Windows Server 2003 SP1 SDK Release Notes

The Microsoft® Platform SDK provides developers with documentation, header files, and sample code necessary to write software for Microsoft Windows® and Microsoft Windows NT®. The Platform SDK simplifies installation by integrating components from different SDKs and installing them in common paths on your hard disk. It can also set the search paths used by Microsoft Visual Studio®.

File System Layout

By default, the Platform SDK is installed to your hard disk in the locations described in the following table. This list is not complete, but covers the most common directories.

Directory	Contents
\Bin	Platform SDK tools
\Bin\Win64	Tools for building applications for the Itanium® architecture, including an x86-IA64 cross-compiler and specific release notes
\Bin\Win64\AMD64	Tools for building applications for the x64 architecture
\Bin\Win64\ia64	Tools for building applications for the Itanium® architecture
\Bin\Win64 \x86\AMD64	Tools for building applications for the x64 architecture, including an xX86-AMD64 cross-compiler and specific release notes
\Help	Platform SDK documentation
\Include	Header and IDL files to support development, including ATL, CRT, and MFC
\Lib	Import libraries and TLB files
\Lib\ia64	Import libraries and TLB files to support development for the Itanium® architecture
\Lib\AMD64	Import libraries and TLB files to support development for the AMD64 architecture
\License	Platform SDK license information
\NoRedist\Win64	DLL and PDB files to support development for the Itanium® architecture - Not for Redistribution
\NoRedist\Win64 \AMD64	DLL and PDB files to support development for the AMD64 architecture - Not for Redistribution
\Redist\Win64	DLL files to support development for the Itanium® architecture
\Redist\Win64\AMD64	DLL files to support development for the AMD64 architecture
\Samples	Platform SDK samples
\Src	Source code for ATL, CRT, and MFC for 64 bit development

Prerelease Content

This edition of the Platform SDK may include some pre-release content, enabling you to start developing your application before the technology is released. However, technologies can change prior to their final release. As a result, before you can release your applications, you must fully test them with the final released software. Please refer to the *License Agreement* for more details about released and prerelease content in the Platform SDK.

The documentation distinguishes prerelease content from released content. Additionally, some prerelease material is a preview of new functionality not yet available in the operating system.

Building Applications

The details of configuring compiler and linker options are described in the *Getting Started* section of the documentation as well as in Win32.mak, which is located in the %MSSDK%\include directory. You can use one of the following methods to build applications from the command prompt using your own makefiles.

The following are important notes for building applications with the Platform SDK.

- The Platform SDK includes several shortcuts in the **Start** menu that open command windows that are configured for building applications. These command windows allow you to target development for particular versions of Windows, and to specify whether the application is to be built with debugging enabled (Debug) or disabled (Retail).
- By default, Win32.mak and SetEnv.bat target Windows XP (32-bit) and Internet Explorer IE 6.0. You can also use these files to target additional versions of Windows. For more information, read the comments included in these files.
- To target a platform other than the ones covered by the command windows or Setenv.bat, set the include and lib environment variables to the directories that contain the header files and libraries to be used, and update either your makefile or source files to define the _WIN32_IE, _WIN32_WINNT, and _WIN32_WINDOWS macros as appropriate. For example, to target an older platform such as Windows NT version 4.0, set the include and lib environment variables to %MSSDK%\include and %MSSDK%\lib respectively, then set _WIN32_WINNT to 0x0400. For more information, see *Getting Started* in the Platform SDK documentation.
- To develop an application that runs on multiple platforms, you must target the least common denominator. For example, to target Windows XP, Windows 2000, and Windows NT version 4.0, do not use the prerelease header files or

libraries. Furthermore, set _WIN32_WINNT to 0x0400 and set _WIN32_IE for the required version of Internet Explorer. As a result, you cannot take advantage of the new features of Windows XP.
- It is possible to develop an application for the latest version of Windows using a computer that is running an older version of Windows. For example, you can build an application that uses features of Windows XP on a computer running Windows 98. However, the application will not run on Windows 98 and must be tested on a computer running Windows XP.

