Connect to a UNC Path with Credentials

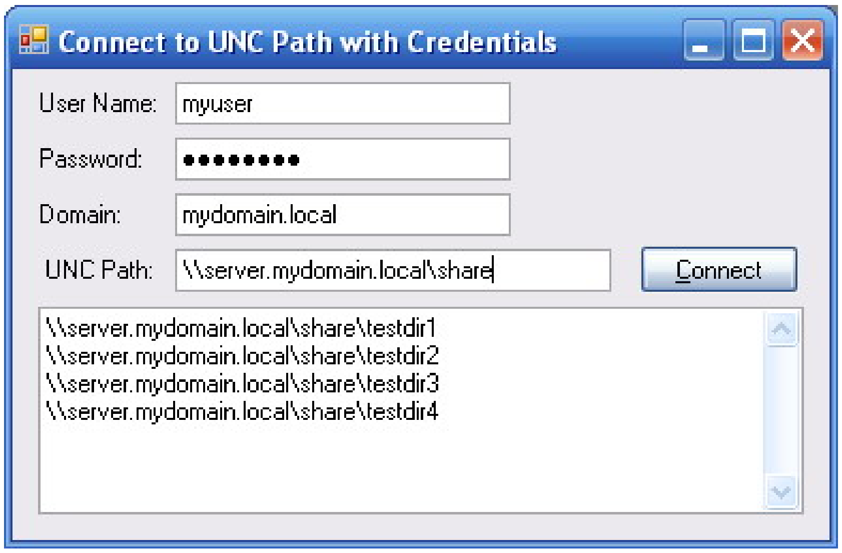
hayes.adrian, 16 Oct 2009 CPOL

Rate:

4.83 (32 votes)

Use NetApi32 to establish and break connections to UNC paths using specified user credentials.

Download source code - 10.7 KB



User Name: myuser
Password: ●●●●●●●●●
Domain: mydomain.local
UNC Path: \\server.mydomain.local\share

```
\\server.mydomain.local\share\testdir1
\\server.mydomain.local\share\testdir2
\\server.mydomain.local\share\testdir3
\\server.mydomain.local\share\testdir4
```

Introduction

This article demonstrates how you can connect to a remote resource via a UNC path and pass user credentials. It implements the `IDisposable` interface so you can use it within a `using()` block.

Background

I had an ASP.NET site where I wanted to access network resources, but did not have sufficient share permissions because the code ran under the ASP user. I also had a service that copied files every night between file shares in two different domains. I wanted a way to access remote resources without opening up security holes by changing permissions or running as a privileged user.

Using the code

The `UNCAccessWithCredentials` class implements the Win32 methods`NetUseAdd()` and `NetUseDel()` to create the remote connections. Connections added with `NetUseAdd` are not visible in Explorer. When active, they will show via the DOS *Net Use* command.

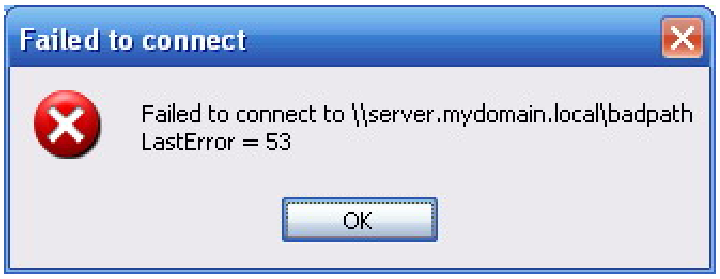
The class also implements the `IDisposable` interface so that we can use a`using()` block. Once the end of the block is reached, the class automatically disconnects the UNC connection in its `Dispose()` methods.

Hide Copy Code

```
using (UNCAccessWithCredentials unc = new UNCAccessWithCredentials())
{
    if (unc.NetUseWithCredentials(uncpath, user, domain, password))
    {
        // Insert your code that requires access to the UNC resource
    }
    else
    {
        // The connection has failed. Use the LastError to get the system error code
        MessageBox.Show("Failed to connect to " + tbUNCPath.Text +
                        "\r\nLastError = " + unc.LastError.ToString(),
                        "Failed to connect",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
// When it reaches the end of the using block, the class deletes the connection.
}
```

If you require persistent connections, declare an instance of the class and use the`NetUseWithCredentials()` method to connect. Once connected, you can access the remote resource at need until disconnected. Do not forget to remove the connection when you are finished using the `NetUseDelete()` method..

Interpreting Errors



Failed to connect to \\server.mydomain.local\badpath
LastError = 53

If the methods fail, they return `false`. If this occurs, use the class' `LastError`property to obtain the Windows System Error Code. You can use the MSDN System Error Codes page to obtain a description of the error.

In the error above, Error 53 is `ERROR_BAD_NETPATH` - "The network path was not found". It looks like I specified the wrong server or share path.

Points of Interest

In researching how to do this task, I found many interesting hacks that got the job done but were not savory. Some used the `LogonUser()` method, which only works if your remote user has logon rights on the executing machine. I also found cases where the programmer ran a shell *net use* command.

Using the API `NetUseAdd` method, we can control the process within code, without having to hack permissions or resort to DOS commands.

I could not find a way to do this in managed code. If you know of one, please post your solution. I would love to see it.