

Re: Prompting for user id/password when using Integrated Security

Source: <http://www.tech-archive.net/Archive/Data/microsoft.public.data.ado/2005-06/msg00254.html>

- *From:* "Stefan Tzanev" <StefanTzanev@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 30 Jun 2005 11:27:03 -0700
-

Daniel,

Thank you for your advice. That's exactly what I did and it works (Using LogonUser and ImpersonateLoggedOnUser API).

For those, interested in the solution, here's an example in VB:

' IMPLEMENTATION (see below for TYPICAL USAGE)

```
'=====
'
'$Workfile: Impersonator.cls $
'
'$Author: $
'$Revision: $
'$Date: $
'
'Description: Encapsulates Win32 impersonation API
'(part of Win32 Authentication API).
'=====
```

Option Explicit

```
' _____ Windows API Declarations _____

Private Declare Function LogonUser Lib "advapi32.dll" Alias "LogonUserA" _
(ByVal lpszUsername As String, ByVal lpszDomain As String, ByVal
lpszPassword As String, _
ByVal dwLogonType As Long, ByVal dwLogonProvider As Long, ByRef phToken
As Long) As Long

Private Declare Function CloseHandle Lib "Kernel32.dll" _
(ByVal hObject As Long) As Long

Private Declare Function ImpersonateLoggedOnUser Lib "advapi32.dll" _
(ByVal hToken As Long) As Long

Private Declare Function RevertToSelf Lib "advapi32.dll" () As Long
```

Re: Prompting for user id/password when using Integrated Security

```
Private Declare Function GetLastError Lib "Kernel32.dll" () As Long
Private Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA" _
(ByVal dwFlags As Long, ByRef lpSource As Long, ByVal dwMessageId As
Long, _
ByVal dwLanguageId As Long, ByVal lpBuffer As String, ByVal nSize As
Long, _
ByRef Arguments As Any) As Long
```

```
' _____ Data definitions _____
' _____ Public Variable Declaration _____
' _____ Private Variable Declaration _____
```

```
Private m_hToken As Long ' handle to the impersonation access token
' that represents a logged-on user
```

```
' _____ Constants _____
```

```
' logon types (original values defined in winbase.h)
Public Enum eLogonTypes
LogonType_Interactive = 2 ' LOGON32_LOGON_INTERACTIVE
LogonType_Network = 3 ' LOGON32_LOGON_NETWORK
LogonType_Batch = 4 ' LOGON32_LOGON_BATCH
LogonType_Service = 5 ' LOGON32_LOGON_SERVICE
LogonType_Unlock = 7 ' LOGON32_LOGON_UNLOCK
LogonType_ClearText = 8 ' LOGON32_LOGON_NETWORK_CLEARTEXT
LogonType_Credentials = 9 ' LOGON32_LOGON_NEW_CREDENTIALS
End Enum
```

```
' logon providers (original values defined in winbase.h)
Public Enum eLogonProviders
LogonProvider_Default = 0 ' LOGON32_PROVIDER_DEFAULT
LogonProvider_WinNT35 = 1 ' LOGON32_PROVIDER_WINNT35
LogonProvider_WinNT40 = 2 ' LOGON32_PROVIDER_WINNT40
LogonProvider_WinNT50 = 3 ' LOGON32_PROVIDER_WINNT50
End Enum
```

```
' FormatMessage flags (original values defined in winbase.h)
Public Enum eFormatMessageFlags
FormatMessage_AllocateBuffer = &H100& ' FORMAT_MESSAGE_ALLOCATE_BUFFER
FormatMessage_IgnoreInserts = &H200& ' FORMAT_MESSAGE_IGNORE_INSERTS
FormatMessage_FromString = &H400& ' FORMAT_MESSAGE_FROM_STRING
FormatMessage_FromHMODULE = &H800& ' FORMAT_MESSAGE_FROM_HMODULE
FormatMessage_FromSystem = &H1000& ' FORMAT_MESSAGE_FROM_SYSTEM
FormatMessage_ArgumentArray = &H2000& ' FORMAT_MESSAGE_ARGUMENT_ARRAY
FormatMessage_MaxWidthMask = &HFF& ' FORMAT_MESSAGE_MAX_WIDTH_MASK
End Enum
```

```
' error codes
```

