From: Aric Weiker
Sent: Thursday, August 29, 2002 2:13 PM
To: Charlie DeJong
Subject: FW: August 2002 Windows SDK Update: Revised Draft: ONE MORE SMALL ADDITION

Importance: High


SettlementAPI Article r4jo.htm...

Charlie,

FYI - Here's the revised O'Reilly article. They implemented all of our requested edits, so this should meet our expectations when this is published later today. PR and LCA have both seen this.

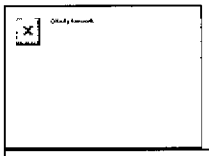
-Aric

-----Original Message-----
From: John Osborn [mailto:josborn@oreilly.com]
Sent: Thursday, August 29, 2002 1:41 PM
To: John Osborn; Aric Weiker; Sara Williams
Cc: Sanglin@oreilly.com; ron@oreilly.com
Subject: RE: August 2002 Windows SDK Update: Revised Draft: ONE MORE SMALL ADDITION
Importance: High

One additionmal change: I added a link to your Settlement Program Site at MSDN following the first paragraph.

John

> -----Original Message-----
> From: John Osborn [mailto:josborn@oreilly.com]
> Sent: Thursday, August 29, 2002 4:30 PM
> To: Aric Weiker; Sara Williams
> Cc: Sanglin@oreilly.com; ron@oreilly.com
> Subject: August 2002 Windows SDK Update: Revised Draft
> Importance: High
>
>
> Aric and Sara:
>
> Here's a revised draft. Give me a call as soon as you're clear to
> review remaining sticking points.
>
> Thanks and regards.
>
> John
>
> Executive Editor, Microsoft .NET and Windows
> O'Reilly and Associates
> 90 Sherman Street, Cambridge, MA 02140
> Phone: 617-499-7588 FAX: 617-661-1116



Published on **The O'Reilly Network** (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/dotnet/2002/08/28/winapi.html
See this if you're having trouble printing code examples

The Once Undocumented Windows APIs

by Chris Sells and Curt Hagenlocher
08/28/2002

There is a common belief among programmers -- particularly those disinclined to believe the truth of anything coming out of Redmond -- that Windows is full of "secret functions." These functions, so the story goes, are used by Microsoft to prevent independent reimplementation of its operating system and to give its own applications programmers an extra advantage With the August 2002 release of its Platform SDK, Microsoft is complying with the terms of the consent decree it signed with the Department of Justice in November 2001 by documenting those interfaces previously regarded as internal that are called by "Microsoft Middleware" (as defined in the proposed consent decree) to obtain services from the Microsoft® Windows® 2000 and Windows XP desktop client operating systems. If you are one of the people who harbor suspicions against Microsoft, then you're not likely to be convinced that the recent release has changed anything. For a lot of the rest of us, including shell integration programmers, cryptographers and the terminally curious, the new information is fascinating and potentially useful, although not always as complete as we'd like.

What's New?

To begin with, the list of nearly 300 new entries in the index contains not only functions, but also COM interfaces, constants, registry key values and overviews. The header files and import libraries have also been updated with the new items. The new information for CryptoAPI and Multimedia seems particularly well done, while information regarding the shell functions is, from our perspective, often undercooked

The new Platform information falls into roughly four categories: the shell, the CryptoAPI, Multimedia and the runtime library

Category 1: The Shell

The newly documented shell functions form a large category that includes functions directly related to the Windows Explorer's business, as well as utility and helper functions. Interested individuals and groups outside of Microsoft have documented many of these items before. As such, the main effect of including them as part of the Platform SDK is likely to be their implicit "blessing" by Microsoft for widespread use or, in certain cases, a clear warning of coming deprecation

The single most important group of functions to have been documented are those that support the default shell folder view object. This is an object that is created using the `SHCreateShellFolderView` and `SHCreateShellFolderViewEx` functions. By passing a pointer to your own `IShellFolder` implementation and a pointer to a callback function, you can get Windows to do the dirty work of creating both the actual window and implementing some closely-related interfaces like `IShellView`. The window will automatically handle painful tasks like changing the toolbar and switching between "Large Icons" and "Small Icons". Microsoft itself makes fairly extensive use of these functions

The internal shell icon cache has now been documented as well. Windows uses this cache to store all the images it needs to draw the desktop and all open folders. By storing them all in one place, the operating system doesn't have to keep opening each individual file just to extract the appropriate icon.

The SDK includes helper functions for a variety of other shell integration tasks as well, including drag and drop, handling variable-sized arrays of pointers and structures, pathname and ITEMIDLIST manipulation. There are also property sheet utilities that are useful for implementing your own shell views.

To get started with the default shell view, look at `SHCreateShellFolderView`. Then, go to http://www.codeproject.com/shell/shlext.asp for a more thorough look at the same information. For drag and drop helpers, start with `DAD_AutoScroll`. For the shell icon cache, look at `IShellImageStore`. For property sheets, try `SHOpenPropSheetW`.

