

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Data;
using System.IO;
using System.Windows.Forms;
using WMPLib;
using sb54_CSAI.MiscFormComponents;
using SB54_CSAI.TrackObjects;
using BS.Utilities;

namespace SB54_CSAI.MediaPlayerControl
{
    #region uctMediaPlayer CLASS

    /// <summary>
    /// uctMediaPlayer is a sub class of a <see cref="System.Windows.Forms.UserControl">
    UserControl </see>
    /// that provides an embedded WindowsMediaPlayer control that is maybe controlled both
    locally
    /// using the controls provided OR may be controlled via the <see cref="sb54_CSAI.
    ClientApp.FrmClient">ReMP3 Client. </see>
    /// This user control also provides a track list that shows the currently selected
    tracks and the order
    /// in which the will be played.
    /// <listheader>The tracklist may be constructed using any of the following methods</
    listheader>
    /// <list type="bullet">
    /// <item><description>Using drag and drop from your local file system</description></
    item>
    /// <item><description>Using the add button provided to add a single file or an entire
    /// directory</description></item>
    /// <item><description>Using the Media Library to add file(s)</description></item>
    /// </list>
    /// <br></br>
    /// <br></br>
    /// Additionally the tracklist may be saved/loaded to/from XML files
    /// </summary>
    public class uctMediaPlayer : System.Windows.Forms.UserControl
    {
        #region Instance fields
        //Instance fields
        private AxWMPLib.AxWindowsMediaPlayer oMMPPlayer;
        private System.Windows.Forms.Splitter oSplitlv;
        private System.Windows.Forms.Panel oPnlMainL;
        private System.Windows.Forms.Panel oPnlFilesBottom;
        private System.Windows.Forms.Panel oPnlFilesTop;
        private System.Windows.Forms.Label oLblFilesTop;
        private System.Windows.Forms.ToolTip toolTip1;
        private System.Windows.Forms.ImageList oImgMmGui;
        private System.Windows.Forms.ComboBox oCmbMmGui;
        private System.ComponentModel.IContainer components;
        private System.Windows.Forms.Button oBtnUp;
        private System.Windows.Forms.Button oBtnDown;
        private System.Windows.Forms.ImageList oImgButtons;
        private System.Windows.Forms.FolderBrowserDialog oFldrBrowser1;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.ImageList oImgTrackItems;
        private System.Windows.Forms.OpenFileDialog oFileBrowser1;
        private System.Windows.Forms.ListBox oLstFiles;
        private System.Windows.Forms.Panel oPnlFilesFill;
        private System.Windows.Forms.ContextMenu oContMen1;
        private System.Windows.Forms.MenuItem oMnuAddFolder;
        private System.Windows.Forms.MenuItem oMnuAddFiles;
        private System.Windows.Forms.Button oBtnAdd;
        private System.Windows.Forms.Button oBtnClear;
        private frmTrack oFrmtrack = new frmTrack();
        private bool ChangingTrackOrder = false;
        private ArrayList supportedFiles = new ArrayList();
        private int CurrentTrack=-1;
        private string hoveredItem="";
```

```
private bool allowPopup =false;
private int topSelectedTrackItem;
private sb54_CSAI.MiscFormComponents.MenuIcons oMnuIcons;
private System.Windows.Forms.Button oBtnSaveTracklist;
private System.Windows.Forms.Button oBtnLoadTracklist;
private System.Windows.Forms.SaveFileDialog oSaveFileBrowser1;
private int bottomSelectedTrackItem;
private bool allowFF=false;
private TrackResolver trResolve = new TrackResolver();
/// <summary>
/// ScanningFolderhandler delegate for the Scan event
/// </summary>
public delegate void CurrentMediaTrackhandler(object sender,
CurrentMediaTrackEventArgs e);
/// <summary>
/// Raised during the scanning of files. Each time a new directory is scanned
/// a new event is raised with the directory name available via the
/// ScanningEventArgs</see>
/// </summary>
public event CurrentMediaTrackhandler currentMediaTrack;

#endregion
#region Public Constructor
/// <summary>
/// Constructor, simply initalises all GUI components and setup all
/// the supported files using the setUpSupportedFiles() method
/// </summary>
public uctMediaPlayer()
{
    // This call is required by the Windows.Forms Form Designer.
    InitializeComponent();

    //setup supported files ArrayList
    setUpSupportedFiles();

    //ofrmTrack popup timer, user must hover over a track
    //for timer1 interval before track description popup window
    //is shown
    timer1.Start();
}
#endregion
#region Public Methods

/// <summary>
/// Raises the OnCurrentMediaTrack event for the MediaPlayerControl. Any users of
the
/// MediaPlayerControl may now subscribe to this event using a
CurrentMediaTrackhandler
/// delegate.
/// </summary>
/// <param name="e">The <see cref="SB54_CSAI.MediaPlayerControl.
CurrentMediaTrackEventArgs"> CurrentMediaTrack event arguments</see></param>
public virtual void OnCurrentMediaTrack(CurrentMediaTrackEventArgs e)
{
    if (currentMediaTrack != null)
    {
        // Invokes the delegates.
        currentMediaTrack(this, e);
    }
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
/// <param name="disposing">>true indicates the object should be disposed</param>
protected override void Dispose( bool disposing )
{
    //if being asked to dispose
    if( disposing )
    {
        //clean up the held resources
```

```
        if( components != null ) { components.Dispose(); }
        if (oFrmtrack != null) { oFrmtrack.Dispose(); }
    }
    //call base class Dispose
    base.Dispose( disposing );
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
/// to inform (by using .NET Remoting) the ReMP3 server (the one that hosts
/// this control) to perform a FastForward (by 1 track) operation
/// </summary>
public void cmdFF()
{
    //if track order NOT being changed
    if (! ChangingTrackOrder)
    {
        //if there is a track selected, perform the FF operation
        if (oLstFiles.SelectedIndex != oLstFiles.Items.Count -1 && oLstFiles.
SelectedIndex < oLstFiles.Items.Count)
        {
            oLstFiles.SelectedIndex = -1;
            CurrentTrack++;
            oLstFiles.SelectedIndex = CurrentTrack;
            redrawTrackItems();
            getNewTrack();
            redrawTrackItems();
        }
    }
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
/// to inform (by using .NET Remoting) the ReMP3 server (the one that hosts
/// this control) to perform a Rewind (by 1 track) operation
/// </summary>
public void cmdREW()
{
    //if track order NOT being changed
    if (! ChangingTrackOrder)
    {
        //if there is a track selected, perform the REW operation
        if (oLstFiles.SelectedIndex > 0)
        {
            oLstFiles.SelectedIndex = -1;
            CurrentTrack--;
            oLstFiles.SelectedIndex = CurrentTrack;
            redrawTrackItems();
            getNewTrack();
            redrawTrackItems();
        }
    }
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
/// to inform (by using .NET Remoting) the ReMP3 server (the one that hosts
/// this control) to clear the current track list of all tracks
/// </summary>
public void cmdCLEAR()
{
    CurrentTrack = -1;
    try
    {
        oLstFiles.SelectedIndex = -1;
        oLstFiles.Items.Clear();
    }
    catch (Exception ex)
    {
    }
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
```

```
/// to inform (by using .NET Remoting) the ReMP3 server (the one that hosts
/// this control) to stop the player playing
/// </summary>
public void cmdSTOP()
{
    oMMPPlayer.Ctlcontrols.stop();
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
/// to get a list of tracks that are present on the Media Player, too keep the
/// view of the Client and the Server synchronized
/// </summary>
public string[] getTrackList()
{
    string[] playerFiles = new string[oLstFiles.Items.Count];
    for (int i=0; i < oLstFiles.Items.Count; i++)
    {
        playerFiles[i] = ((MP3ListItem)oLstFiles.Items[i]).URL;
    }
    return playerFiles;
}

/// <summary>
/// Used by the <see cref="sb54_CSAI.ClientApp.FrmClient">ReMP3 Client </see>
/// to inform (by using .NET Remoting) the ReMP3 server (the one that hosts
/// this control) to start the player to play
/// </summary>
public void cmdPLAY()
{
    //if track order NOT being changed
    if (! ChangingTrackOrder)
    {
        //if there is currently no track selected
        if (oLstFiles.SelectedIndex == -1)
        {
            //and there are some tracks here to play
            if (oLstFiles.Items.Count > 0)
            {
                //select the 1st track
                CurrentTrack =0;
                oLstFiles.SelectedIndex = CurrentTrack;
            }
        }
        //get the track
        getNewTrack();
    }
}

/// <summary>
/// Adds file(s) to the track list. If nested directories names
/// are included within the input array, recursive calls will be
/// made to ensure that all files and directories are processed
/// <br></br>
/// Only files that are available are added. So if the files within a
/// saved XML file point to remote files, and the remote PC is not
/// available via a ping, then the files that are unavailable will
/// not be loaded
/// </summary>
/// <param name="files">An array of strings that represent the files
/// or folders that should be added to the track list</param>
public void addFiles(String[] files)
{
    try
    {
        // loop through the string array, adding each filename to the ListBox
        foreach( string file in files )
        {
            System.IO.DirectoryInfo d = new System.IO.DirectoryInfo(file);
```

```

        //is the item actually a directory
        if (d.Attributes == FileAttributes.Directory)
        {
            //if it is, need to see if it contains any valid files
            FileInfo[] fi = d.GetFiles();
            DirectoryInfo[] di = d.GetDirectories();

            ArrayList arDirFiles = new ArrayList();

            //loop through getting all files and directories for the added
            folder
            for (int i=0;i< fi.Length; i++ )
            {
                arDirFiles.Add(fi[i].FullName);
            }

            for (int i=0;i< di.Length; i++ )
            {
                arDirFiles.Add(di[i].FullName);
            }

            //convert arraylist to string array
            String[]
            string[] dirFiles = arDirFiles.ToArray(typeof(System.String)) as
            directory
            //make recursive call based on new files retrieved for this
            addFiles(dirFiles);
        }
        //sanity check, see if valid file type
        if (d.Extension.Length !=0 && isSupportedFiled(d.Extension))
        {
            if (trResolve.checkFileIsOnline(file))
            {
                string fs = trResolve.getFullFilePath(file);
                oLstFiles.Items.Add( new MP3ListItem(fs));
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Problem with MediaPlayerControl " +
        "\r\n\r\n\r\n" + ex.Message,
        "Adding Files", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    return;
}
}
#endregion
#region Component Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    System.Resources.ResourceManager resources = new System.Resources.
ResourceManager(typeof(uctMediaPlayer));
    this.oSplitlv = new System.Windows.Forms.Splitter();
    this.oPnlMainL = new System.Windows.Forms.Panel();
    this.oMMPPlayer = new AxWMPLib.AxWindowsMediaPlayer();
    this.oPnlFilesBottom = new System.Windows.Forms.Panel();
    this.oBtnLoadTracklist = new System.Windows.Forms.Button();
    this.oBtnSaveTracklist = new System.Windows.Forms.Button();
    this.oBtnClear = new System.Windows.Forms.Button();
    this.oCmbMmGui = new System.Windows.Forms.ComboBox();
    this.oBtnUp = new System.Windows.Forms.Button();
    this.oBtnDown = new System.Windows.Forms.Button();
    this.oBtnAdd = new System.Windows.Forms.Button();
}

```