Re: Prompting for user id/password when using Integrated Security

```
Public Enum eImpersonatorErrors
```

```
UnknownError = vbObjectError + 513
```

```
End Enum
```

```
' _____ Methods _____
```

```
' ~~~~~
```

```
' GetErrorDescription()
```

```
' Args : nErrorNumber – error code
```

```
' Description : Returns an error description that corresponds to an error  
' code.
```

```
' Returns : The error description
```

```
' Notes : Make sure that the Select is updated when the
```

```
' eImpersonatorErrors enum changes.
```

```
'
```

```
' ~~~~~
```

```
Private Function GetErrorDescription(ByVal nErrorNumber As Long) As String
```

```
On Error GoTo GetErrorDescriptionError
```

```
Const sDescrPrefix As String = "Impersonator – "
```

```
Select Case nErrorNumber
```

```
Case Else
```

```
GoTo GetErrorDescriptionError
```

```
End Select
```

```
Exit Function
```

```
GetErrorDescriptionError:
```

```
GetErrorDescription = sDescrPrefix & "unknown error."
```

```
End Function
```

```
' ~~~~~
```

```
'
```

```
' Method : Class_Initialize
```

```
' Description : Default class initialization
```

```
' Notes :
```

```
'
```

```
' ~~~~~
```

```
Private Sub Class_Initialize()
```

```
On Error Resume Next
```

```
m_hToken = 0
```

```
End Sub
```

```
' ~~~~~
```

```
'
```

```
' Method : Class_Terminate
```

Re: Prompting for user id/password when using Integrated Security

' Description : Terminate impersonation

' Notes :

,

Private Sub Class_Terminate()

Destroy

End Sub

,

' Method : Create

' Description : Starts impersonation for a given user account.

' Args : lpszUsername – name of the user account to impersonate

' lpszDomain – name of the domain or server whose account
database contains the lpszUsername account

' lpszPassword – clear-text password for the user account
specified by lpszUsername

' Returns : True if successful, False on error.

,

Public Function Create(ByVal lpszUsername As String, _

ByVal lpszDomain As String, _

ByVal lpszPassword As String) As Boolean

On Error GoTo CreateError

Create = False

Dim nSuccess As Long

' log on as impersonated user

nSuccess = LogonUser(lpszUsername, lpszDomain, lpszPassword, _
LogonType_Network, LogonProvider_Default, m_hToken)

If nSuccess = 0 Then

Call MsgBox(GetLastErrorDescr, vbOKOnly + vbCritical, "LOGON ERROR")

Exit Function

End If

' impersonate

nSuccess = ImpersonateLoggedOnUser(m_hToken)

If nSuccess = 0 Then

Call MsgBox(GetLastErrorDescr, vbOKOnly + vbCritical, "IMPERSONATION
ERROR")

Exit Function

End If

' return success code

Create = True

Exit Function

Re: Prompting for user id/password when using Integrated Security

CreateError:
Exit Function

End Function

```
-----  
,  
' Sub : Destroy  
' Description : Stops current impersonation.  
' Args : None  
' Notes :  
,
```

```
-----  
Public Sub Destroy()
```

```
On Error Resume Next  
Call RevertToSelf  
Call CloseHandle(m_hToken)  
m_hToken = 0
```

End Sub

```
-----  
,  
' Sub : GetLastErrorDescr  
' Description : Gets the system description of the last thread error.  
' Args : None  
' Returns : Error description string  
,
```

```
-----  
Public Function GetLastErrorDescr() As String
```

```
On Error GoTo GetLastErrorDescrError  
GetLastErrorDescr = "Unknown error."
```

```
Dim dwErrorCode As Long ' error code  
dwErrorCode = GetLastError ' get last Windows thread error
```

```
Dim sErrorDescr As String ' error description buffer  
sErrorDescr = Space$(256) ' pre-allocate
```

```
Dim dwLanguageId As Long ' language id  
dwLanguageId = 0& ' neutral language
```

```
' get error description  
Dim nRet As Long  
nRet = FormatMessage(FormatMessage_FromSystem, 0&, dwErrorCode, _  
dwLanguageId, sErrorDescr, 256&, 0&)
```

```
' return error description  
If nRet > 0 Then
```

