**APL+Win running under a Windows Service**

When the APL+Win ActiveX component is instantiated behind a Windows service, it is important to understand the user profile which is loaded by that Windows service. A Windows service runs in the security context of a specified user account. The user name and password are used by the CreateService function at the time the service is installed on the server hardware. The user name and password associated with a Windows service can be modified using the ChangeServiceConfig function. The QueryServiceConfig function may be used to obtain the user name, but not the password associated with a service.

When the Windows service starts the Microsoft Service Control Manager (SCM) uses the credentials of the user account associated with the service, however the SCM loads the user profile associated with the SCM and not the user profile associated with the user account under which the service runs.

This behavior is by Microsoft design and is not a function of APL+Win or APLNext Application Services. Go to [http://msdn.microsoft.com/en-gb/library/windows/desktop/ms686005(v=vs.85).aspx](http://msdn.microsoft.com/en-gb/library/windows/desktop/ms686005%28v=vs.85%29.aspx) for more information.

This means that an application, such as the APL+Win ActiveX component, running under a Windows service will have access to the SCM user profile using the associated Win32 API methods. In this scenario APL+Win use of the Win32 API to obtain user profile information, e.g. from Windows environment variables, may yield unexpected results.

For example ⎕wcall 'ExpandEnvironmentStrings' ‘%appdata%’ (512⍴⎕tcnul) 512 may not return the expected application data folder path specific to the service user’s credential, e.g. ‘myusername.mydomain’. That Win32 API method will return the application data folder path for the SCM user profile, e.g. “c:\windows\system32\config\systemprofile\AppData\Roaming” and not “C:\Users\ myusername\Roaming”.

It is possible to obtain the “%appdata%” folder path for the profile associated with the user account under which the Windows service is running from the from the Windows registry on the target server. The APLNext.GetUserAppDataFolder .Net assembly illustrates how to obtain this folder path from the Windows registry. It may be possible to ‘translate’ the methods used in this C# program to Win32 API methods using the source code for the APLNext.GetUserAppDataFolder .Net assembly or the CSE examples.

**Run the tool as an ActiveX component from APL+Win**

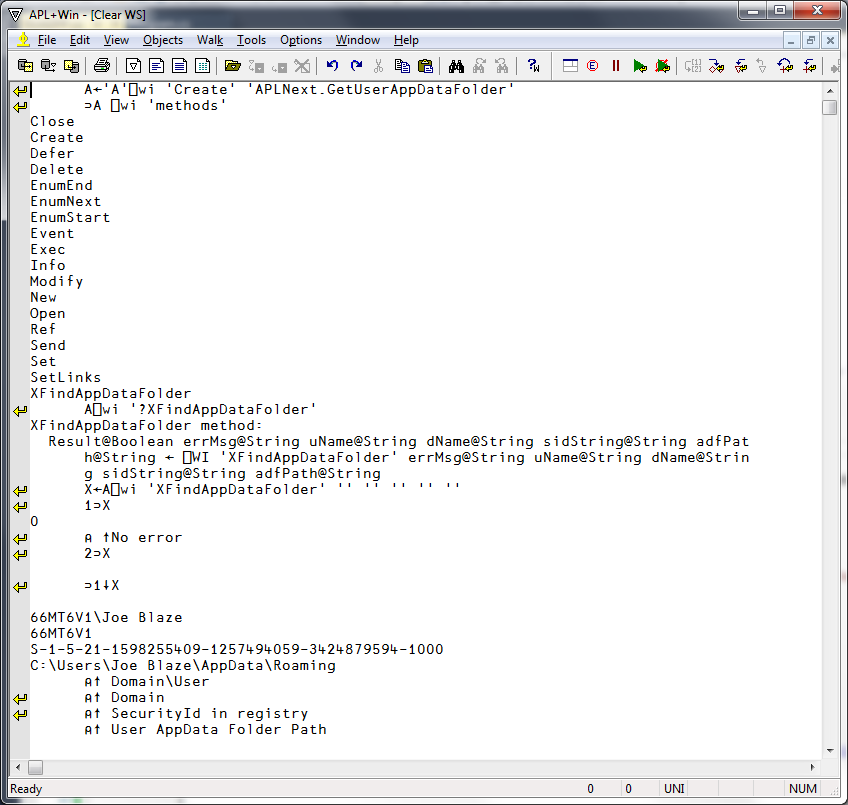
The APLNext.GetUserAppDataFolder ActiveX control must be registered on the target workstation. For a production environment add the COM (ActiveX) registration of the .Net assembly to the installer for the application system. For *ad hoc* use COM registration of the .Net assembly may be done manually using the .Net registration tool "REGASM.EXE" run from the command prompt window with elevated credentials, for example, where "...\" represents the path to the "APLNext.GetUserAppDataFolder.dll" on the target workstation:

|  |
| --- |
| %SystemRoot%\Microsoft.NET\Framework\v4.0.30319\regasm.exe ...\APLNext.GetUserAppDataFolder.dll |

Once the .Net Assembly is COM registered on the target workstation, the APL+Win ⎕wi interface system function can be used with these APL+Win Executable Statements:

|  |
| --- |
| A←'A'⎕wi 'Create' 'APLNext.GetUserAppDataFolder'  ⊃A ⎕wi 'methods'  A⎕wi '?XFindAppDataFolder'  X←A⎕wi 'XFindAppDataFolder' '' '' '' '' ''  1⊃X  ⊃1↓X |

Typical workstation results using the APLNext.GetUserAppDataFolder as an ActiveX component:

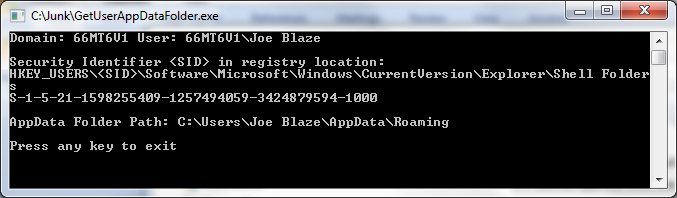


**Run the tool as a Windows executable**

Using Windows Explorer

* Copy the files "APLNext.GetUserAppDataFolder.dll" and "GetUserAppDataFolder.exe" to the desired location on the target workstation
* Double click the file "GetUserAppDataFolder.exe" to run the Windows executable.