```

this.oImgButtons = new System.Windows.Forms.ImageList(this.components);
this.oPnlFilesTop = new System.Windows.Forms.Panel();
this.oLblFilesTop = new System.Windows.Forms.Label();
this.toolTip1 = new System.Windows.Forms.ToolTip(this.components);
this.oLstFiles = new System.Windows.Forms.ListBox();
this.oImgMmGui = new System.Windows.Forms.ImageList(this.components);
this.oFldrBrowser1 = new System.Windows.Forms.FolderBrowserDialog();
this.timer1 = new System.Windows.Forms.Timer(this.components);
this.oImgTrackItems = new System.Windows.Forms.ImageList(this.components);
this.oFileBrowser1 = new System.Windows.Forms.OpenFileDialog();
this.oPnlFilesFill = new System.Windows.Forms.Panel();
this.oContMen1 = new System.Windows.Forms.ContextMenu();
this.oMnuAddFolder = new System.Windows.Forms.MenuItem();
this.oMnuAddFiles = new System.Windows.Forms.MenuItem();
this.oMnuIcons = new sb54_CSAI.MiscFormComponents.MenuIcons(this.components);
this.oSaveFileBrowser1 = new System.Windows.Forms.SaveFileDialog();
this.oPnlMainL.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.oMMPlayer)).BeginInit();
this.oPnlFilesBottom.SuspendLayout();
this.oPnlFilesTop.SuspendLayout();
this.oPnlFilesFill.SuspendLayout();
this.SuspendLayout();
//
// oSplitlv
//
this.oSplitlv.BackColor = System.Drawing.Color.Black;
this.oSplitlv.Location = new System.Drawing.Point(560, 0);
this.oSplitlv.Name = "oSplitlv";
this.oSplitlv.Size = new System.Drawing.Size(8, 688);
this.oSplitlv.TabIndex = 2;
this.oSplitlv.TabStop = false;
//
// oPnlMainL
//
this.oPnlMainL.AutoScroll = true;
this.oPnlMainL.AutoScrollMinSize = new System.Drawing.Size(350, 392);
this.oPnlMainL.Controls.Add(this.oMMPlayer);
this.oPnlMainL.Dock = System.Windows.Forms.DockStyle.Left;
this.oPnlMainL.Location = new System.Drawing.Point(0, 0);
this.oPnlMainL.Name = "oPnlMainL";
this.oPnlMainL.Size = new System.Drawing.Size(560, 688);
this.oPnlMainL.TabIndex = 1;
//
// oMMPlayer
//
this.oMMPlayer.ContainingControl = this;
this.oMMPlayer.Dock = System.Windows.Forms.DockStyle.Fill;
this.oMMPlayer.Enabled = true;
this.oMMPlayer.Location = new System.Drawing.Point(0, 0);
this.oMMPlayer.Name = "oMMPlayer";
this.oMMPlayer.OcxState = ((System.Windows.Forms.AxHost.State)(resources.
GetObject("oMMPlayer.OcxState")));
this.oMMPlayer.Size = new System.Drawing.Size(560, 688);
this.oMMPlayer.TabIndex = 0;
this.toolTip1.SetToolTip(this.oMMPlayer, "Media player");
this.oMMPlayer.PlayStateChange += new AxWMPLib.
_WMPOCXEvents_PlayStateChangeEventHandler(this.oMMPlayer_PlayStateChange);
this.oMMPlayer.OpenStateChange += new AxWMPLib.
_WMPOCXEvents_OpenStateChangeEventHandler(this.oMMPlayer_OpenStateChange);
this.oMMPlayer.MediaError += new AxWMPLib._WMPOCXEvents_MediaErrorEventHandler
(this.oMMPlayer_MediaError);
this.oMMPlayer.MediaChange += new AxWMPLib._WMPOCXEvents_MediaChangeEventHandler
(this.oMMPlayer_MediaChange);
//
// oPnlFilesBottom
//
this.oPnlFilesBottom.AutoScroll = true;
this.oPnlFilesBottom.AutoScrollMinSize = new System.Drawing.Size(240, 40);
this.oPnlFilesBottom.BackColor = System.Drawing.Color.White;
this.oPnlFilesBottom.Controls.Add(this.oBtnLoadTracklist);
this.oPnlFilesBottom.Controls.Add(this.oBtnSaveTracklist);
this.oPnlFilesBottom.Controls.Add(this.oBtnClear);
this.oPnlFilesBottom.Controls.Add(this.oCmbMmGui);

```

