnabled = (DWORD)0x01;
        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,              &dwDisp);
        RegSetValueEx(hReg, TEXT("GuestOnly"), 0, REG_DWORD,
            (const unsigned char *)&dwEnabled, sizeof(DWORD));
```

AutoStartOnConnect

`AutoStartOnConnect` is the next piece in our personal Sync Stations; it is a key that also resides under `HKLM\Software\Microsoft\Windows CE Services`. When a device connects to the system, ActiveSync enumerates all string values under the `AutoStartOnConnect` key and attempts to launch the specified applications.

So, in order to launch an app on a device's connection, we must create a name/string value pair (any unique name will be fine) under the key, and then set the value to the path of our custom sync application. Or, in code, we'd first create/open the `HKLM\Software\Microsoft\Windows CE Services\AutoStartOnConnect` key, retrieve the app path from an edit box, and then write that value to the registry…

```
        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services\
\AutoStartOnConnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);
        memset(szExename, 0, STRLEN);
        if (GetDlgItemText(hwnd, IDC_EDT_CONNECTEXE, szExename, STRLEN) !=0)
        {
           RegSetValueEx(hReg, TEXT("TestApp"), 0, REG_SZ,
              (BYTE* const)&szExename, strlen(szExename)+1);
        }
```

Note that although we're calling `RegCreateKeyEx()`, the `AutoStartOnConnect` key should already exist in your registry. Also, although we're using the value name of "`TestApp`", any unique name would work just fine as all of the values are enumerated regardless of how they're named.

AutoStartOnDisconnect

`AutoStartOnDisconnect` is also a key that resides under `HKLM\Software\Microsoft\Windows CE Services`. Just as with `AutoStartOnConnect`, when a device disconnects from the system, ActiveSync enumerates all string values under the `AutoStartOnDisconnect` key and attempts to launch the specified applications.

So, just as we did above, we must create a name/string value pair under the key, and then set the value. Or, in code, we'd first create/open the `HKLM\Software\Microsoft\Windows CE Services\AutoStartOnDisconnect` key, retrieve the app path from an edit box, and then write that value to the registry…

```
        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE
Services\\AutoStartOnDisconnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);
        memset(szExename, 0, STRLEN);
        if (GetDlgItemText(hwnd, IDC_EDT_DISCONNECTEXE, szExename, STRLEN) !
=0)
        {
           RegSetValueEx(hReg, TEXT("TestApp"), 0, REG_SZ ,
              (BYTE* const)&szExename, strlen(szExename)+1);
        }
```

The same notes that applied above apply here as well; the `AutoStartOnDisconnect` key should already exist in your registry and you're free to use any unique name (instead of "`TestApp`").

When we put all of these code blocks together, we get the "OK" button handler for our test application, as shown in Listing 1.

Case Study: Med-i-nets.com

To further illustrate these keys and why you might want your own sync station, let's take a look at how med-i-nets.com relies on this very set up. Med-i-nets.com offers a complete online prescription ordering system for medical professionals, with a web-based backend and a Windows CE-based client.

Each medical professional who subscribes to the service has a Windows CE device, which is used to display the list of today's patients, his or her prescription history, new prescriptions, and so on. At the beginning of the work day, the user synchronizes their device via the med-i-nets.com website and a clever browser plugin which does the

But wait! There's more!

So how many of these undocumented ActiveSync registry goodies are there?

Well, of course, no one knows – but there are definitely more than the three we've looked at here.

To find more of these hidden and undocumented keys, you might want to open the main CE services executables (CeAppMgr.exe, Wcescomm.exe, and WCESMgr.exe) in an editor such as notepad and browse around for a bit. Of course, this will require more experimentation because the full registry key is often omitted and you'll likely only find the actual value's

work of removing old patients and adding new ones.

name. Given the power that some of these keys offer, though, the payoff can definitely be worth the investment.

Everything was proceeded according to the spec, when we realized that the typical doctor's office/practice scenario could involve one PC for the entire office, with multiple doctors, nurses, etc. using that PC.

Now, for many of us who use CE devices in our regular work, this would not be an issue. We'd simply create a partnership for each device or instruct the user that when prompted, they can simply choose to connect in Guest mode. However, having to make the Guest versus Partner prompt and then creating multiple partnerships –or even one for that matter– might seem like too much work to a doctor who is already quite busy. The ideal situation, it was determined, would be to set up the PC in such a way that the user would never be prompted at all. And that's where the custom Sync Station settings come in.

The way the setup works now is that the user is given some custom software to install on their office PC, which of course sets these registry keys for them. Now, when they cradle their device, not only are they connected in Guest mode, but also a separate desktop application (listed under the `AutoStartOnConnect` key) prompts them to go to the med-i-net.com website and synchronize their patients and prescriptions. (Naturally, this application will also launch their browser and take them to the site.) And, when the user is done synchronizing and disconnects their CE device, a second application (listed under the `AutoStartOnDisconnect` key) checks to see whether the browser is still open and offers to close it for the user.

Clearly, the custom Sync Station scenario has real-world applications and is not limited to daily conference updates.

Hopefully, the code and the case study here will inspire even more uses of these registry keys. But please keep in mind that Microsoft is very clear that these keys are unsupported and undocumented. Clearly, though, they're too powerful to ignore.

Download ActiveSyncKeys

Acknowledgements

The author would like to thank James Miller of Microsoft and Chris De Herrera of CEWindows.NET for their assistance in preparing this article.

Listing 1: "OK" button handler for sample application

