

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Management;
using System.Net;
using BS.Utilities;
using System.Collections;

namespace SB54_CSAI.TrackObjects
{
    #region TrackResolver CLASS
    /// <summary>
    /// This class provides some general methods to check whether a file is online.
    /// The file is 1st checked to see if it is a network file. if the file is a network
    file
    /// a Ping to the remote PC is performed. If the PC is online, then the file is deemed
    to be
    /// available.
    /// <br></br>
    /// <br></br>
    /// This class was written and using an article contained at the following URL
    /// http://www.codeproject.com/script/profile/whos\_who.asp?id=757023, where the original
    /// author was Wesley Brown
    /// </summary>
    public class TrackResolver
    {
        #region Instance Fields
        //Instance Fields
        private BS.Utilities.Ping netMon = new BS.Utilities.Ping();
        private ArrayList localDrives = new ArrayList();

        #endregion
        #region Public Constructor
        /// <summary>
        /// Simply constructs a new TrackResolver object
        /// </summary>
        public TrackResolver ()
        {
            //get all local drives
            getLocalDrives();
        }
        #endregion
        #region Public Methods

        /// <summary>
        /// Checks to see if the file is local, or remote PC file. If the file
        /// is local simply return true, if the file is remote try and ping
        /// the remote PC, if it is available return true, else false
        /// </summary>
        /// <param name="file">The full name of the file to check</param>
        /// <returns>true if this file is available</returns>
        public bool checkFileIsOnline(string file)
        {
            string getHostAtStart=string.Empty;
            string hostName=string.Empty;

            //see if the file is on another PC
            if (file.StartsWith(@"\\\"))
            {
                //strip the string to get hostname
                getHostAtStart = @file.Substring(2);
                int firstSlash = getHostAtStart.IndexOf(@"\\");
                hostName = getHostAtStart.Substring(0,firstSlash);
                //see if the host is available
                if (pingRemotePC(hostName))
                {
                    return true;
                }
            }
            else
            {

```

```

        return false;
    }
    }
    else
    {
        return true;
    }
}
#endregion
#region Private Methods

/// <summary>
/// Takes a file parameter and checks to see if this file is on a local hard disk,
and if
/// it is. The computer name string is appended to the file name and then returned.
/// If the file is not local the file name is returned un-modified
/// </summary>
/// <param name="file">the file to check</param>
/// <returns>the new file name</returns>
public string getFullFilePath(string file)
{
    //get the selected file
    FileInfo f = new FileInfo (@file);

    //If the folder is not local, simply return file string
    if (! localDrives.Contains(f.FullName.Substring(0,2)))
    {
        return file;
    }
    else
    {
        //otherwise create a full UNC path
        int ColonIdx = file.IndexOf(":",0);
        file = @file.Substring(0,ColonIdx) + @"$" + @file.Substring(ColonIdx+1);
        return @"\\\" + Dns.GetHostName() + @"\\" + file;
    }
}

/// <summary>
/// Gets a list of all local drives using a ManagementObjectSearcher object
/// combined with a SELECT * From Win32_LogicalDisk query
/// </summary>
private void getLocalDrives()
{
    //Local disk type drive = 3
    const int LocalDisk = 3;
    //get a list of all drives on users system, then filter out Local Drives and add
    //them to the localDrives collection
    ManagementObjectSearcher query = new ManagementObjectSearcher("SELECT * From
Win32_LogicalDisk ");
    ManagementObjectCollection queryCollection = query.Get();
    foreach ( ManagementObject mo in queryCollection)
    {
        int diskType = int.Parse(mo["DriveType"].ToString());
        if (diskType == LocalDisk)
        {
            localDrives.Add(mo["Name"].ToString());
        }
    }
}

/// <summary>
/// Pings the hostname provided
/// </summary>

```

```
/// <param name="hostname">hostname to ping</param>
/// <returns>true if hostname is available</returns>
private bool pingRemotePC(string hostname)
{
    //see if there is a host name
    if (hostname.Equals(string.Empty))
        return false;

    //get back the PingResponse for this host, try ping 2 times
    PingResponse response = netMon.PingHost(hostname,2);

    //if null, not good
    if (response == null)
    {
        return false;
    }
    else
    {
        //process response
        return ProcessResponse(response);
    }
}

/// <summary>
/// Processes the response, returning true if the response ==
/// PingResponseType.Ok
/// </summary>
/// <param name="response">PingResponse to process</param>
/// <returns>true if response == PingResponseType.Ok</returns>
private bool ProcessResponse(PingResponse response)
{
    //simply find out what sort of response we got
    switch (response.PingResult)
    {
        case PingResponseType.Ok:
            return true;
            break;
        case PingResponseType.RequestTimedOut:
            return false;
            break;
        case PingResponseType.InternalError:
            return false;
            break;
        case PingResponseType.ConnectionError:
            return false;
            break;
        case PingResponseType.CouldNotResolveHost:
            return false;
            break;
        default:
            return false;
            break;
    }
}
#endregion
}
#endregion
}
```