```
this.oPnlFilesBottom.Controls.Add(this.oBtnUp);
this.oPnlFilesBottom.Controls.Add(this.oBtnDown);
this.oPnlFilesBottom.Controls.Add(this.oBtnAdd);
this.oPnlFilesBottom.Dock = System.Windows.Forms.DockStyle.Bottom;
this.oPnlFilesBottom.Location = new System.Drawing.Point(568, 640);
this.oPnlFilesBottom.Name = "oPnlFilesBottom";
this.oPnlFilesBottom.Size = new System.Drawing.Size(440, 48);
this.oPnlFilesBottom.TabIndex = 3;
//
// oBtnLoadTracklist
//
this.oBtnLoadTracklist.BackColor = System.Drawing.Color.White;
this.oBtnLoadTracklist.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.oBtnLoadTracklist.ForeColor = System.Drawing.Color.Black;
this.oBtnLoadTracklist.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnLoadTracklist.Image")));
this.oBtnLoadTracklist.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.oBtnLoadTracklist.Location = new System.Drawing.Point(256, 8);
this.oBtnLoadTracklist.Name = "oBtnLoadTracklist";
this.oBtnLoadTracklist.Size = new System.Drawing.Size(64, 24);
this.oBtnLoadTracklist.TabIndex = 14;
this.oBtnLoadTracklist.Text = "Load";
this.oBtnLoadTracklist.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.toolTip1.SetToolTip(this.oBtnLoadTracklist, "Open a saved tracklist");
this.oBtnLoadTracklist.Click += new System.EventHandler(this.
oBtnLoadTracklist_Click);
//
// oBtnSaveTracklist
//
this.oBtnSaveTracklist.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.oBtnSaveTracklist.ForeColor = System.Drawing.Color.Black;
this.oBtnSaveTracklist.Image = ((System.Drawing.Image)(resources.GetObject(
"oBtnSaveTracklist.Image")));
this.oBtnSaveTracklist.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.oBtnSaveTracklist.Location = new System.Drawing.Point(184, 8);
this.oBtnSaveTracklist.Name = "oBtnSaveTracklist";
this.oBtnSaveTracklist.Size = new System.Drawing.Size(64, 24);
this.oBtnSaveTracklist.TabIndex = 13;
this.oBtnSaveTracklist.Text = "Save";
this.oBtnSaveTracklist.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.toolTip1.SetToolTip(this.oBtnSaveTracklist, "Save a new tracklist");
this.oBtnSaveTracklist.Click += new System.EventHandler(this.
oBtnSaveTracklist_Click);
//
// oBtnClear
//
this.oBtnClear.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.oBtnClear.ForeColor = System.Drawing.Color.White;
this.oBtnClear.Image = ((System.Drawing.Image)(resources.GetObject("oBtnClear.
Image")));
this.oBtnClear.Location = new System.Drawing.Point(40, 8);
this.oBtnClear.Name = "oBtnClear";
this.oBtnClear.Size = new System.Drawing.Size(24, 24);
this.oBtnClear.TabIndex = 12;
this.toolTip1.SetToolTip(this.oBtnClear, "Clear the tracklist");
this.oBtnClear.Click += new System.EventHandler(this.oBtnClear_Click);
//
// oCmbMmGui
//
this.oCmbMmGui.DrawMode = System.Windows.Forms.DrawMode.OwnerDrawFixed;
this.oCmbMmGui.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this.oCmbMmGui.ItemHeight = 20;
this.oCmbMmGui.Items.AddRange(new object[] {
"none",
"mini",
"full"});
this.oCmbMmGui.Location = new System.Drawing.Point(80, 8);
this.oCmbMmGui.MaxDropDownItems = 3;
this.oCmbMmGui.Name = "oCmbMmGui";
this.oCmbMmGui.Size = new System.Drawing.Size(96, 26);
this.oCmbMmGui.TabIndex = 9;
this.toolTip1.SetToolTip(this.oCmbMmGui, "Change the media player control GUI");
this.oCmbMmGui.SelectedIndexChanged += new System.EventHandler(this.
```

```
oCmbMmGui_SelectedIndexChanged);
    this.oCmbMmGui.DrawItem += new System.Windows.Forms.DrawItemEventHandler(this.
oCmbMmGui_DrawItem);
    //
    // oBtnUp
    //
    this.oBtnUp.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnUp.ForeColor = System.Drawing.Color.White;
    this.oBtnUp.Image = ((System.Drawing.Image)(resources.GetObject("oBtnUp.Image"
)));
    this.oBtnUp.Location = new System.Drawing.Point(328, 6);
    this.oBtnUp.Name = "oBtnUp";
    this.oBtnUp.Size = new System.Drawing.Size(26, 26);
    this.oBtnUp.TabIndex = 1;
    this.toolTip1.SetToolTip(this.oBtnUp, "Move Selected Track(s) Up");
    this.oBtnUp.Click += new System.EventHandler(this.oBtnUp_Click);
    //
    // oBtnDown
    //
    this.oBtnDown.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnDown.ForeColor = System.Drawing.Color.White;
    this.oBtnDown.Image = ((System.Drawing.Image)(resources.GetObject("oBtnDown.
Image")));
    this.oBtnDown.Location = new System.Drawing.Point(356, 6);
    this.oBtnDown.Name = "oBtnDown";
    this.oBtnDown.Size = new System.Drawing.Size(26, 26);
    this.oBtnDown.TabIndex = 2;
    this.toolTip1.SetToolTip(this.oBtnDown, "Move Selected Track(s) Down ");
    this.oBtnDown.Click += new System.EventHandler(this.oBtnDown_Click);
    //
    // oBtnAdd
    //
    this.oBtnAdd.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
    this.oBtnAdd.ForeColor = System.Drawing.Color.White;
    this.oBtnAdd.Image = ((System.Drawing.Image)(resources.GetObject("oBtnAdd.Image"
)));
    this.oBtnAdd.Location = new System.Drawing.Point(8, 8);
    this.oBtnAdd.Name = "oBtnAdd";
    this.oBtnAdd.Size = new System.Drawing.Size(24, 24);
    this.oBtnAdd.TabIndex = 11;
    this.toolTip1.SetToolTip(this.oBtnAdd, "Add files to the track list");
    this.oBtnAdd.Click += new System.EventHandler(this.oBtnAdd_Click);
    //
    // oImgButtons
    //
    this.oImgButtons.ColorDepth = System.Windows.Forms.ColorDepth.Depth32Bit;
    this.oImgButtons.ImageSize = new System.Drawing.Size(17, 17);
    this.oImgButtons.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgButtons.ImageStream")));
    this.oImgButtons.TransparentColor = System.Drawing.Color.Transparent;
    //
    // oPnlFilesTop
    //
    this.oPnlFilesTop.Controls.Add(this.oLblFilesTop);
    this.oPnlFilesTop.Dock = System.Windows.Forms.DockStyle.Top;
    this.oPnlFilesTop.Location = new System.Drawing.Point(568, 0);
    this.oPnlFilesTop.Name = "oPnlFilesTop";
    this.oPnlFilesTop.Size = new System.Drawing.Size(440, 16);
    this.oPnlFilesTop.TabIndex = 4;
    //
    // oLblFilesTop
    //
    this.oLblFilesTop.BackColor = System.Drawing.Color.Black;
    this.oLblFilesTop.Dock = System.Windows.Forms.DockStyle.Fill;
    this.oLblFilesTop.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.oLblFilesTop.ForeColor = System.Drawing.Color.White;
    this.oLblFilesTop.Location = new System.Drawing.Point(0, 0);
    this.oLblFilesTop.Name = "oLblFilesTop";
    this.oLblFilesTop.Size = new System.Drawing.Size(440, 16);
    this.oLblFilesTop.TabIndex = 6;
    this.oLblFilesTop.Text = "Track List";
    this.oLblFilesTop.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
```