Visual C++® Search Paths and Registering Environment Variables

Selecting to Register Environment Variables places the Platform SDK bin, include, and library directories at the beginning of the search paths used when building programs in the Visual Studio IDE. (Note that to target a specific version of Windows, you must still define the _WIN32_WINNT, _WIN32_IE, and _WIN32_WINDOWS macros as appropriate.)

To register the SDK bin, include, and library directories with Microsoft Visual Studio® version 6.0 and Visual Studio .NET, click Start, point to All Programs, point to Microsoft Platform SDK for Windows Server 2003 SP1, point to Visual Studio Registration, and then click Register PSDK Directories with Visual Studio. This registration process places the SDK bin, include, and library directories at the beginning of the search paths, which ensures that the latest headers and libraries are used when building applications in the IDE.

Note that for Visual Studio 6.0 integration to succeed, Visual Studio 6.0 must run at least once before you select Register PSDK Directories with Visual Studio. Also note that when this option is run, the IDEs should not be running.

To develop a 32-bit C/C++ application on 64-bit Windows, do not register environment variables when you install Visual C++ 6.0. Instead, open a command window and run Vcvars32.bat (from the Visual C++ \bin folder), followed by Setenv.bat (from the SDK bin folder), specifying the appropriate switches (such as /SVR32 /2000 /XP32).

Tested Compilers

The Platform SDK has been tested with Visual Studio .NET 2003 and the 64-bit cross compilers included within the Platform SDK. For more information about Visual Studio, including the latest service packs, see the Microsoft Visual Studio home page at http://msdn.microsoft.com/vstudio/

The C/C++ SDK samples can be built with other compilers, but these are not completely tested by the SDK team. If you are using Visual C++ 6.0, it is recommended that you upgrade because the import libraries included with this release of the Platform SDK have a different format than that used by Visual C++ 6.0. For information at the current level of support for Visual C++ 6.0; please refer to http://support.microsoft.com/default.aspx?id=fh;[ln];lifeprodv

Development Notes

The following notes describe issues with the Platform SDK.

General Notes

- If you get a linker error to the effect that "__SEH_prolog" or "__SEH_epilog" is unresolved, temporarily add Sehprolg.obj to your link line. The SDK was built with the Visual C++ .NET compiler, and as a result some of the generated code for SEH was changed to use these helper functions. After you upgrade to Visual C++ .NET, you can remove this object file from your link line. Sehprolg.obj has been included in the \lib directory of the Platform SDK.

- The C Runtime (CRT) headers and libraries in the Platform SDK available for targeting 64-bit applications (on either x64 or ia64 Windows platforms) are slightly different from the versions of CRT delivered in Visual Studio 2003 for targeting x86 Windows platforms. Writing code that can target x86 using VS 2003 may not work “as is” when targeting 64-bit Windows platforms using the C++ build tools in the Platform SDK.
    - For example, if you have code that utilizes iostream.h; the code is likely to build using the 64-bit build tools in the Platform SDK, but *not* when targeting x86 using VS 2003.
    - The Platform SDK 64-bit build tools use a version of CRT that still has support for the older style iostream.h, (*iostream.h is no longer supported in VS 2003. VS 2003 supports iostream instead.)*
    - As a consequence of this, there are some samples in the PSDK, using iostream.h that build for 64-bit Windows platforms, but not for 32-bit platforms.
    - For more information about differences between iostream.h and iostream, please refer to INFO: Standard C++ Library Frequently Asked Questions link - http://support.microsoft.com/default.aspx?scid=kb;en-us;154419

- **AP Verifier** is not included in the Platform SDK, it is part of the application compatibility toolkit and can be downloaded at the following URL.. http://www.microsoft.com/downloads/details.aspx?FamilyId=D7D08414-0136-492F-9AD9-CE8AEB7500C7&displaylang=en

- **IPv6 Internet Connection Firewall** (ICF) API was an interim solution, now replaced by the, new for Windows XP SP2, Windows Firewall. The API documentation, headers and libraries for the IPv6 Internet Connection Firewall SDK are still available as a related component of the Advanced Networking Pack for Windows XP.

