Process Creation Flags (WinBase.h) - Win32 apps

By Karl-Bridge-Microsoft

Archived: 2026-04-05 22:57:12 UTC

The following process creation flags are used by the **CreateProcess**, **CreateProcessAsUser**, **CreateProcessWithLogonW**, and **CreateProcessWithTokenW** functions. They can be specified in any combination, except as noted.

```
  BOOL creationResult;

creationResult = CreateProcess(
    NULL,                   // No module name (use command line)
    cmdLine,                // Command line
    NULL,                   // Process handle not inheritable
    NULL,                   // Thread handle not inheritable
    FALSE,                  // Set handle inheritance to FALSE
    NORMAL_PRIORITY_CLASS | CREATE_NEW_CONSOLE | CREATE_NEW_PROCESS_GROUP, // creation flags
    NULL,                   // Use parent's environment block
    NULL,                   // Use parent's starting directory
    &startupInfo,           // Pointer to STARTUPINFO structure
    &processInformation);   // Pointer to PROCESS_INFORMATION structure
```

Example from Windows classic samples on GitHub.

Constant/value	Description
CREATE_BREAKAWAY_FROM_JOB 0x01000000	The child processes of a process associated with a job are not associated with the job. If the calling process is not associated with a job, this constant has no effect. If the calling process is associated with a job, the job must set the **JOB_OBJECT_LIMIT_BREAKAWAY_OK** limit.
CREATE_DEFAULT_ERROR_MODE 0x04000000	The new process does not inherit the error mode of the calling process. Instead, the new process gets the default error mode. This feature is particularly useful for multithreaded shell applications that run with hard errors disabled. The default behavior is for the new process to inherit

Constant/value	Description
	the error mode of the caller. Setting this flag changes that default behavior.
CREATE_NEW_CONSOLE 0x00000010	The new process has a new console, instead of inheriting its parent's console (the default). For more information, see Creation of a Console. This flag cannot be used with **DETACHED_PROCESS**.
CREATE_NEW_PROCESS_GROUP 0x00000200	The new process is the root process of a new process group. The process group includes all processes that are descendants of this root process. The process identifier of the new process group is the same as the process identifier, which is returned in the *lpProcessInformation* parameter. Process groups are used by the **GenerateConsoleCtrlEvent** function to enable sending a CTRL+BREAK signal to a group of console processes. If this flag is specified, CTRL+C signals will be disabled for all processes within the new process group. This flag is ignored if specified with **CREATE_NEW_CONSOLE**.
CREATE_NO_WINDOW 0x08000000	The process is a console application that is being run without a console window. Therefore, the console handle for the application is not set. This flag is ignored if the application is not a console application, or if it is used with either **CREATE_NEW_CONSOLE** or **DETACHED_PROCESS**.
CREATE_PROTECTED_PROCESS 0x00040000	The process is to be run as a protected process. The system restricts access to protected processes and the threads of protected processes. For more information on how processes can interact with protected processes, see Process Security and Access Rights. To activate a protected process, the binary must have a special signature. This signature is provided by Microsoft but not currently available for non-Microsoft binaries. There are currently four protected processes: media foundation, audio engine, Windows

Constant/value	Description
	error reporting, and system. Components that load into these binaries must also be signed. Multimedia companies can leverage the first two protected processes. For more information, see Overview of the Protected Media Path. **Windows Server 2003 and Windows XP:** This value is not supported.
CREATE_PRESERVE_CODE_AUTHZ_LEVEL 0x02000000	Allows the caller to execute a child process that bypasses the process restrictions that would normally be applied automatically to the process.
CREATE_SECURE_PROCESS 0x00400000	This flag allows secure processes, that run in the Virtualization-Based Security environment, to launch.
CREATE_SEPARATE_WOW_VDM 0x00000800	This flag is valid only when starting a 16-bit Windows-based application. If set, the new process runs in a private Virtual DOS Machine (VDM). By default, all 16-bit Windows-based applications run as threads in a single, shared VDM. The advantage of running separately is that a crash only terminates the single VDM; any other programs running in distinct VDMs continue to function normally. Also, 16-bit Windows-based applications that are run in separate VDMs have separate input queues. That means that if one application stops responding momentarily, applications in separate VDMs continue to receive input. The disadvantage of running separately is that it takes significantly more memory to do so. You should use this flag only if the user requests that 16-bit applications should run in their own VDM.
CREATE_SHARED_WOW_VDM 0x00001000	The flag is valid only when starting a 16-bit Windows-based application. If the DefaultSeparateVDM switch in the Windows section of WIN.INI is **TRUE**, this flag overrides the switch. The new process is run in the shared Virtual DOS Machine.

Constant/value	Description
CREATE_SUSPENDED 0x00000004	The primary thread of the new process is created in a suspended state, and does not run until the **ResumeThread** function is called.
CREATE_UNICODE_ENVIRONMENT 0x00000400	If this flag is set, the environment block pointed to by *lpEnvironment* uses Unicode characters. Otherwise, the environment block uses ANSI characters.
DEBUG_ONLY_THIS_PROCESS 0x00000002	The calling thread starts and debugs the new process. It can receive all related debug events using the **WaitForDebugEvent** function.
DEBUG_PROCESS 0x00000001	The calling thread starts and debugs the new process and all child processes created by the new process. It can receive all related debug events using the **WaitForDebugEvent** function. A process that uses **DEBUG_PROCESS** becomes the root of a debugging chain. This continues until another process in the chain is created with **DEBUG_PROCESS**. If this flag is combined with **DEBUG_ONLY_THIS_PROCESS**, the caller debugs only the new process, not any child processes.
DETACHED_PROCESS 0x00000008	For console processes, the new process does not inherit its parent's console (the default). The new process can call the **AllocConsole** function at a later time to create a console. For more information, see Creation of a Console. This value cannot be used with **CREATE_NEW_CONSOLE**.
EXTENDED_STARTUPINFO_PRESENT 0x00080000	The process is created with extended startup information; the *lpStartupInfo* parameter specifies a **STARTUPINFOEX** structure. **Windows Server 2003 and Windows XP:** This value is not supported.
INHERIT_PARENT_AFFINITY 0x00010000	The process inherits its parent's affinity. If the parent process has threads in more than one processor group, the new process inherits the group-relative affinity of an arbitrary group in use by the parent.

Constant/value	Description
	Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP: This value is not supported.

On 32-bit Windows, 16-bit applications are simulated by ntvdm.exe, not run as individual processes. Therefore, the process creation flags apply to *ntvdm.exe*. Because ntvdm.exe persists after you run the first 16-bit application, when you launch another 16-bit application, the new creation flags are not applied, except for **CREATE_NEW_CONSOLE** and **CREATE_SEPARATE_WOW_VDM**, which create a new ntvdm.exe.

Requirement	Value
Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	WinBase.h (include Windows.h)