Category 2: Crypto functions

Many of the cryptography functions that were added to the August Platform SDK release are recommended for internal use only and are likely to go away, but for every such function, the SDK documentation suggests another that achieves the same result. Nevertheless, information on the internal functions is still useful to those of you interested in Windows security, because of the visibility it provides to the internals thereof. There is one set of newly documented internal crypto functions for which no alternative is suggested: APIs for administering the cryptography catalog.

For a glimpse at Security internals, try looking at `WTHelperGetProvSignerFromChain` or `WintrustLoadFunctionPointers`. For new XP functionality, look at `CredUIReadSSOCredW`. For administration of the cryptography catalog, start with `CryptCATAdminAddCatalog`.

Category 3: Multimedia

The newly published multimedia APIs are quite well documented. Much of information provided describes the architecture of source filters under Windows Media Player 6.4. While these filters are described as "legacy", the help entry goes on to declare that these are still the default source filters used in the playback of .wma, .wmv and .asf files. However, more recent multimedia information can be found in the documentation for the DirectX APIs.

To dig into the new multimedia information, start at the documentation for "Windows Media Source Filter"

Category 4: RTL

The language-independent runtime library (RTL) functions are low-level services provided by the operating system to compilers for use in the compiled output Because these are low-level, there is usually no language-specific runtime library that would bother wrapping them. RTL functions are rarely used by programmers directly, but are instead intended to be used in code generated by compilers. The newly-documented functions include simple string and memory operations, as well as which is used for exception handling. Most can be easily recognized by the prefix "Rtl". Unless you're writing a compiler, you aren't likely to need these functions.

Top Ten Previously Undocumented Things

Here is a list of cool, useful and funny things you'll find in the August 2002 SDK update. Each has previously been considered "internal" by Microsoft, and could still disappear in future versions of Windows.

10. Function `PerUserInit`

```
void PerUserInit();
```

The `PerUserInit` function sets up "My Documents" and performs other per-user initialization the first time a specific user logs into a specific computer. But, as the documentation says, "Applications do not need to call this function because the OS already does so "

For more information, click reference here.

9. The Shell Folder Band Object

The documentation tells us that the Quick Launch Bar is an example of a folder band. It also says that the Shell Folder Band object has a `CLSID` of `CLSID_ISFB` and, implements the `IShellFolderBand` interface and is used to manage your folder band collection. It does not, unfortunately, tell us how to use the `ShellFolderBand` object to do any of these things

For more information, click reference here.

8. Function AsyncInstallDistributionUnit

```
HRESULT AsyncInstallDistributionUnit(
        LPCWSTR szDistUnit,
        LPCWSTR szTYPE,
        LPCWSTR szExt,
        DWORD dwFileVersionMS,
        DWORD dwFileVersionLS,
        LPCWSTR szURL,
        IBindCtx* pbc,
        LPVOID pvReserved,
        DWORD flags
);
```

We're not entirely sure what this function does or how it's used, and the documentation certainly doesn't say, but our guess is that it's used to download and install handlers for particular types of MIME data. `AsyncInstallDistributionUnit` is implemented on every version of Windows starting with Windows 95 and including Windows CE, so whatever it does, it seems to be fairly popular.

For more information, click reference here

7. Function `PathIsSlow`

```
BOOL PathIsSlow(LPCTSTR pszFile, LPCTSTR pszIconPath);
```

The `PathIsSlow` function returns `TRUE` if the file passed as the first argument is on a high-latency network connection. Before opening a file, you can first call this function and then, when TRUE, ask the user if he's really, really sure he wants to open that 4 MB file over a 28.8 Kbps line.

For more information, click reference here.

6. Interface `IInsertItem`

```
interface IInsertItem : public IUnknown {
        // IInsertItem methods
```

```
        STDMETHOD (InsertItem)(LPCITEMIDLIST pidl);
);
```

It's hard to imagine a more abstract interface than this. A single method serves to insert an `ITEMIDLIST` into an arbitrary list. Even the new documentation fails to specify which objects implement this interface, but it's not hard to imagine that IInsertItem has something to do with iterating over the contents of a shell folder

For more information, click reference here.

5. Interface `IActiveDesktopP`

```
interface IActiveDesktopP : public IUnknown {
        // IActiveDesktopP methods
        STDMETHOD SetSafeMode(DWORD dwFlags);
        STDMETHOD EnsureUpdateHTML();
        STDMETHOD SetScheme(LPCWSTR pwszSchemeName,
            DWORD dwFlags);
        STDMETHOD GetScheme(LPWSTR pwszSchemeName,
            DWORD *lpdwcchBuffer, DWORD dwFlags);
);
```

'Safe Mode', in this case, appears to lock the Active Desktop from changes. The documentation describes only the `SetSafeMode` method of `IactiveDesktop` and implies that there is no way to un-set Safe Mode, although the header file adds flags like `SSM_CLEAR` and `SSM_REFRESH`.

For more information, click reference here.