- **CheckV4.exe -** There is a porting guide available ("IPv6 Guide for Windows Sockets Applications") which references the checkv4 utility and gives examples of IP agnostic code (code samples and such).
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/ipv6_guide_for_windows_sockets_applications_2.asp

Samples

- When building RPC and COM applications you need to use the proper MIDL depending upon your target CPU. When invoking MIDL.exe to build an application with the Platform SDK be sure to use:
  - the /x64 flag when targeting an x64 platform
  - the /ia64 flag when targeting an ia64 platform,
  - the /win32 flag when targeting an x86 platform.

- Using msvctrd.dll
  - A few samples may fail to run because msvctrd.dll cannot be found.
  - This can occur when targeting a debug build of a sample (or any code for that matter) either on ia64 or x64 versions of Windows.
  - This is because, even though the PSDK ships this file for both ia64 and x64, it is not part of the PSDK command

environment PATH. To workaround this issue either set your PATH to reference either %MSSDK%\NoRedist\Win64 \AMD64 and %MSSDK%\NoRedist\Win64 depending on whether you are running on an x64 or ia64 version of Windows.

- Some of the Visual Basic samples have command-line makefiles. To build these samples from the command line, you must specify the tools directory in your path.

- Some TAPI sample only build for x86
    - NetDS\Tapi\Tapi3\Cpp\Incoming
    - NetDS\Tapi\Tapi3\Cpp\Msp\MSPBase
    - NetDS\Tapi\Tapi3\Cpp\Msp\SampleMSP
    - NetDS\Tapi\Tapi3\Cpp\Outgoing
    - NetDS\Tapi\Tapi3\Cpp\pluggable
    - NetDS\Tapi\Tapi3\Cpp\tapirecv
    - NetDS\Tapi\Tapi3\Cpp\tapisend

- The following include files can be found by installing the Build Environment for the Internet Development SDK; schannel.h, schnlsp.h, xenroll.h, and shlwapi.h.

- Some Samples Need the .NET Framework SDK to Build
    - \Multimedia\WindowsMediaServices9\CacheProxy
    - \Multimedia\WindowsMediaServices9\Logging
    - \NetDS\MessageQueuing\csharp_draw
    - \Security\capicom\c_sharp\chainsh
    - \Security\capicom\c_sharp\storesh
    - \TabletPC\AutoClaims
    - \TabletPC\AutoClaims15
    - \TabletPC\InkClipboard

    - \TabletPC\InkCollection
    - \TabletPC\InkDivider
    - \TabletPC\InkErase
    - \TabletPC\InkHitTest
    - \TabletPC\InkRecognition
    - \TabletPC\InkZoom
    - \TabletPC\ScannedPaperForm
    - \TabletPC\Serialization
    - \UDDI\publish
    - \UDDI\runtimeclient
    - \UDDI\wsdldiscovery

- Visual Basic 6.0 Samples are not supported on 64-bit Windows.
- The following samples fail to build properly under any 64-bit Retail build environment but do build correctly under a 64-bit Debug build environment.
    - netds\MessageQueuing\MQPers\Snd
    - netds\MessageQueuing\MQPers\Rcv
    - netDS\MessageQueuing\MQPers\Graphobj

Working with DirectShow requires DirectX 9.0 SDK

To build the DirectShow samples, you must also install the DirectX 9.0 SDK Update (February 2005) or later. The DirectX SDK can be downloaded from http://www.microsoft.com/downloads/details.aspx?familyid=77960733-06e9-47ba-914a-844575031b81&displaylang=en

Obtaining Headers for Rights Management APIs

The headers for the Rights Management APIs mentioned in the documentation are available from the Rights Management Services (RMS) client SDK, available from http://go.microsoft.com/fwlink/?LinkId=17148 . The RMS SDK also includes sample code demonstrating the use of these APIs.

Release Note for MSXML SDK in PSDK

MSXML development requires that you obtain and install a copy of the version-specific MSXML SDK targeted by your application. To do so, you can select and download the current updated version of MSXML 3.0 or MSXML 4.0 as appropriate from the Microsoft Web site.

For MSXML 3.0 Service Pack 5 (SP5), the following download is available:

· **MSXML 3.0 SP5 Software Development Kit (SDK)**

- http://www.microsoft.com/downloads/details.aspx?FamilyId=2CF40AE6-368C-4B6B-A185-2DFA92FB7993&displaylang=en

After obtaining the download, run the MSXML 3.0 SP5 SDK download (xmlsdk.msi) and execute it to install headers and libraries. The MSXML 3.0 SP5 SDK installs these files by default into %ProgramFiles%\Microsoft XML Parser SDK\.

