

## Re: Impersonation with users of 2 domain (nor trusted)

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-10/msg00465.html>

---

- *From:* bruce barker <nospam@xxxxxxxxxxx>
  - *Date:* Mon, 29 Oct 2007 10:32:54 -0700
- 

try logon type interactive which defaults to a primary token, as I don't believe network allows access to network resources on another box even when converted to primary.

-- bruce (sqlwork.com)

DB wrote:

Hi guys,  
I'm trying to solve this problem:  
in my software I need to use files that are in different machine of 2 domain  
at which I could access using different user rights.

If the software run on a machine of domain A and need to access to a file  
with an user of domain B I get the error 1326 "Error during access: unknown  
user or wrong password".  
User and password are right !

The same with machin of domain B and user of domain A.

What's wrong ?

Grazie  
DB

Here's my code:

```
try
{
using (new Impersonation("domenico", "dominioA",
"password"))
{
StreamWriter SW = new
StreamWriter(File.Create(@"\\DEV2\C\PROVA.TXT"));
SW.WriteLine("PROVA");
SW.Close();
}
```

Re: Impersonation with users of 2 domain (nor trusted)

```
}  
}  
catch (Exception ex)  
{  
    MessageBox.Show(ex.Message);  
}
```

```
using System;  
using System.ComponentModel;  
using System.Runtime.InteropServices;  
using System.Security.Principal;
```

```
namespace Utility
```

```
{  
    public enum LogonType : int  
    {  
        /// <summary>  
        /// This logon type is intended for users who will be interactively  
        /// using the computer, such as a user being logged on  
        /// by a terminal server, remote shell, or similar process.  
        /// This logon type has the additional expense of caching logon  
        /// information for disconnected operations;  
        /// therefore, it is inappropriate for some client/server  
        /// applications,  
        /// such as a mail server.  
        /// </summary>  
        Interactive = 2,  
  
        /// <summary>  
        /// This logon type is intended for high performance servers to  
        /// authenticate plaintext passwords.  
        /// The LogonUser function does not cache credentials for this logon  
        /// type.  
        /// </summary>  
        Network = 3,  
  
        /// <summary>  
        /// This logon type is intended for batch servers, where processes  
        /// may be executing on behalf of a user without  
        /// their direct intervention. This type is also for higher  
        /// performance servers that process many plaintext  
        /// authentication attempts at a time, such as mail or Web servers.  
        /// The LogonUser function does not cache credentials for this logon  
        /// type.  
        /// </summary>  
        Batch = 4,  
  
        /// <summary>
```

Re: Impersonation with users of 2 domain (nor trusted)

```
/// Indicates a service-type logon. The account provided must have
the service privilege enabled.
```

```
/// </summary>
```

```
Service = 5,
```

```
/// <summary>
```

```
/// This logon type is for GINA DLLs that log on users who will be
interactively using the computer.
```

```
/// This logon type can generate a unique audit record that shows
when the workstation was unlocked.
```

```
/// </summary>
```

```
Unlock = 7,
```

```
/// <summary>
```

```
/// This logon type preserves the name and password in the
authentication package, which allows the server to make
connections to other network servers while impersonating the
client. A server can accept plaintext credentials
```

```
/// from a client, call LogonUser, verify that the user can access
the system across the network, and still
```

```
/// communicate with other servers.
```

```
/// NOTE: Windows NT: This value is not supported.
```

```
/// </summary>
```

```
NetworkClearText = 8,
```

```
/// <summary>
```

```
/// This logon type allows the caller to clone its current token and
specify new credentials for outbound connections.
```

```
/// The new logon session has the same local identifier but uses
different credentials for other network connections.
```

```
/// NOTE: This logon type is supported only by the
LOGON32_PROVIDER_WINNT50 logon provider.
```

```
/// NOTE: Windows NT: This value is not supported.
```

```
/// </summary>
```

```
NewCredentials = 9,
```

```
}
```

```
public enum LogonProvider : int
```

```
{
```

```
/// <summary>
```

```
/// Use the standard logon provider for the system.
```

```
/// The default security provider is negotiate, unless you pass NULL
for the domain name and the user name
```

```
/// is not in UPN format. In this case, the default provider is
NTLM.
```

```
/// NOTE: Windows 2000/NT: The default security provider is NTLM.
```

```
/// </summary>
```

```
Default = 0,
```

```
}
```

```
public class Impersonation : IDisposable
```

Re: Impersonation with users of 2 domain (nor trusted)

```
{
#region Dll Imports
/// <summary>
/// Closes an open object handle.
/// </summary>
/// <param name="hObject">A handle to an open object.</param>
/// <returns><c>True</c> when succeeded; otherwise
<c>false</c>.</returns>
[DllImport("kernel32.dll")]
private static extern Boolean CloseHandle(IntPtr hObject);

/// <summary>
/// Attempts to log a user on to the local computer.
/// </summary>
/// <param name="username">This is the name of the user account to
log on to.
/// If you use the user principal name (UPN) format,
user@DNSdomainname, the
/// domain parameter must be <c>null</c>.</param>
/// <param name="domain">Specifies the name of the domain or server
whose
/// account database contains the lpszUsername account. If this
parameter
/// is <c>null</c>, the user name must be specified in UPN format.
If this
/// parameter is ".", the function validates the account by using
only the
/// local account database.</param>
/// <param name="password">The password</param>
/// <param name="logonType">The logon type</param>
/// <param name="logonProvider">The logon provides</param>
/// <param name="userToken">The out parameter that will contain the
user
/// token when method succeeds.</param>
/// <returns><c>True</c> when succeeded; otherwise
<c>false</c>.</returns>
[DllImport("advapi32.dll", CharSet=CharSet.Auto, SetLastError=true)]
private static extern bool LogonUser( string username, string
domain,
string password, LogonType
logonType,
LogonProvider logonProvider,
out IntPtr userToken );

/// <summary>
/// Creates a new access token that duplicates one already in
existence.
/// </summary>
/// <param name="token">Handle to an access token.</param>
/// <param name="impersonationLevel">The impersonation
level.</param>
```