4. Functions `RegisterShellHookWindow` and `UnregisterShellHookWindow`

```
BOOL RegisterShellHookWindow(HWND hWnd);
BOOL UnregisterShellHookWindow(HWND hWnd);
```

This pair of functions registers and unregisters a specified window to receive certain messages for shell notifications. The messages received are similar to those that can be received after calling the `SetWindowsHookEx` function and specifying `WH_SHELL` for the hook type.

For more information, click reference here.

3. Function `SHCreateStdEnumFmtEtc`

With the release of the August 2002 SDK Update, the shell now probably contains every COM helper function you ever wished Microsoft would provide, but were too lazy to

write `SHCreateStdEnumFmtEtc` is an especially useful function that takes an array of `FORMATETC` structures and returns an `IEnumFORMATETC` pointer. `IEnumFORMATETC` must be implemented by all data objects to support calls to `IDataObject::EnumFormatEtc`.

For more information, click reference here

2. Functions `ReadCabinetState` and `WriteCabinetState`

```
BOOL ReadCabinetState(CABINETSTATE* lpState, int cbLength);
BOOL WriteCabinetState(CABINETSTATE* lpState);

typedef struct {
     WORD cLength;
     WORD nVersion;
     BOOL fFullPathTitle:1;
     BOOL fSaveLocalView:1;
     BOOL fNotShell:1;
     BOOL fSimpleDefault:1;
     BOOL fDontShowDescBar:1;
     BOOL fNewWindowMode:1;
     BOOL fShowCompColor:1;
     BOOL fDontPrettyNames:1;
     BOOL fAdminsCreateCommonGroups:1;
     UINT fUnusedFlags:7;
     UINT fMenuEnumFilter;
} CABINETSTATE;
```

These functions work with the newly documented `CABINETSTATE` structure to update some of the global "Folder Options" for the shell.

For more information, click reference here.

1. Function `CheckNameLegalDOS8Dot3`

A function to check the format of a DOS 8.3 filename, and newly added in Windows XP, seems an ironic addition to an operating system said to mark the final demise of DOS.

For more information, click reference here

State of the Documntation

Once you've looked at a certain amount of Microsoft documentation, differences in formatting really begin to stand out. To the knowledgeable programmer, the "vintage" of a particular entry or section is clearly identifiable by the style in which it is formatted. Most of the new Platform SDK entries share a common formatting template. One of the most obvious characteristics of this template is that there is no longer any mention of the Windows 9x series. In fact, many of the functions that are only now being documented were first present in Windows 95 And yet, their "Minimum operating systems" entry

reads "Windows 2000". Microsoft would claim that coverage of release prior to Windows 2000 is not required by the consent decree

For more information, click reference here

Here are some of the other flaws that we noticed after just a little bit of research:

- Missing Topics. Henk Devos claims at http://www.theregister.co.uk/content/4/26803.html to have discovered two new interfaces, `IDelegateFolder` and `IBrowserFrameOptions` in the SDK, but neither is included in the SDK documentation. Microsoft claims that neither is required by the consent decree to be documented, but that answer won't satisfy those looking for more insight into what appear to be IE hooks into the operating system
- Incorrect information. We've already noted that the "Minimum operating system" specified for a function is often wrong in the SDK documentation We've also found that better information can sometimes be found elsewhere `SHFormatDrive`, for instance, is more accurately and more completely documented by than Microsoft at http://support.microsoft.com/default.aspx?scid=KB;EN-US;q173688&The SDK entryfor `DSA_Create` is also subtly wrong, and the difference between Dynamic Structure Arrays and Dynamic Pointer Arrays was not clear to us until we found better documentation at http://www.geocities.com/SiliconValley/4942/arrays.html
- Obsolete Topics. Some topics are clearly marked as going away in a future version, so be very sure of what you're getting yourself into before using them. `RegisterShellHookWindow` and `IsHungAppWindow` are in this category, among others.
- Alternative Topics. Some of the topics have recommended alternatives. `CertGetNameString` should be used with the `CERT_NAME_FRIENDLY_DISPLAY_TYPE` flag instead of calling `GetFriendlyNameOfCert` directly.
- Incomplete Topics. As an example of an incomplete topic, the entry for `ITargetFrame` doesn't have information about how the interface is used or which objects implement it. Eight of the fourteen methods are listed as "Not currently implemented."

Conclusion

Otto von Bismarck said it best, "If you like laws and sausages, you should never watch either one being made." Ironically, it's the law that has determined that Microsoft reveal just a bit more about how they make their own sausages than anyone else. As a chef that uses a lot of Microsoft sausage in my own recipes, I appreciate having as much insight as I can get into how they're made I may well decide to avoid the obsolete sausages in favor of the more stable, well-known and well-supported sausages, but I feel much better having the choice.

References

- Undocumented Windows 95 Dynamic Array Routines
- Namespace extensions - the undocumented Windows Shell
- Namespace Extensions: the IDelegateFolder mystery

Chris Sells is an independent consultant specializing in distributed applications in .NET and COM, as well as an instructor for DevelopMentor. He is hosting the upcoming Web Services DevCon.

Curt Hagenlocher is responsible for research and development at Motek, Inc. and has over fifteen years of programming experience.

Return to .NET DevCenter