Re: Prompting for user id/password when using Integrated Security

Re: Prompting for user id/password when using Integrated Security

```
GetLastErrorDescr = Left(sErrorDescr, nRet)
End If
```

```
GetLastErrorDescrError:
Exit Function
```

```
End Function
```

```
'-----
'TYPICAL USAGE (using the Impersonator class):
```

```
'=====
'$Workfile: ImpersonatorClient.frm $
```

```
'$Author: $
'$Revision: $
'$Date: $
```

```
'Description: Database login form that uses Impersonator class.
'
```

```
'-----
' Sub : DoSomethingWithData
```

```
' Description : Connect to the database on the specified db
' server using Windows authentication and do something
with the data.
```

```
' Args : None
```

```
' Notes : This function uses txtServer, txtDomain, txtUserID and
txtPassword
```

```
' TextBox form controls to get the user-specified values
for the db
```

```
' server, the Windows credentials of the impersonated user.
'
```

```
'-----
Private Sub DoSomethingWithData()
```

```
On Error GoTo DoSomethingWithDataError
```

```
Dim conn As ADODB.Connection
Dim oImpersonator As Impersonator
```

```
Set oImpersonator = New Impersonator
```

```
If Not oImpersonator.Create(txtUserID.Text, txtDomain.Text,
txtPassword.Text) Then
```

```
Exit Sub
```

Re: Prompting for user id/password when using Integrated Security

End If

```
' login as current/impersonated user using Windows Authentication
' (that is, set the connection property "Integrated Security" to "SSPI" by
' adding "Integrated Security = SSPI;" to the connection string)
Set conn = OpenConnection(txtServer.Text)
```

```
If conn Is Nothing Then
Call MsgBox("Failed to connect to the specified database.", _
vbOKOnly + vbCritical, "CONNECTION ERROR")
Exit Sub
End If
```

```
' do something with the database data
```

```
' disconnect
On Error Resume Next
conn.Close
Set conn = Nothing
Set oImpersonator = Nothing
Exit Sub
```

```
DoSomethingWithDataError:
Set conn = Nothing
Set oImpersonator = Nothing
Call MsgBox("Unknown error.", vbOKOnly + vbCritical, "ERROR")
Exit Sub
```

End Sub

```
~~~~~
'
' Function : OpenConnection
' Description : Connects to the database on the specified db
' server using Windows authentication.
' Args : sServerName – database server name
' Returns : A reference to the open connection.
' On error returns Nothing.
' Notes : Substitute "MyDatabase" in the connection string with
your actual
' database (catalog) name.
'
~~~~~
```

```
Private Function OpenConnection(ByRef sServerName As String) As
ADODB.Connection
```

```
On Error GoTo OpenConnectionError
Set OpenConnection = Nothing
```

```
' set connection string for Windows authentication
Dim sDataSource As String
```

Re: Prompting for user id/password when using Integrated Security

Re: Prompting for user id/password when using Integrated Security

```
sDataSource = Iif(LCase(sServerName) = "local" Or Len(Trim(sServerName))  
= 0, _  
"(local)", sServerName)
```

```
Dim sConnString As String  
sConnString = "Provider = SQLOLEDB;" & _  
"Data Source = " & sDataSource & ";" & _  
"Initial Catalog = MyDatabase;" & _  
"Integrated Security = SSPI"
```

```
' connect  
Dim conn As ADODB.Connection  
Set conn = New ADODB.Connection  
conn.ConnectionString = sConnString  
conn.CursorLocation = adUseClient  
conn.ConnectionTimeout = 1 ' [secs]  
conn.Open
```

```
' return open connection  
Set OpenConnection = conn  
Set conn = Nothing  
Exit Function
```

```
OpenConnectionError:  
Exit Function
```

```
End Function
```

```
'=====
```

"Daniel Crichton" wrote:

```
> Stefan wrote on Mon, 27 Jun 2005 15:14:03 -0700:  
>  
>> I want to allow a user of an ADO 2.8 application to login to SQL Server  
>> using Windows authentication ("Integrated Security = SSPI") even if she is  
>> not the current Windows user. (For example, prompt the user for her  
>> Windows account information in a way, similar to that used in Windows  
>> Explorer when mapping to a network drive under different user  
>> credentials.) How can I provide explicitly the Windows user id/password  
>> information (different from those of the currently logged in user) when  
>> using Windows authentication to connect to the SQL Server? ("Provider =  
>> SQLOLEDB;Integrated Security = SSPI")  
>> Thank you.  
>  
> Have you looked into using the LogonUser Win32 API call to make the  
> application run with the user credentials entered?  
>  
> Dan  
>
```

>
>
.

• **References:**

- ◆ **Prompting for user id/password when using Integrated Security**
 ◇ From: Stefan Tzanev
- ◆ **Re: Prompting for user id/password when using Integrated Security**
 ◇ From: Daniel Crichton

- Prev by Date: **What happens to non-committed transaction**
- Next by Date: **Re: Cached ADO**
- Previous by thread: **Re: Prompting for user id/password when using Integrated Security**
- Next by thread: **Install MDAC via AD**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**