For MSXML 4.0 Service Pack 2 (SP2), the following download is available:

· **MSXML 4.0 SP2 Software Development Kit (SDK)**

- http://www.microsoft.com/downloads/details.aspx?FamilyID=3144b72b-b4f2-46da-b4b6-c5d7485f2b42&DisplayLang=en

After obtaining the download (msxml.msi), complete a custom installation as follows:

1. Run msxml.msi and advance through the End-User License Agreement and Customer Information wizard pages.
2. On the Choose Setup Type page, select **Customize**.
3. In Custom Setup, click **XML SDK** and select **Entire feature will be installed on local hard drive**.
4. Complete setup.

The MSXML 4.0 SP2 SDK installs all header and library files by default into the %ProgramFiles%\MSXML 4.0 directory.

Linker Warnings When You Build C++ Code

SYMPTOMS

You receive one of the following error messages at compile or link time:

Linker Tools Error LNK2001
'unresolved external symbol __security_cookie '

Linker Tools Error LNK2001
'unresolved external symbol __security_check_cookie '

CAUSE

In Visual Studio 2002 a new compiler switch /GS was introduced to the Visual C++ compiler. When this switch is set, the compiler injects buffer overrun detection code in the compiled code. On functions that the compiler thinks might be subject to buffer overruns, the compiler inserts a security cookie before the return address. The security cookie is computed once at module load and its value is pushed to the stack on function entry. Then, on function exit, the compiler helper is called to make sure the cookie's value is still the same. If the value was changed, this is treated as a sign of a buffer overrun in the stack corruption. In this way it is possible to detect some direct buffer overruns into the return address. Eventhough the /GS has been significantly improved in Visual Studio 2003 and Visual Studio 2005, unfortunately it does not protect user code from all possible buffer overrun security attacks. However the list of conditions that /GS can protect user code against has continued to grow and because of its minimal performance impact, the recent versions of Visual C++ compiler set the /GS switch by default. In this way an application receives this buffer overrun protection at runtime and this may significantly reduce number of attacks and exploits to which this application is vulnerable. One can find more details about /GS in the article referenced below.

The Visual C++ compilers that come with the Platform SDK for Windows Server 2003 SP1 sets /GS switch by default. When compiling source code, the compiler introduces references to code that insert a security cookie on the stack on function entry and checks the value of the security cookie on function exit. In addition to this, most of libraries in the Platform SDK have been built with /GS switch set and they already reference code that is required to enable /GS protection at runtime. In Visual Studio the code for /GS resides in the C Runtime Library, which is the default library pulled in by the linker. However the version of the C Runtime Library shipped with the Platform SDK does not contain the same code as the Visual Studio's

CRT. This was done for a number of reasons. One of them is that the Platform SDK allows users to build application and services that can target different modes of the OS. The check of the security cookie added to the compiled code has to be done in different ways depending on what OS mode a service or an application is running in. Because of this reason and several others, it was decided to provide three libraries that implement verification of the security cookie.

bufferoverflowU.lib	This library implements functionality for security cookie verification that can be used in the user mode and in applications that use Win32 API. Most of applications will link to this library.
bufferoverflowK.lib	This library implements check of a security cookie that works in the kernel mode of the OS. Services and subsystems that run in the kernel mode should be linked to this library.
bufferoverflow.lib	This library implements functionality for security cookie verification that can be used in the user mode. However it is different from bufferoverflowU.lib because it can be used in services and applications that do not use Win32 API.

