

# Re: Change redirection of stdout of child process

---

*Source:*

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2005-07/msg00245.htm>

---

- *From:* "Skywing" <skywing\_NO\_SPAM\_@xxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Mon, 11 Jul 2005 14:15:45 -0400
- 

Note that if the remote process caches the values returned by GetStdHandle then the handle switching approach will not operate as expected.

"James Brown" <remove\_james\_dot\_brown7\_at\_virgin\_dot\_net> wrote in message [news:42d299c8\\$0\\$12883\\$cc9e4d1f@xxxxxxxxxxxxxxxxxxxxxx](mailto:news:42d299c8$0$12883$cc9e4d1f@xxxxxxxxxxxxxxxxxxxxxx)

> "Michael Bruschkewitz" <brusch4@xxxxxxx> wrote in message  
> [news:42d26a84\\$0\\$18010\\$9b4e6d93@xxxxxxxxxxxxxxxxxxxxxx](mailto:news:42d26a84$0$18010$9b4e6d93@xxxxxxxxxxxxxxxxxxxxxx)

>> Hello,

>> I redirected stdout/stderr of the child process

>> (CreateProcess, si.dwFlags = STARTF\_USESTDHANDLES) to write directly  
>> into a file.

>>

>> When the file which gets the childs stdout/stderr exceeds a certain  
>> size, I want to rename the file and/or redirect the stdout/stderr to  
>> another file. I do not want to copy the file because of its size.

>>

>> I do not want to redirect stdout to a pipe which is handled by the  
>> parent because I think this would/may cause unnecessary process changes  
>> and therefore is a performance penalty.

>>

>> The parent acts as debugger and gets regularly control over the child.

>>

>> I currently have no idea how/if this is possible but don't want to dig  
>> into the wrong direction.

>>

>> Maybe it would be possible to inject a thread into the childs process  
>> space, but I would like a more simple solution.

>>

>> Regards,

>> Michael Bruschkewitz

>>

>>

>

> You have a problem because there is no mechanism in Windows to  
> change the pipes of a foreign process at runtime (only at creation using  
> the  
> method you have already found).

>

## Re: Change redirection of stdout of child process

> Your first course of action is to use pipes – you really should test this  
> thoroughly and discount it only when you \*know\* it is no good, so don't  
> assume that there is a problem with pipes in Windows.  
>  
> Your suggestion of injecting a thread into the child (which subsequently  
> calls  
> SetStdHandle for STDIN/OUT/ERR) is reasonable but "fiddly".  
>  
> My alternate solution (pretend that I have discounted pipes but you really  
> should profile this option yourself) is to use WriteProcessMemory to write  
> new handles directly into the foreign process address space at runtime.  
>  
> This is basically a task of:  
>  
> 1. Create the handle in \*your\* address space.  
> 2. DuplicateHandle to make the pipe/file/handle/ for the \*child\* process  
> 3. WriteProcessMemory into the Process Environment Block (PEB) of the  
> child process, because this is where the STDIN/OUT/ERR handles live.  
>  
> I assume you already know how to do this, if you don't then its probably  
> best  
> that you go back to the "pure pipes" option.  
>  
> Before you do this you should ReadProcessMemory to get the current  
> handle values, and after writing the new values, call CreateRemoteThread  
> three times,  
> specifying CloseHandle as the thread routine, and a foreign HANDLE as the  
> parameter – this will cause  
> the current handles to be cleaned up correctly. I have left this task up  
> to you.  
>  
> So you basically choose between:  
> A. Proper code injection, using only one thread, but lots of work  
> B. WriteProcessMemory, very little work, but the cleanup = 3 threads  
> instead of 1.  
> You choose.  
>  
> //  
> // Inject a handle from current process to destination, storing the  
> // handle-value at the specified address. The previous handle \*VALUE\* is  
> returned –  
> // this handle is only valid in the foreign process.  
> //  
> static HANDLE InjectHandle(HANDLE hDestProc, HANDLE hHandle, PVOID  
> pWriteAddr)  
> {  
> HANDLE hCurProc = GetCurrentProcess();  
> HANDLE hDuplicate = 0;  
> HANDLE hOldHandle;  
>  
> if(hDestProc == 0 || pWriteAddr == 0)

## Re: Change redirection of stdout of child process

```
> return NULL;
>
> // Nothing to do
> if(hHandle == 0)
> return NULL;
>
> if(!ReadProcessMemory(hDestProc, pWriteAddr, &hOldHandle, sizeof(HANDLE),
> NULL))
> return NULL;
>
> // Duplicate the handle for remote process
> if(!DuplicateHandle(hCurProc, hHandle, hDestProc, &hDuplicate, 0, TRUE,
> DUPLICATE_SAME_ACCESS))
> return NULL;
>
> // Write handle into address space
> if(!WriteProcessMemory(hDestProc, pWriteAddr, &hDuplicate, sizeof(HANDLE),
> NULL))
> {
> CloseHandle(hDuplicate);
> return NULL;
> }
>
> return hOldHandle;
> }
>
> //
> // Inject STDIO handles into the PROCESS_PARAMETERS structure for
> // the destination process. Under NT/2K/XP/NET this structure is
> // always located at 0x20000.. but we try to locate it anyway.
> //
> // We can use this feature to spawn a child with redirected STDIO, but
> // without the limitation of having to let the child inherit all our
> // open file handles (yucky imo)
> //
> BOOL InjectStdHandles(HANDLE hDestProc, HANDLE hStdInput, HANDLE
> hStdOutput, HANDLE hStdError)
> {
> DWORD dwPPAddr = 0x20000;
>
> // Get pointer to PROCESS_PARAMETERS structure for THIS process
> // Assume that it will be in same location for CHILD process
> __asm
> {
> mov eax, fs:[0x30] // Pointer to PEB through TEB
> mov eax, [eax+0x10] // PROCESS_PARAMETERS
> mov [dwPPAddr], eax
> }
>
> if(!InjectHandle(hDestProc, hStdInput, (PVOID)(dwPPAddr+0x18)))
> return FALSE;
```

## Re: Change redirection of stdout of child process

```
>
> if(!InjectHandle(hDestProc, hStdOutput, (PVOID)(dwPPAddr+0x1C)))
> return FALSE;
>
> if(!InjectHandle(hDestProc, hStdError, (PVOID)(dwPPAddr+0x20)))
> return FALSE;
>
> return TRUE;
> }
>
>
> You need to modify the function above to return the current HANDLES
> somehow – i.e. use the return value from InjectHandle and do something
> with them
> (i.e. CreateRemoteThread / CloseHandle). Be careful that the handles
> returned
> by InjectHandle are valid handles (created by you) and not the default
> console handles.
>
> hThread = CreateRemoteThread(hDestProc, 0, 0, CloseHandle, hOldHandle, 0,
> 0, 0);
> CloseHandle(hThread);
>
> James
>
> --
> www.catch22.net
> Free win32 software, sourcecode and tutorials
>
>
>
```

---

### • *Follow-Ups:*

- ◆ **Re: Change redirection of stdout of child process**  
◇ From: James Brown

### • *References:*

- ◆ **Change redirection of stdout of child process**  
◇ From: Michael Bruschkewitz
- ◆ **Re: Change redirection of stdout of child process**  
◇ From: James Brown

- Prev by Date: **Re: Access Violation startig EXE from Login Script**
- Next by Date: **Getting seteuid/setegid functionality out of Windows**

Re: Change redirection of stdout of child process

- Previous by thread: ***Re: Change redirection of stdout of child process***
- Next by thread: ***Re: Change redirection of stdout of child process***
- Index(es):
  - ◆ ***Date***
  - ◆ ***Thread***