

```

using System;
using System.Runtime.InteropServices;
using System.Security;
using System.Collections;
using System.Windows.Forms;

namespace sb54_CSAI.ClientApp
{
    #region NetworkBrowser CLASS
    /// <summary>
    /// Provides a mechanism for supplying a list of all PC names in the local network.
    /// This collection of PC names is used in the <see cref="sb54_CSAI.ClientApp.
    FrmClientLoader">
    /// ReMP3 client loader form </see>to allow the user to pick the PC that is running the
    /// ReMP3 server application.
    /// <br></br>
    /// This class makes use of a DllImport instruction. The pupose of which is as follows:
    /// When a DllImport declaration is made in managed code (C#) it is a call to a legacy
    /// unmanaged code module, normally a C++ Dynamic Link Library. These C++ Dll's are
    /// usually part of the operating system API, or some other vendors API, and must be
    /// used to carry out operations that are not native within the managed code API. This
    is
    /// fairly normal within the windows world. The only thing that needs careful
    consideration
    /// is the construction of the correct type of STRUCTS, object pointers, and attribute
    markers,
    /// which all contribute to making the link between managed (C#) and unmanaged code (C+
    +)
    /// more seamless
    /// <br></br>
    /// This class makes use of the following Dll calls
    /// <list type="bullet">
    /// <item>
    /// <description> Netapi32.dll : NetServerEnum, The NetServerEnum function lists all
    servers
    /// of the specified type that are visible in a domain. For example, an application can
    call
    /// NetServerEnum to list all domain controllers only or all SQL servers only.
    /// You can combine bit masks to list several types. For example, a value of 0x00000003
    /// combines the bit masks for SV_TYPE_WORKSTATION (0x00000001) and SV_TYPE_SERVER
    (0x00000002).
    /// </description>
    /// </item>
    /// <item>
    /// <description> Netapi32.dll : NetApiBufferFree, The NetApiBufferFree function frees
    /// the memory that the NetApiBufferAllocate function allocates. Call NetApiBufferFree
    /// to free the memory that other network management functions return.</description>
    /// </item>
    /// </list>
    /// </summary>
    public sealed class NetworkBrowser
    {
        #region Dll Imports

        //declare the Netapi32 : NetServerEnum method import
        [DllImport("Netapi32", CharSet=CharSet.Auto, SetLastError=true),
        SuppressUnmanagedCodeSecurityAttribute]

        /// <summary>
        /// Netapi32.dll : The NetServerEnum function lists all servers
        /// of the specified type that are visible in a domain. For example, an application
    can call
    /// NetServerEnum to list all domain controllers only or all SQL servers only.
    /// You can combine bit masks to list several types. For example, a value of
    0x00000003
    /// combines the bit masks for SV_TYPE_WORKSTATION (0x00000001) and SV_TYPE_SERVER
    (0x00000002)
    /// </summary>
        public static extern int NetServerEnum(
            string ServerName, // must be null
            int dwLevel,
            ref IntPtr pBuf,
            int dwPrefMaxLen,

```

```

    out int dwEntriesRead,
    out int dwTotalEntries,
    int dwServerType,
    string domain, // null for login domain
    out int dwResumeHandle
    );

//declare the Netapi32 : NetApiBufferFree method import
[DllImport("Netapi32", SetLastError=true),
SuppressUnmanagedCodeSecurityAttribute]

/// <summary>
/// Netapi32.dll : The NetApiBufferFree function frees
/// the memory that the NetApiBufferAllocate function allocates. Call
NetApiBufferFree
/// to free the memory that other network management functions return.
/// </summary>
public static extern int NetApiBufferFree(
    IntPtr pBuf);

//create a _SERVER_INFO_100 STRUCTURE
[StructLayout(LayoutKind.Sequential)]
public struct _SERVER_INFO_100
{
    internal int sv100_platform_id;
    [MarshalAs(UnmanagedType.LPWSTR)]
    internal string sv100_name;
}
#endregion
#region Public Constructor
/// <summary>
/// Constructor, simply creates a new NetworkBrowser object
/// </summary>
public NetworkBrowser()
{
}
#endregion
#region Public Methods
/// <summary>
/// Uses the DllImport : NetServerEnum with all its required parameters
/// (see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netmgmt/
netmgmt/netserverenum.asp
/// for full details or method signature) to retrieve a list of domain
SV_TYPE_WORKSTATION
/// and SV_TYPE_SERVER PC's
/// </summary>
/// <returns>Arraylist that represents all the SV_TYPE_WORKSTATION and
SV_TYPE_SERVER
/// PC's in the Domain</returns>
public ArrayList getNetworkComputers()
{
    //local fields
    ArrayList networkComputers= new ArrayList();
    const int MAX_PREFERRED_LENGTH = -1;
    int SV_TYPE_WORKSTATION = 1;
    int SV_TYPE_SERVER = 2;
    IntPtr buffer = IntPtr.Zero;
    IntPtr tmpBuffer = IntPtr.Zero;
    int entriesRead = 0;
    int totalEntries = 0;
    int resHandle = 0;
    int sizeofINFO = Marshal.SizeOf(typeof(_SERVER_INFO_100));

    try
    {
        //call the DllImport : NetServerEnum with all its required parameters
        //see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/
netmgmt/netmgmt/netserverenum.asp
        //for full details of method signature
        int ret = NetServerEnum(null, 100, ref buffer, MAX_PREFERRED_LENGTH,
            out entriesRead,

```

```

        out totalEntries, SV_TYPE_WORKSTATION|SV_TYPE_SERVER, null, out
        resHandle);
//if the returned with a NERR_Success (C++ term), =0 for C#
if (ret == 0)
{
    //loop through all SV_TYPE_WORKSTATION and SV_TYPE_SERVER PC's
    for (int i = 0; i < totalEntries ; i++)
    {
        //get pointer to, Pointer to the buffer that received the data from
        //the call to NetServerEnum. Must ensure to use correct size of
        //STRUCTURE to ensure correct location in memory is pointed to
        tmpBuffer = new IntPtr((int)buffer + (i * sizeofINFO));
        //Have now got a pointer to the list of SV_TYPE_WORKSTATION and
        //SV_TYPE_SERVER PC's, which is unmanaged memory
        //Needs to Marshal data from an unmanaged block of memory to a
        //managed object, again using STRUCTURE to ensure the correct data
        //is marshalled
        _SERVER_INFO_100 svrInfo = (_SERVER_INFO_100)
            Marshal.PtrToStructure(tmpBuffer, typeof(_SERVER_INFO_100));

        //add the PC names to the ArrayList
        networkComputers.Add(svrInfo.sv100_name);
    }
}
}
catch (Exception ex)
{
    MessageBox.Show("Problem with accessing network computers in NetworkBrowser " +
        "\r\n\r\n\r\n\r\n" + ex.Message,
        "Form Loading ERROR", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    return null;
}
finally
{
    //The NetApiBufferFree function frees
    //the memory that the NetApiBufferAllocate function allocates
    NetApiBufferFree(buffer);
}
//return entries found
return networkComputers;
}
}
#endregion
}
#endregion
}

```