```
//
// oLstFiles
//
this.oLstFiles.AllowDrop = true;
this.oLstFiles.BackColor = System.Drawing.Color.White;
this.oLstFiles.Dock = System.Windows.Forms.DockStyle.Fill;
this.oLstFiles.DrawMode = System.Windows.Forms.DrawMode.OwnerDrawFixed;
this.oLstFiles.HorizontalExtent = 800;
this.oLstFiles.HorizontalScrollbar = true;
this.oLstFiles.ItemHeight = 18;
this.oLstFiles.Location = new System.Drawing.Point(0, 0);
this.oLstFiles.Name = "oLstFiles";
this.oLstFiles.ScrollAlwaysVisible = true;
this.oLstFiles.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended;
this.oLstFiles.Size = new System.Drawing.Size(440, 616);
this.oLstFiles.TabIndex = 2;
this.toolTip1.SetToolTip(this.oLstFiles, "Drag files here from your Operating
System file explorer folders");
this.oLstFiles.KeyDown += new System.Windows.Forms.KeyEventHandler(this.
oLstFiles_KeyDown);
this.oLstFiles.DoubleClick += new System.EventHandler(this.
oLstFiles_DoubleClick);
this.oLstFiles.DragDrop += new System.Windows.Forms.DragEventHandler(this.
oLstFiles_DragDrop);
this.oLstFiles.MouseMove += new System.Windows.Forms.MouseEventHandler(this.
oLstFiles_MouseMove);
this.oLstFiles.MouseLeave += new System.EventHandler(this.oLstFiles_MouseLeave);
this.oLstFiles.DragEnter += new System.Windows.Forms.DragEventHandler(this.
oLstFiles_DragEnter);
this.oLstFiles.DrawItem += new System.Windows.Forms.DrawItemEventHandler(this.
oLstFiles_DrawItem);
//
// oImgMmGui
//
this.oImgMmGui.ColorDepth = System.Windows.Forms.ColorDepth.Depth32Bit;
this.oImgMmGui.ImageSize = new System.Drawing.Size(21, 20);
this.oImgMmGui.ImageStream = ((System.Windows.Forms.ImageListStreamer)(resources
.GetObject("oImgMmGui.ImageStream")));
this.oImgMmGui.TransparentColor = System.Drawing.Color.Transparent;
//
// oFldrBrowser1
//
this.oFldrBrowser1.SelectedPath = "Select the folder to add to the track list";
//
// timer1
//
this.timer1.Interval = 2000;
this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
//
// oImgTrackItems
//
this.oImgTrackItems.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
this.oImgTrackItems.ImageSize = new System.Drawing.Size(17, 18);
this.oImgTrackItems.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgTrackItems.ImageStream")));
this.oImgTrackItems.TransparentColor = System.Drawing.Color.Transparent;
//
// oFileBrowser1
//
this.oFileBrowser1.Multiselect = true;
this.oFileBrowser1.Title = "Select the individual files to add to the track list";
";
//
// oPnlFilesFill
//
this.oPnlFilesFill.AutoScroll = true;
this.oPnlFilesFill.BackColor = System.Drawing.Color.White;
this.oPnlFilesFill.Controls.Add(this.oLstFiles);
this.oPnlFilesFill.Dock = System.Windows.Forms.DockStyle.Fill;
this.oPnlFilesFill.Location = new System.Drawing.Point(568, 16);
this.oPnlFilesFill.Name = "oPnlFilesFill";
this.oPnlFilesFill.Size = new System.Drawing.Size(440, 624);
this.oPnlFilesFill.TabIndex = 5;
```

```

        //
        // oContMen1
        //
        this.oContMen1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
this.oMnuAddFolder,
this.oMnuAddFiles});
        //
        // oMnuAddFolder
        //
        this.oMnuAddFolder.Index = 0;
        this.oMnuIcons.SetMenuIcons(this.oMnuAddFolder, "1");
        this.oMnuAddFolder.OwnerDraw = true;
        this.oMnuAddFolder.Text = "Add an entire folder to the current track list";
        this.oMnuAddFolder.Click += new System.EventHandler(this.oMnuAddFolder_Click);
        //
        // oMnuAddFiles
        //
        this.oMnuAddFiles.Index = 1;
        this.oMnuIcons.SetMenuIcons(this.oMnuAddFiles, "0");
        this.oMnuAddFiles.OwnerDraw = true;
        this.oMnuAddFiles.Text = "Add a number of files to the current track list";
        this.oMnuAddFiles.Click += new System.EventHandler(this.oMnuAddFiles_Click);
        //
        // oMnuIcons
        //
        this.oMnuIcons.ImageList = this.oImgTrackItems;
        //
        // uctMediaPlayer
        //
        this.Controls.Add(this.oPnlFilesFill);
        this.Controls.Add(this.oPnlFilesTop);
        this.Controls.Add(this.oPnlFilesBottom);
        this.Controls.Add(this.oSplit1v);
        this.Controls.Add(this.oPnlMainL);
        this.Name = "uctMediaPlayer";
        this.Size = new System.Drawing.Size(1008, 688);
        this.Resize += new System.EventHandler(this.uctMediaPlayer_Resize);
        this.oPnlMainL.ResumeLayout(false);
        ((System.ComponentModel.ISupportInitialize)(this.oMMPlayer)).EndInit();
        this.oPnlFilesBottom.ResumeLayout(false);
        this.oPnlFilesTop.ResumeLayout(false);
        this.oPnlFilesFill.ResumeLayout(false);
        this.ResumeLayout(false);
    }
#endregion
#region Private Methods
#region GUI User interactions