RESOLUTION

To resolve errors thrown by the linker, you need to link your project using with one of the bufferoverlow*.lib listed above. For example, let's assume there is the following simple program, which we would like to build with the Platform SDK:

```
#include <stdio.h>
#include <string.h>

void foo(char* szIn)
{
  char szBuf[10];
  strcpy(szBuf, szIn);
  puts(szBuf);
}
int main(int argc, char* argv[])
{
    if(argc>1)
      foo(argv[1]);
}
```

This example illustrates incorrectly written code that asks for buffer overrun protection. The Visual C++ compiler will add this protection to the compiled code which will fail to link because the linker is not able to resolve the reference to __security_cookie and __security_check_cookie:

```
>cl a.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 14.00.40310.32 for AMD64
Copyright (C) Microsoft Corporation.  All rights reserved.
```

```
a.cpp
Microsoft (R) Incremental Linker Version 8.00.40310.31
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:a.exe
a.obj
a.obj : error LNK2019: unresolved external symbol __security_cookie referenced in function "void
__cdecl foo(char *)" (?foo@@YAXPEAD@Z)
a.obj : error LNK2019: unresolved external symbol __security_check_cookie referenced in function "void
__cdecl foo(char *)" (?foo@@YAXPEAD@Z)
a.exe : fatal error LNK1120: 2 unresolved externals
```

To resolve these errors you need to specify that the linker should also use bufferoverflowU.lib (assuming that your application targets the user mode and can use Win32 API):

```
>cl a.cpp bufferoverflowU.lib
Microsoft (R) C/C++ Optimizing Compiler Version 14.00.40310.32 for AMD64
Copyright (C) Microsoft Corporation.  All rights reserved.

a.cpp
Microsoft (R) Incremental Linker Version 8.00.40310.31
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:a.exe
a.obj
bufferoverflowU.lib
>
```

Or if you use a two step build process:

```
>cl –c a.cpp bufferoverflowU.lib
Microsoft (R) C/C++ Optimizing Compiler Version 14.00.40310.32 for AMD64
Copyright (C) Microsoft Corporation.  All rights reserved.

a.cpp

>link a.obj bufferoverflowU.lib
Microsoft (R) Incremental Linker Version 8.00.40310.31
Copyright (C) Microsoft Corporation.  All rights reserved.

>
```

REFERENCES

1. /GS (Buffer Security Check), Visual C++ Compiler Options,

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/vclrfgsbuffersecurity.asp

2. Compiler Security Checks in Depth, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcext/html/vxlrfVCProjectEngineLibraryVCCLCompilerToolBufferSecurityCheck.asp

---

Development Notes for AMD® 64(64-bit) and x64 bit Architecture

This edition of the Platform SDK enables you to target 64-bit Windows on AMD 64 Architecture and x64.

The Platform SDK includes additional release notes for the **Microsoft® C/C++ Compiler and Debugger (Targeting AMD64 Bit and x64 Architecture).** These release notes are located in the same directory as the compiler \Bin\Win64\x86\amd64. For this release not all documentation has been brought up to date with the name x64.

The release notes also have some minor corrections:

- The document fails to mention that SP1 is supported.
- The document incorrectly states the location of the atl, crt and mfc include files - please refer to the table at the beginning of this document.
- DX 9 is no longer in Beta.

x64 bit Build Environments Identical

The x64 build environment for XP and for Windows Server 2003 SP1 (both retail and debug) are identical. This means that when building a sample using either of these build environments, the generated output files will be placed in the same directory. Thus for example if you first build a sample targeting Windows Server 2003 SP1 (debug) for x64 and then later try to build the same sample for XP (debug) for x64, the sample may not get rebuilt and thus you may end up trying to execute a Windows Server 2003 SP1 .exe on XP; which won't work in general.

Development Notes for Itanium® (64-bit)

Building Itanium® 64-bit Applications with the Platform SDK

The Platform SDK includes header files to target versions of 64-bit Windows. Products built with these header files may be distributed pursuant to the terms and conditions of the Platform SDK end user license agreement.

The Platform SDK includes additional release notes for the **Microsoft® C/C++ Compiler and Debugger (Targeting IA 64 Bit Architecture).** These release notes are located in the same directory as the compiler. There are several issues with thecompiler docs:

- The document fails to mention that SP1 is supported.
- The document incorrectly states the location of the atl, crt and mfc include files - please refer to the table at the beginning of this document.

The following are important notes for building 64-bit applications.