## Re: Impersonation with users of 2 domain (nor trusted)

```
/// <param name="duplication">Reference to the token to
duplicate.</param>
/// <returns></returns>
[DllImport("advapi32.dll", CharSet=CharSet.Auto, SetLastError=true)]
private static extern bool DuplicateToken( IntPtr token, int
impersonationLevel,
ref IntPtr duplication );

/// <summary>
/// The ImpersonateLoggedOnUser function lets the calling thread
impersonate the
/// security context of a logged-on user. The user is represented by
a token handle.
/// </summary>
/// <param name="userToken">Handle to a primary or impersonation
access token that represents a logged-on user.</param>
/// <returns>If the function succeeds, the return value is
nonzero.</returns>
[DllImport("advapi32.dll", SetLastError=true)]
static extern bool ImpersonateLoggedOnUser( IntPtr userToken );
#endregion

#region Private members
/// <summary>
/// <c>true</c> if disposed; otherwise, <c>>false</c>.
/// </summary>
private bool _disposed;

/// <summary>
/// Holds the created impersonation context and will be used
/// for reverting to previous user.
/// </summary>
private WindowsImpersonationContext _impersonationContext;
#endregion

#region Ctor & Dtor
/// <summary>
/// Initializes a new instance of the <see cref="Impersonation"/>
class and
/// impersonates with the specified credentials.
/// </summary>
/// <param name="username">his is the name of the user account to
log on
/// to. If you use the user principal name (UPN) format,
/// user@DNS_domain_name, the lpszDomain parameter must be
<c>null</c>.</param>
/// <param name="domain">The name of the domain or server whose
account
/// database contains the lpszUsername account. If this parameter is
/// <c>null</c>, the user name must be specified in UPN format. If
this
```

## Re: Impersonation with users of 2 domain (nor trusted)

```
/// parameter is ".", the function validates the account by using
only the
/// local account database.</param>
/// <param name="password">The plaintext password for the user
account.</param>
public Impersonation( String username, String domain, String
password )
{
IntPtr userToken = IntPtr.Zero;
IntPtr userTokenDuplication = IntPtr.Zero;

// Logon with user and get token.
bool loggedOn = LogonUser( username, domain, password,
LogonType.Network, LogonProvider.Default,
out userToken );

if( loggedOn )
{
try
{
// Create a duplication of the usertoken, this is a
solution
// for the known bug that is published under KB article
Q319615.
if( DuplicateToken( userToken, 2, ref
userTokenDuplication ) )
{
// Create windows identity from the token and
impersonate the user.
WindowsIdentity identity = new WindowsIdentity(
userTokenDuplication );
_impersonationContext = identity.Impersonate();
}
}
else
{
// Token duplication failed!
// Use the default ctor overload
// that will use Marshal.GetLastWin32Error();
// to create the exceptions details.
throw new Win32Exception();
}
}
finally
{
// Close usertoken handle duplication when created.
if( !userTokenDuplication.Equals( IntPtr.Zero ) )
{
// Closes the handle of the user.
CloseHandle( userTokenDuplication );
userTokenDuplication = IntPtr.Zero;
}
}
}
```

Re: Impersonation with users of 2 domain (nor trusted)

```
// Close usertoken handle when created.
if( !userToken.Equals( IntPtr.Zero ) )
{
    // Closes the handle of the user.
    CloseHandle( userToken );
    userToken = IntPtr.Zero;
}
}
}
else
{
    // Logon failed!
    // Use the default ctor overload that
    // will use Marshal.GetLastWin32Error();
    // to create the exceptions details.
    throw new Win32Exception();
}
}

/// <summary>
/// Releases unmanaged resources and performs other cleanup
operations before the
/// <see cref="Born2Code.Net.Impersonation"/> is reclaimed by
garbage collection.
/// </summary>
~Impersonation()
{
    Dispose( false );
}
#endregion

#region Public methods
/// <summary>
/// Reverts to the previous user.
/// </summary>
public void Revert()
{
    if( _impersonationContext != null )
    {
        // Revert to previous user.
        _impersonationContext.Undo();
        _impersonationContext = null;
    }
}
#endregion

#region IDisposable implementation.
/// <summary>
/// Performs application-defined tasks associated with freeing,
releasing, or
```

## Re: Impersonation with users of 2 domain (nor trusted)

```
/// resetting unmanaged resources and will revert to the previous
user when
/// the impersonation still exists.
/// </summary>
public void Dispose()
{
    Dispose( true );
    GC.SuppressFinalize(this);
}

/// <summary>
/// Performs application-defined tasks associated with freeing,
releasing, or
/// resetting unmanaged resources and will revert to the previous
user when
/// the impersonation still exists.
/// </summary>
/// <param name="disposing">Specify <c>>true</c> when calling the
method directly
/// or indirectly by a user's code; Otherwise <c>>false</c>.
protected virtual void Dispose( bool disposing )
{
    if( !_disposed )
    {
        Revert();

        _disposed = true;
    }
}
#endregion
}
```