Printable Version

Using Network Functions in Visual C#.NET (Part II - Group Functions)

By Michael Bright

Welcome to the second installment of articles on how to use Platform Invoke to manage users and groups using Visual C#.NET. As you will have seen in Part 1, we discussed all the function relating to managing users on a local machine. In this article we shall look at how to Manage Groups on a local computer (i.e. Adding Groups, Deleting) and also how users interact with these groups (i.e. Adding/Removing users to groups). This article aims to flow from Part 1 and assume that Part 1 has been read. In this article we are going to look at the following functions

- NetLocalGroupAdd
- NetLocalGroupDel
- NetLocalGroupGetInfo
- NetLocalGroupSetInfo
- NetLocalGroupEnum
- NetLocalGroupGetMembers

Getting Started

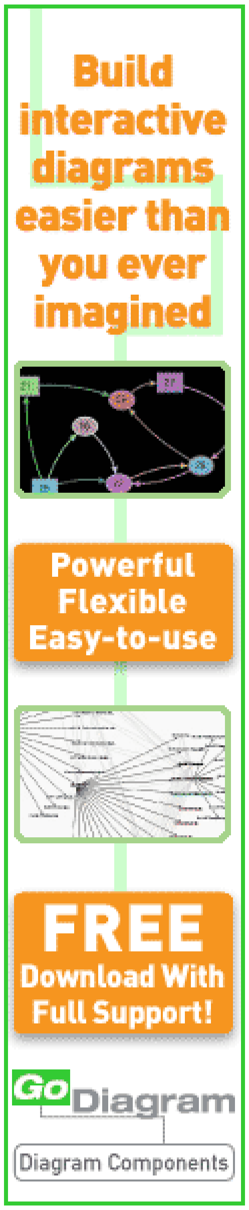
As we have discussed part one of this article, to use Platform invoke and the Network Function API's, we need to use InteropServices, which is declare in the below code snippet.

```
///// CODE SNIPPET 1.0

using System.Runtime.InteropServices;

//// END OF CODE
```

Once we have included the access we can now include the declarations of the DLL's we are going to use and the Structures (structs) associated with them. Each of the calls needed will be discussed under the headings releating to what they do.

Adding a Local Group using C#

Adding a local group in C# is just as easy as adding user to the local computer. And also just like the NetUserAdd function, it also requre a struct to hold the information about the group whihc it is going to add. In this case LOCALGROUP_INFO_1, and LOCALGROUP_INFO_0. Using one of these structs and the below declation we can successfully add a local group to the local computer or remote computer.


Build interactive diagrams easier than you ever imagined
Powerful Flexible Easy-to-use
FREE Download With Full Support!
GoDiagram
Diagram Components

Best sites

```
///// CODE SNIPPET 1.1 Declaration

[DllImport("Netapi32.dll")]
extern static int NetLocalGroupAdd([MarshalAs
(UnmanagedType.LPWStr)] string servername, int level, ref
LOCALGROUP_INFO_1 buf, int parm_err);

//// END OF CODE
```

Now we have declared the API we now must define the strucutre we have used in it.

```
///// CODE SNIPPET 1.2 Structure Declaration

[StructLayout(LayoutKind.Sequential,
CharSet=CharSet.Unicode)]
public struct LOCALGROUP_INFO_1
{
[MarshalAs(UnmanagedType.LPWStr)] public string
lgrpi1_name;
[MarshalAs(UnmanagedType.LPWStr)] public string
lgrpi1_comment;
}

//// END OF CODE
```

Once we have both of these block of code declare we can now use the call from within our program, i should aslo note that in this case we have used the strucutre about you can use any of the other three, and if you click the link you can convert the code from C++ to C# pretty easily. Below is a code snippet to show the usage of the NetLocalGroupAdd function.

```
///// CODE SNIPPET 1.3 NetLocalGroupAdd

LOCALGROUP_INFO_1 NewGroup = new LOCALGROUP_INFO_1(); //
Create an new instance of the LOCALGROUP_INFO_1 struct

NewGroup.lgrpi1_name = "TestGroupOne"; // Allocates the
Group Name
NewGroup.lgrpi1_comment = "My First Group Made through
C#"; // Comment on the New Group

if(NetLocalGroupAdd(null ,1 ,ref NewGroup, 0)!=0) // If
the call fails we get a non-zero value
{
MessageBox.Show("Error Adding
Group","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
}

//// END OF CODE
```

So as we discussed the above code will add a group to the machine you are currently on, but as I said if you want to add the group to a differant machine n you network you can replace the first null parameter of the call with the name of the machine.

Removing a Local Group using C#

Compared to the previous function removing a group is far easier, as with the above code the function returns a non-zero vaule if it fails, to remove a group from the local machine you can use the following declaration in the below code snippet.

```
///// CODE SNIPPET 1.4 Declaration

[DllImport("Netapi32.dll")]
extern static int NetLocalGroupDel([MarshalAs
(UnmanagedType.LPWStr)] stringservername,[MarshalAs
(UnmanagedType.LPWStr)] string groupname);

//// END OF CODE
```

The usage for the NetLocalGroupDel call would be, as is shown below in the code snippet.

```
///// CODE SNIPPET 1.5 NetUserDel

if(NetLocalGroupDel(null ,"TestGroupOne")!=0) // If the
call fails we get a non-zero value
{
MessageBox.Show("Error Removing
Group","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
}

//// END OF CODE
```

As also with the NetLocalGroupAdd call, the user can make this call to remote machines by replacing the null string in the first parameter wtih that of the unicode representtion or the computers name

Getting Group Information then modifying it using C#

To obtain group information within Visual C#.NET we need to used the native call NetLocalGroupGetInfo, this call as with NetLocalGorupAdd uses a strucutre (struct) to manage the data, however in this case it return data to a struct as opposed to taking it out. In assosciation with NetLocalGroupGetInfo , to change the information you have obtained you can use the function NetLocalGroupSetInfo . In the below code snippet we have the required declartions for both calls and please NOTE for this function we are using the LOCALGROUP_INFO_1 strucutre from above again.

```
///// CODE SNIPPET 1.6 Declarations

[DllImport("Netapi32.dll")]
extern static int NetLocalGroupGetInfo([MarshalAs
(UnmanagedType.LPWStr)] string servername,[MarshalAs
(UnmanagedType.LPWStr)] string groupname,int
level,outIntPtr bufptr);

[DllImport("Netapi32.dll")]
extern static int NetLocalGroupSetInfo([MarshalAs
(UnmanagedType.LPWStr)] string servername,[MarshalAs
(UnmanagedType.LPWStr)] string groupname,int level,ref
LOCALGROUP_INFO_1 buf,int parm_err);
```

//// END OF CODE

Using these declartions we can obtain and modify a users settings, with ease, if we look at the next code snippet we will obtain the user information for the user we made earlier "UserTestOne" and then we will change some user information.

///// CODE SNIPPET 1.7 NetLocalGroupGetInfo

```
IntPtr bufPtr;
LOCALGROUP_INFO_1 Group = new LOCALGROUP_INFO_1();
if(NetLocalGroupGetInfo(null, "TestGroupOne",1,out
bufPtr)!=0)
{
MessageBox.Show("Error Getting Group
Info","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
Group = (LOCALGROUP_INFO_1)Marshal.PtrToStructure(bufPtr,
typeof(LOCALGROUP_INFO_1));
MessageBox.Show("Group Name: " + Group.lgrpi1_name + "
Group Comments: " + Group.lgrpi1_comment);

// In this case we have used Marshaling to obtain the data
this is the only method i found that
// work but im sure someone can correct me on that?
```

//// END OF CODE

///// CODE SNIPPET 1.8 NetUsetSetInfo

```
LOCALGROUP_INFO_1 Update = new LOCALGROUP_INFO_1();
Update.lgrpi1_comment = "This is Our C# Updated Comment";
if(NetLocalGroupSetInfo(null, "TestGroupOne",1,ref
Update,0)!=0)
{
MessageBox.Show("Error Setting Group
Info","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
```

//// END OF CODE

Obtaining the list of Groups using C#

When managing a network it is also important what we have a acurate list of the groups available on the network, or local machine. To obtain this we must use the NetLocalGroupEnum function, which returns the names and data elements of each group to structures. In this case we are going to use the structure LOCALGROUP_INFO_1 to pass the return values into.. It is also important to note that this function uses the network buffer, which must be freed after to free resources, this is done using the NetAPIBufferFree function also listed. To start with in this code snippet is the declarations.