    /// <summary>
    /// Event handler that occurs when user drags item into the track list
    /// Simple enables drop effects for the track list
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oLstFiles_DragEnter(object sender, System.Windows.Forms.DragEventArgs
e)
    {
        try
        {
            // make sure they're actually dropping files (not text or anything else)
            if( e.Data.GetDataPresent(DataFormats.FileDrop, false) == true )
                // allow them to continue
                // (without this, the cursor stays a "NO" symbol
                e.Effect = DragDropEffects.All;
        }
        catch (Exception ex)
        {
        }
    }
}

    /// <summary>

```

```
/// Event handler that occurs when user drops item into the track list
/// Relieves all the files from the drag data and attempts to add the
/// files to the track list
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLstFiles_DragDrop(object sender, System.Windows.Forms.DragEventArgs e)
{
    try {
        // transfer the filenames to a string array
        string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
        //add the dragged files to track list
        addFiles(files);
    }
    catch (Exception ex)
    {
    }
}

/// <summary>
/// Event handler that occurs when user selects a new GUI style for
/// the embedded WindowsMediaPlayer control.
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oCmbMmGui_SelectedIndexChanged(object sender, System.EventArgs e)
{
    //change the embedded WindowsMediaPlayer control GUI style to that chosen
    oMMPPlayer.uiMode = oCmbMmGui.SelectedItem.ToString();
}

/// <summary>
/// Event handler that occurs when user selects the up button. Will attempt
/// to move all elected tracks up by 1 position
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnUp_Click(object sender, System.EventArgs e)
{
    //sanity check
    if (oLstFiles.SelectedIndex < 0)
        return;
    //if at top, return
    topSelectedItem = oLstFiles.SelectedIndices[0];
    if (topSelectedItem == 0)
        return;
    //move items up by 1 position
    for (int i=0 ; i < oLstFiles.SelectedIndices.Count ; i++)
    {
        moveItemUp(oLstFiles.SelectedIndices[i]);
    }
}

/// <summary>
/// Event handler that occurs when user selects the down button. Will attempt
/// to move all elected tracks down by 1 position
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnDown_Click(object sender, System.EventArgs e)
{
    //sanity check
    if (oLstFiles.SelectedIndex < 0)
        return;
    bottomSelectedItem = oLstFiles.SelectedIndices[oLstFiles.SelectedIndices.
Count-1];
    //if at bottom, return
    if (bottomSelectedItem == oLstFiles.Items.Count -1)
        return;

    //move items down by 1 position
```

```
        for (int i = oLstFiles.SelectedIndices.Count - 1 ; i >= 0 ; i--)
        {
            moveItemDown(oLstFiles.SelectedIndices[i]);
        }
    }

    /// <summary>
    /// Event handler that occurs when user selects the clear button. Will clear
    /// the track list
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oBtnClearList_Click(object sender, System.EventArgs e)
    {
        oLstFiles.Items.Clear();
        oMMPlayer.Ctlcontrols.stop();
    }

    /// <summary>
    /// Event handler that occurs when user selects the key down when the
    /// track list has focus. If the Key Code of the key == Delete will
    /// delete all selected items from the track list
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oLstFiles_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
    {
        //If the Key Code of the key == Delete
        if (e.KeyCode == Keys.Delete)
        {
            //if no item selected return
            if (oLstFiles.SelectedIndex == -1)
            {
                return;
            }
            else
            {
                //set global flag to state we are changing order
                ChangingTrackOrder = true;
                // Simply try and change the state of the player to stop
                oMMPlayer.Ctlcontrols.stop();
                //remove all selected items
                for (int i=oLstFiles.Items.Count -1; i> -1; i--)
                {
                    if (oLstFiles.SelectedItems.Contains(oLstFiles.Items[i]))
                    {
                        oLstFiles.Items.RemoveAt(i);
                    }
                }
                //set global flag to state we are no longer changing order
                ChangingTrackOrder = false;
                //get new track, if there is one to get, user may have deleted
                //all items
                if (oLstFiles.Items.Count > 0)
                {
                    oLstFiles.SelectedIndex = 0;
                    getNewTrack();
                }
            }
        }
    }

    /// <summary>
    /// Event handler that occurs when user selects the save track list button.
    /// Attempts to save the track list contents to an XML file of the users
    /// choice, makes use of the <see cref="SB54_CSAI.MediaPlayerControl.XMLReadWrite">
    XMLReadWrite class</see>
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oBtnSaveTracklist_Click(object sender, System.EventArgs e)
    {
        //is there at least 1 track
```

```

    if (oLstFiles.Items.Count < 1)
    {
        MessageBox.Show("There are no tracks to save" +
            "\r\n\r\n\r\n" + "please retry when there are some tracks to save",
            "XML Tracklist Save ERROR", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        return;
    }
    //ok, got at least 1 track, so construct filebrowser
    oSaveFileBrowser1.Title = "Please select where to save your track list to";
    oSaveFileBrowser1.Filter = "XML Files | *.xml";
    oSaveFileBrowser1.FilterIndex = 1;
    oSaveFileBrowser1.FileName = "Tracklist.xml";
    //only continue if user selected OK
    if (oSaveFileBrowser1.ShowDialog(this) == DialogResult.OK)
    {
        //it must be an XML file
        FileInfo fi = new FileInfo(oSaveFileBrowser1.FileName);
        if (!fi.Extension.ToLower().Equals(".xml"))
        {
            MessageBox.Show("Problem with MediaPlayer Control SaveTracklist " +
                "\r\n\r\n\r\n" + " The saved tracklist file must be an XML file,
please retry",
                "XML Tracklist Save ERROR", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        else
        {
            //XML file, so save contents to the users XML file using the
XMLReadWrite class
            XMLReadWrite oTracklistXMLWriter = new XMLReadWrite();
            oTracklistXMLWriter.WriteXMLTrackList(@oSaveFileBrowser1.FileName,
oLstFiles.Items.GetEnumerator());
        }
    }
}

/// <summary>
/// Event handler that occurs when user selects the load track list button.
/// Attempts to load the track list contents of the user specified XML file,
/// makes use of the <see cref="SB54_CSAI.MediaPlayerControl.XMLReadWrite">
XMLReadWrite class</see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnLoadTracklist_Click(object sender, System.EventArgs e)
{
    //construct filebrowser
    oFileBrowser1.Title = "Please select where to load the track list from";
    oFileBrowser1.Filter = "XML Files | *.xml";
    oFileBrowser1.FilterIndex = 1;
    oFileBrowser1.Multiselect = false;
    //only continue if user selected OK
    if (oFileBrowser1.ShowDialog(this) == DialogResult.OK)
    {
        //it must be an XML file
        FileInfo fi = new FileInfo(oFileBrowser1.FileName);
        if (!fi.Extension.ToLower().Equals(".xml"))
        {
            MessageBox.Show("Problem with MediaPlayer Control LoadTracklist " +
                "\r\n\r\n\r\n" + " The requested tracklist file must be an XML file,
please retry",
                "XML Tracklist Load ERROR", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        else
        {
            //tell user it may take a while
            DialogResult res = MessageBox.Show("This operation could take a some
time, as the track list may contain references\r\n " +
                "to a remote computers tracks. Where the remote computer status
needs to be\r\n " +
                "checked for availability, prior to loading its tracks\r\n\r\n\r\n"

```

```

+
        "Only those files that are available will be added to the track list"
\r\n\r\n" +
        "Are you sure you wish to proceed ?",
        "XML Tracklist Load", MessageBoxButtons.YesNo,
        MessageBoxIcon.Information);
    //they are sure, so proceed
    if (res == DialogResult.Yes)
    {
        //XML file, so attempt to load contents of the XML file using the
XMLReadWrite class
        oLstFiles.Items.Clear();
        XMLReadWrite oTracklistXMLReader = new XMLReadWrite();
        oTracklistXMLReader.ReadXMLTrackList(oFileBrowser1.FileName);
        string[] readXMLFiles = oTracklistXMLReader.getXMLReadFiles;
        addFiles(readXMLFiles);
        //tell user that we are now done
        MessageBox.Show("The XML has been parsed and the tracklist now
contains\r\n" +
XML file",
                        "all the available tracks that were present in the
                        "XML Tracklist Load", MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
    }
}
}
}

/// <summary>
/// Event handler that occurs when user selects the add button. Will display
/// 2 menu items, one to add files, one to add folders
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnAdd_Click(object sender, System.EventArgs e)
{
    Button b = (Button)sender;
    oContMen1.Show(b, new Point(5,5));
}

/// <summary>
/// Event handler that occurs when user moves the mouse with the track list in focus
/// Will attempt to get track list item from current mouse position, to show on
popup window
/// <see cref="SB54_CSAI.MediaPlayerControl.frmTrack">frmTrack </see>where the
current item
/// details will be displayed
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLstFiles_MouseMove(object sender, System.Windows.Forms.MouseEventHandler
e)
{
    //could be -1 if no items exist
    int idx = oLstFiles.IndexFromPoint(new Point(e.X,e.Y));

    //is it a valid index
    if (idx > -1)
    {
        try
        {
            if (oLstFiles.SelectedItems.Count > 0)
            {
                //if yes, get item to show on popup details window
                hoveredItem = ((MP3ListItem)oLstFiles.Items[idx]).URL.ToString();
                //making sure the file is available
                if (trResolve.checkFileIsOnline(hoveredItem))
                {
                    allowPopup = true;
                }
            }
        }
    }
}
}
}
}

```

```
        else
        {
            allowPopup = false;;
        }
    }

}

catch (InvalidCastException ex)
{
    //this should not happen, it is working, but sometimes throws this
    //InvalidCastException, even though its working fine doing the cast
    //above
}
}
else
{
    //not valid, so dont allow popup
    allowPopup = false;
}
}

/// <summary>
/// Event handler that occurs when user selects the Add Folder menu. Prompts the
user for
/// a folder to add track list files from. Calls the internal AddFolder() method
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMnuAddFolder_Click(object sender, System.EventArgs e)
{
    AddFolder();
}

/// <summary>
/// Event handler that occurs when user selects the Add Files menu. Prompts the user
for
/// a folder to add track list files from. Calls the internal AddFiles() method
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMnuAddFiles_Click(object sender, System.EventArgs e)
{
    AddFiles();
}

/// <summary>
/// Event handler that occurs when user selects the Clear button. Will clear the
/// track list of all items
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oBtnClear_Click(object sender, System.EventArgs e)
{
    cmdCLEAR();
}

/// <summary>
/// Event handler that occurs when user moves the mouse out of the track list focus
area.
/// Will hide the popup window
/// <see cref="SB54_CSAI.MediaPlayerControl.frmTrack">frmTrack </see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLstFiles_MouseLeave(object sender, System.EventArgs e)
{
    // frmTrack shown, hide it.
    if (oFrmtrack != null)
    {
        oFrmtrack.Hide();
    }
}
}
```

```

/// <summary>
/// Event handler that occurs when user double clicks the track list focus area.
/// On double click will attempt to get the new track at the index the user double
/// clicked
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oLstFiles_DoubleClick(object sender, System.EventArgs e)
{
    //get the new user selected track
    getNewTrack();
    redrawTrackItems();
}