- The tools for developing 64-bit applications are supported only on Windows XP, and Windows Server 2003.
- To set your development environment to target 64-bit Windows, you must either open the *Windows XP 64-bit Build Environment* command window or run the Setenv.bat batch script with the /SVR64 switch. This command window is available on Windows 2000 and Windows XP. Additionally, Setenv.bat returns an error if you use the /SVR64 switch on any version of Windows other than Windows 2000 or Windows XP.
- Applications built to target 64-bit Windows XP do not run on any other version of Windows.
- When performing a debug build of a 64-bit application, disable incremental linking. Otherwise, running the application causes an application violation.
- For details about the 64-bit development tools (compiler and debugger), install the Platform SDK, then see *Microsoft C/C++ IA-64 Compiler Release Notes*, which is installed to your hard disk at %MSSDK%\Bin\Win64\. .
- To get the correct compiler options to build your code, you should include Win32.mak in your makefiles. Win32.mak is installed in the %MSSDK%\Include directory. If you do not use Win32.mak, you must modify your compiler command line appropriately.
- Typical compiler warnings that indicate improper handling of 64-bit data are the following:

  ```
  warning C4305: 'return' : truncation from 'unsigned __int64 ' to 'long '

  warning C4311: 'type cast' : pointer truncation from 'int *__ptr64 ' to 'int '
  ```

warning C4312: 'type cast' : conversion from 'int ' to 'int *__ptr64 ' of greater size

warning C4313: 'printf' : '%p' in format string conflicts with argument 2 of type '__int64 '

warning C4318: passing constant zero as the length to memset

warning C4319: '~' : zero extending 'unsigned long ' to 'unsigned __int64 ' of greater size

warning C4242: '=' : conversion from 'unsigned int' to 'unsigned short', possible loss of data

warning C4244: 'return' : conversion from '__int64' to 'unsigned int', possible loss of data

- If you reference Win32.mak in your makefile and use the $(cflags) environment variable, some compiler warnings will be suppressed. For the list of suppressed warnings, see %MSSDK%\Include\Pre64Pra.h. To re-enable these warnings, just comment out the pragma.
- For more information about developing applications for 64-bit Windows, see *Development Guides* in the Platform SDK documentation.

Debugging 64-bit Applications

There are two debuggers in the Platform SDK that can be used to debug applications on 64-bit Windows:

- A remote debugger supported only on 64-bit Windows. For more information, see *Microsoft C/C++ IA-64 Compiler and Debugger Release Notes*, which is installed to your hard disk at %MSSDK%\Bin\Win64\readme.doc.
- A 64-bit native package that is required for debugging user-mode applications running on an Intel® Itanium® processor. For more information, see Debugging Tools for Windows.

Redistributables

The Microsoft Platform SDK, February 2001 Edition, was the last to ship redistributable components as an integral part of the SDK. These redistributable components are now available to you and your customers online. For more information, see Platform SDK Redistributables.

Archived or Deprecated Content

Periodically, content is removed from the Platform SDK and placed in an online archive for download. This ensures that the Platform SDK content remains current, while making older content available for customers who need it. To view a complete list of archived content available for download, see Platform SDK Archive.

The current version of the Windows Template Library (WTL), formerly packaged in the Platform SDK, can be downloaded from Windows Template Library Download.

MkTypLib.exe, is no longer included in the Platform SDK as it is obsolete. Please use the MIDL compiler instead. The tool is still available in the the February 2003 Platform SDK available on the web if required.

ProxyCfg.exe is no longer included in the Platform SDK as it is installed by the operating system.

OLE-COM Object Viewer is not supported on Windows 2000 Advanced Server

Manifestchck.vbs has been removed and replaced with MT.exe - the command line syntax would be "mt.exe -manifest <mymanifest.man> -validate_manifest"

Developer Support

For information on Microsoft Developer Support, see the document installed to your hard disk at %MSSDK%\setup \html\support.htm.

License Agreement

The contents included in the Platform SDK are licensed to you, the end user. Your use of the SDK is subject to the terms of an End User License Agreement ("EULA") accompanying the SDK and located in the \License subdirectory. You must read and accept the terms of the EULA before you access or use the SDK. If you do not agree to the terms of the EULA, you are not authorized to use the SDK.

The license agreement is installed to your hard disk at %MSSDK%\license\license.htm.

Feedback

Submit bug reports or suggestions for the Platform SDK to sdkfdbk@microsoft.com.