///// CODE SNIPPET 1.9 Declarations

```
[DllImport("Netapi32.dll")]
extern static int NetLocalGroupEnum([MarshalAs
(UnmanagedType.LPWStr)] string servername,int level,out
IntrPtr bufptr,int prefmaxlen,out int entriesread,out int
totalentries,out int resumehandle);

[DllImport("Netapi32.dll")]
extern static int NetApiBufferFree(IntPtr Buffer);

//// END OF CODE
```

Once we have made the declaration we can go out and use our code, to create the collection of the users, to do this we use the below code snippet.

```
///// CODE SNIPPET 2.0 NetLocalGroupEnum (Some code
attributed by Nick Holmes)

int EntriesRead;
int TotalEntries;
int Resume;
IntPtr bufPtr;

NetLocalGroupEnum(null, 1, 2, out bufPtr, -1, out
EntriesRead, out TotalEntries, out Resume);

if(EntriesRead> 0)
{
LOCALGROUP_INFO_1[] Groups = new LOCALGROUP_INFO_1
[EntriesRead];
IntPtr iter = bufPtr;
for(int i=0; i < EntriesRead; i++)
{
Groups[i] = (LOCALGROUP_INFO_1)Marshal.PtrToStructure
(iter, typeof(LOCALGROUP_INFO_1));
iter = (IntPtr)((int)iter + Marshal.SizeOf(typeof
(LOCALGROUP_INFO_1)));
MessageBox.Show(Groups[i].lgrpi1_name + " " + Group
[i].lgrpi1_comment);
}
NetworkAPI.NetApiBufferFree(bufPtr);
}

//// END OF CODE
```

The above code intern returns each use listed on the local machine via MessageBox, as with all other function you can chose a remote computer again by replacing the null string.

Identifying the members of a group using C#

The last function we are going to look at in this article is the NetLocalGroupGetMembers, this allows us to suggest a group and obtain a list of all the users which are members of that group. This can be done using the below declartion in the code snippet also note that we need to use the struct LOCALGROUP_MEMBERS_INFO_1.

```
///// CODE SNIPPET 2.1 Declarations

[StructLayout(LayoutKind.Sequential,
CharSet=CharSet.Unicode)]
public struct LOCALGROUP_MEMBERS_INFO_1
{
public int lgrmi1_sid;
public int lgrmi1_sidusage;
public string lgrmi1_name;
}

[DllImport("Netapi32.dll")]
extern static int NetLocalGroupGetMembers([MarshalAs
(UnmanagedType.LPWStr)] string servername,[MarshalAs
(UnmanagedType.LPWStr)] string localgroupname,int
level,out IntPtr bufptr,int prefmaxlen,out int
entriesread,out int totalentries,out int resumehandle);

[DllImport("Netapi32.dll")]
extern static int NetApiBufferFree(IntPtr Buffer);

//// END OF CODE
```

Once we have made these declaration the usage for it would be.

```
///// CODE SNIPPET 2.4 NetUserGetLocalGroups

int EntriesRead;
int TotalEntries;
int Resume;
IntPtr bufPtr;

NetworkAPI.NetLocalGroupGetMembers(null,
this.cmbGroups.Text, 1, out bufPtr, -1, out EntriesRead,
out TotalEntries, out Resume);

if(EntriesRead> 0)
{
NetworkAPI.LOCALGROUP_MEMBERS_INFO_1[] Members = new
NetworkAPI.LOCALGROUP_MEMBERS_INFO_1[EntriesRead];
IntPtr iter = bufPtr;
for(int i=0; i < EntriesRead; i++)
{
Members[i] = (NetworkAPI.LOCALGROUP_MEMBERS_INFO_1)
Marshal.PtrToStructure(iter, typeof
(NetworkAPI.LOCALGROUP_MEMBERS_INFO_1));
iter = (IntPtr)((int)iter + Marshal.SizeOf(typeof
(NetworkAPI.LOCALGROUP_MEMBERS_INFO_1)));
MessageBox.Show(Members[i].lgrmi1_name);
}
NetworkAPI.NetApiBufferFree(bufPtr);
}

//// END OF CODE
```

The above example returns the groups to which the user Adminstrator belongs, again you can specify any user and any computer name. In this article we has looked at and discussed the User related Network Function available through Platform Invoke on the .NET Framework. So you can see that all this code does infact work, i have included a samply Example application with all the function working in it. Available for download below.

Network Function Examples ~
Visual Studio.NET Project + Source (49K)
Runtime EXE (28K)

About the Author

My name is Michael Bright, I'm a university student studying a Bsc in Computer Science at University College Chester. I have a love for all thing computers and have only worked with C# for about 3 months. I am currently training for my MCP in it. My other interests are C, VB, HTML, ASP and also Flash. I have interests in Developing Network app's and Network Security. My Web site is csharp.brightweb.co.uk where you can find examples of my Work, including my Defender Security applications.

Saturday 25 January, 2003 2:52 PM

Finally, a profiler that's even quicker to find fault than your boss.