Typical results running the Windows executable:



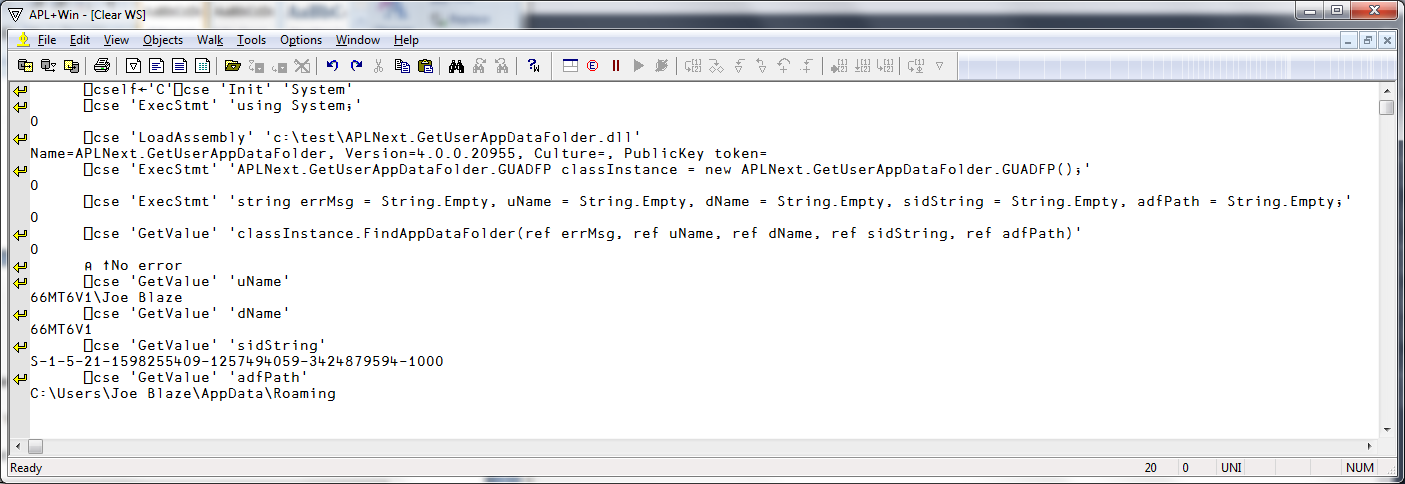
**Use the .Net Assembly with the APLNext C# Script Engine**

Copy the "APLNext.GetUserAppDataFolder.dll" to the desired location on the target workstation.

APL+Win executable statements to use the .Net assembly:

|  |
| --- |
| ⎕cself←'C'⎕cse 'Init' 'System'  ⎕cse 'ExecStmt' 'using System;'  ⎕cse 'LoadAssembly' 'c:\test\APLNext.GetUserAppDataFolder.dll'  ⎕cse 'ExecStmt' 'APLNext.GetUserAppDataFolder.GUADFP classInstance = new APLNext.GetUserAppDataFolder.GUADFP();'  ⎕cse 'ExecStmt' 'string errMsg = String.Empty, uName = String.Empty, dName = String.Empty, sidString = String.Empty, adfPath = String.Empty;'  ⎕cse 'GetValue' 'classInstance.FindAppDataFolder(ref errMsg, ref uName, ref dName, ref sidString, ref adfPath)'  ⎕cse 'GetValue' 'uName'  ⎕cse 'GetValue' 'dName'  ⎕cse 'GetValue' 'sidString'  ⎕cse 'GetValue' 'adfPath' |

Typical work session results:



**Use the underlying C# techniques with the APLNext C# Script Engine**

`APL+Win executable statements to run the underlying C# statements in the CSE:

|  |
| --- |
| ⎕cself←'C'⎕cse 'Init' 'System'  ⎕cse 'ExecStmt' 'using System;'  0  ⎕cse 'ExecStmt' 'using System.Security.Principal;'  0  ⎕cse 'ExecStmt' 'using Microsoft.Win32;'  0  ⎕cse 'ExecStmt' 'string uName = Environment.UserName;'  0  ⎕cse 'ExecStmt' 'string dName = Environment.UserDomainName;'  0  ⎕cse 'ExecStmt' 'uName = dName + @"\" + uName;'  0  ⎕cse 'ExecStmt' 'NTAccount f = new NTAccount(uName);'  0  ⎕cse 'ExecStmt' 'SecurityIdentifier s = (SecurityIdentifier)f.Translate(typeof(SecurityIdentifier));'  0  ⎕cse 'ExecStmt' 'string regKeyFolders = @"HKEY\_USERS\<SID>\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders";'  0  ⎕cse 'ExecStmt' 'string regValueAppData = @"AppData";'  0  ⎕cse 'GetValue' 'Registry.GetValue(regKeyFolders.Replace("<SID>", s.ToString()), regValueAppData, null) as string'  C:\Users\Joe Blaze\AppData\Roaming  ⎕cse 'GetValue' 'uName'  66MT6V1\Joe Blaze  ⎕cse 'GetValue' 's.ToString()'  S-1-5-21-1598255409-1257494059-3424879594-1000 |

Typical workstation results:

