Creating a new process, simple?

There are could be different situations when you need to create an additional process from your main process. One of those situations is when you need to isolate some code of your service because that code could be a reason of resource leak or can damage your common process somehow. Another case is when your application needs to perform action that requires changing primary process token. Finally you may need to run a 3rd party application.

From first glance creating a new process is a very simple operation. That is actually true. There are 3 different functions CreateProcess, CreateProcessAsUser and CreateProcessWithLogon in Windows API that you can use to create a new process. However there are situations when they don’t work in the way you could expect.

# How to run a process from the Windows service

To allow several users work with the same computer at the same time Windows isolates processes of each user with special object called ***session***. During system startup it creates session with ID=0. OS uses this session to isolate windows services processes from users’ processes. Also Windows creates another session (ID=1) during startup and run first instance of winlogon.exe under that session. Therefore first logged in (via console) user will work under session with ID=1. When another user wants to login to computer system creates new session for him.

Window always creates a new session before user logs in. Therefore it happens when a new remote desktop connection is established or when user connected via console uses “switch user” function. Once a session is created, Windows runs a new instance of winlogon.exe. So the login screen, where user types his credentials, always relates to its own session.

Every session has own ***windows station***. Windows station is another isolation object that has came to us from older version of system. Every window station contains a dedicated ***clipboard*** and a number of ***desktops***.

Here is the typical hierarchy of sessions, window stations and desktops in Windows Vista and Seven (Windows 7).



In older systems like XP or 2003, it is different. The user logged in via console and the services are running in the same session and interactive services even use the same windows station and the same desktop.

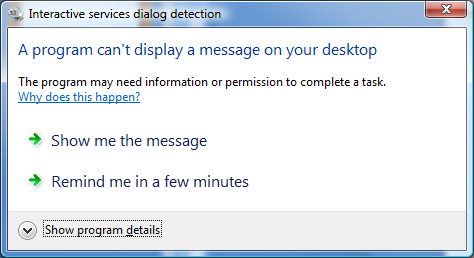


When you are running a process from windows service under user account, that process may not have (usually it doesn’t) access to the window station and to the desktop related to session 0. Of course, if your process doesn’t create any window then you are just lucky. However it’s actually not easy to write an application that doesn’t create any windows at all. Typical Windows application infrastructure is based on a window even if this window is newer shown. The console application written on the most of languages (like Visual C++ or C#) creates a window too. Therefore in 99% of cases our application needs the access to the window station and to the desktop.

|  |
| --- |
| Information |
| When you run a console application and the user under which the process that is running doesn’t have an access to the window station on some versions of OS you may see the error message like this: |

To solve this problem you can add permissions to the window station and the desktop. Usually services run under the separate window station “Service-0x0-3e7$”. However if the option “Allow interact with Desktop” is on for the service then it runs under “WinSta0”. The difference between those windows stations is that “Service-0x0-3e7$” can’t be ever interactive so user will be not able to see the GUI of applications are running there. In case of “WinSta0” user can see the GUI but remember that in Windows Vista/Seven/2008 interactive services and logged in user have different window stations with the same name.

In Windows Vista/Seven/2008, ones a window is created into the Session0\Winsta0\Default , operation system will show a dialog like this. This dialog allows switching active user desktop to Session0\Winsta0\Default.



It’s actually not really important if you service runs in interactive mode or not. You always can specify window station and a desktop while creating a new process. There is a special parameter in StartupInfo structure for this.

You also can use the Developex free tool DevxExec to run a process from the windows service under another user.

DevxExec.exe /user: Administrator /password:qwerty “cmd.exe”

This command automatically grand to user “Administrator” permissions on the appropriate window station and the desktop and run a command shell.

# How to run an application for a user from the Windows service

Let’s see the following scenario. You have a Windows service running. User Marcus has logged to the computer. And you want yours service to start the application, that will be shown to Marcus on his current desktop. In other words, you need to run a new process in another session.

It sounds like an easy task. Yes, you can change a window station and a desktop, but unfortunately, there is no option to specify a session ID to CreateProcess\* functions. However Windows security ***token*** is always related to the session and fortunately we can change the session for the security token. Therefore if we create a new token for our user (for example using LogonUser), then change a session for this token using SetTokenInformation function and finally, call CreateProcessAsUser, the new process will be created in the appropriate session.

Again you can use DevxExec. It has a special parameter /sessionid to specify the session you what to run a new process in. For example this command could start a command shell for a logged in user under Administrator.

DevxExec.exe /user: Marcus /password:qwerty /sessionid:1 “cmd.exe”

# How to run an application for a user from the Windows service (part 2)

Here is another scenario. User Marcus has logged to the computer. And you want your service to start an application as Annette that Marcus can work with on his current desktop. The solution above will work in that scenario too, but only if Annette is a member of the local “Administrators” group. Otherwise she will not have access to the Marcus’s desktop and windows station.

So our service (that runs in session = 0) need to give permission to Annette to Marcus’s desktop (session ≠ 0). The problem is how to get a handle of desktop that lives in another session. If you find out how to do it please let me know☺. Unfortunately I haven’t found how to do it but here is a workaround solution:

1. Get a primary process (OpenProcessToken).
2. Create a primary copy of this token (DuplicateTokenEx)
3. Modify session id of the replica (SetTokenInformation).
4. Create a new temporary process using that taken with CreateProcessAsUser.
5. This process will be run into the Marcus’s session therefore it could access his window station and his desktop and give permission to Annette.

DevxExec works approximately that way so you can a process for either user in either session.

To be continued…

|  |
| --- |
| *Mike Makarov*  *Developex CTO* |