```
case IDOK:

{
     //write settings to registry!
     //first set GuestOnly value.

     hChk = GetDlgItem(hwnd, IDC_CHK_GUESTENABLE);
     if (BST_CHECKED == SendMessage(hChk, BM_GETSTATE, 0, 0))
     {
        //if GuestOnly check box is checked,
        //then create the value and set its value to 1.
        dwEnabled = (DWORD)0x01;

        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);

        RegSetValueEx(hReg, TEXT("GuestOnly"), 0, REG_DWORD,
            (const unsigned char *)&dwEnabled, sizeof(DWORD));
     }
     else
     {
        //otherwise, b/c this is an unsupported value,
        //delete it instead of simply setting it to 0

        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);
        RegDeleteValue(hReg, TEXT("GuestOnly"));

     }

     //now set AutoStartOnConnect key/value
     hChk = GetDlgItem(hwnd, IDC_CHK_CONNECT);

     if (BST_CHECKED == SendMessage(hChk, BM_GETSTATE, 0, 0))
     {
        //if the AutoStartOnConnect check box is checked,
        //then write the app name to the registry

        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services\
\AutoStartOnConnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);

        memset(szExename, 0, STRLEN);
        if (GetDlgItemText(hwnd, IDC_EDT_CONNECTEXE, szExename, STRLEN) !=0)
        {
           RegSetValueEx(hReg, TEXT("TestApp"), 0, REG_SZ,
              (BYTE* const)&szExename, strlen(szExename)+1);
        }
     }
     else
```

```
     {
        //otherwise, delete the value.
        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services\
\AutoStartOnConnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);

        RegDeleteValue(hReg, TEXT("TestApp"));
     }

     //now set AutoStartOnDisconnect key/value
     hChk = GetDlgItem(hwnd, IDC_CHK_DISCONNECT);
     if (BST_CHECKED == SendMessage(hChk, BM_GETSTATE, 0, 0))
     {
        //if the AutoStartOnDisconnect check box is checked,
        //then write the app name to the registry

        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services\
\AutoStartOnDisconnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);
        memset(szExename, 0, STRLEN);
        if (GetDlgItemText(hwnd, IDC_EDT_DISCONNECTEXE, szExename, STRLEN) !
=0)
        {
           RegSetValueEx(hReg, TEXT("TestApp"), 0, REG_SZ ,
              (BYTE* const)&szExename, strlen(szExename)+1);
        }
     }
     else
     {
        //otherwise, delete the value.
        RegCreateKeyEx(HKEY_LOCAL_MACHINE,
            TEXT("Software\\Microsoft\\Windows CE Services\
\AutoStartOnDisconnect"),
            0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, (PHKEY)
&hReg,
            &dwDisp);
        RegDeleteValue(hReg, TEXT("TestApp"));
     }

     //close the reg key
     RegCloseKey(hReg);
     EndDialog(hwnd, 0);
     return TRUE;

}
```