#endregion
#region GUI look and feel methods

/// <summary>
/// Event handler that occurs when the <see cref="SB54_CSAI.MediaPlayerControl.
uctMediaPlayer">main control </see>
/// is resized. Attempt to aintain aspect ratio of embedded WindowsMediaPlayer
/// control and track list in available space, by moving splitter when the
/// <see cref="SB54_CSAI.MediaPlayerControl.uctMediaPlayer">main control </see> is
resized.
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void uctMediaPlayer_Resize(object sender, System.EventArgs e)
{
    oPnlMainL.Width = (int) ( oMMPPlayer.Width + oLstFiles.Width)/ 100 * 60;
    oSplitlv.Location = new Point (oPnlMainL.Top+1,oPnlMainL.Left+1);
}

/// <summary>
/// Event handler that occurs when the Media Player combo box is drawn.
/// Siplly provide custom drawing of combo box items
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oCmbMmGui_DrawItem(object sender, System.Windows.Forms.
DrawItemEventArgs e)
{
    //check index is valid
    if (e.Index != -1)
    {
        // Draw the background of the ComboBox control for each item.
        e.DrawBackground();

        Image img = oImgMmGui.Images[e.Index];
        e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

        // Draw the current item text based on the current Font and the custom brush
settings.
        e.Graphics.DrawString(oCmbMmGui.Items[e.Index].ToString(), e.Font, Brushes.
Black,
            new Point(img.Width+5,e.Bounds.Y + 2));
        e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

        // If the ComboBox has focus, draw a focus rectangle around the selected
item.
        e.DrawFocusRectangle();
    }
}

/// <summary>
/// Event handler that occurs when the Track List list box is drawn.
/// Siplly provide custom drawing of Track List items

```

```

    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oLstFiles_DrawItem(object sender, System.Windows.Forms.
DrawItemEventArgs e)
    {
        //check index is valid
        if (e.Index != -1)
        {
            // Draw the background of the ListBox control for each item.
            e.DrawBackground();

            Image img = oImgTrackItems.Images[0];
            e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

            //If the index being drawn is the currently playing track, paint its text
            BOLD
            if (e.Index == CurrentTrack)
            {
                //create a new BOLD font, but base it on the font associated with the
                list items
                Font fPlaying = new Font(e.Font.FontFamily.GetName(0), e.Font.Size,
                FontStyle.Bold);
                // Draw the current item text based on the current Font and the custom
                brush settings.
                e.Graphics.DrawString(oLstFiles.Items[e.Index].ToString(), fPlaying,
                Brushes.Black,
                new Point(img.Width+5, e.Bounds.Y));
            }
            //otherwise simply paint it using a normal weight font
            else
            {
                // Draw the current item text based on the current Font and the custom
                brush settings.
                e.Graphics.DrawString(oLstFiles.Items[e.Index].ToString(), e.Font,
                Brushes.Black,
                new Point(img.Width+5, e.Bounds.Y));
            }
            e.Graphics.DrawImage(img, new Point(e.Bounds.X, e.Bounds.Y));

            // If the ListBox has focus, draw a focus rectangle around the selected item
            e.DrawFocusRectangle();
        }
    }

    /// <summary>
    /// Event handler that occurs when the internal timer ticks.
    /// The timer is used in conjunction with the mouseOver event of the track list
    /// such that the user must be hovering over an item for (n) time before
    /// the popup window
    /// <see cref="SB54_CSAI.MediaPlayerControl.frmTrack">frmTrack </see> is shown
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void timer1_Tick(object sender, System.EventArgs e)
    {
        //if popup allowed, must have valid track hovered over from Track List
        if (allowPopup)
        {
            //so show popup
            oFrmtrack.TrackName = hoveredItem;
            oFrmtrack.Show();
            oFrmtrack.setDetails();
        }
    }

    /// <summary>
    /// Event handler that occurs embedded WindowsMediaPlayer control has a MediaError.
    /// Simple show the user the error message from the embedded WindowsMediaPlayer
    control.
    /// </summary>

```

```
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMMPayer_MediaError(object sender, AxWMPLib.
_WMPPOCXEvents_MediaErrorEvent e)
{
    // If the Player encounters a corrupt or missing file,
    // show the hexadecimal error code and URL.
    // Most likely error is : C00D1197: Cannot play the file, which is described in
detail at
    // http://www.microsoft.com/windows/windowsmedia/player/9series/playererrors.
aspx#c00d1197\_0x00000000
    try
    {
        //get the offending media item

        //If the embedded media player can not play the file because
        // a) The file type may not be supported
        // b) The codec that was used to compress the file may not be supported
        // There is not much that can be done apart from alert the user to this.
        IWMPMedia2 errSource = e.pMediaObject as IWMPMedia2;
        IWMPErrorItem errorItem = errSource.Error;
        MessageBox.Show("Problem with MediaPlayer Control uctMediaPlayer " +
            "\r\n\r\n\r\n" + errorItem.errorCode.ToString("X")
            + " in " + errSource.sourceURL,
            "Media Control ERROR", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        cmdFF();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Problem with MediaPlayer Control uctMediaPlayer " +
            "\r\n\r\n\r\n\r\n" + ex.ToString(),
            "Media Control ERROR", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

/// <summary>
/// Event handler that occurs embedded WindowsMediaPlayer control PlayStateChange
changes.
/// If the state is stopped and the track has finished playback (check the current
Media player
/// position against the current media player MediaItem duration. If they are equal,
then the
/// MediaItem must be finished play back (End Of Stream), this is done by the
oMMPayer_MediaChange
/// method) attempt to get the next track using the cmdFF() method
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMMPayer_PlayStateChange(object sender, AxWMPLib.
_WMPPOCXEvents_PlayStateChangeEvent e)
{
    //get the state
    WMPPlayState newState = (WMPPlayState)e.newState;
    switch (newState)
    {
        //IF Stopped, that could mean that the end of stream has been reached OR
that the user simply
        //clicked stop, so need to check which one is the current state
        case WMPPlayState.wmppsStopped:
        {
            //the method oMMPayer_MediaChange, checks the current media player
position against
            //the current media item duration, and drives the allowFF flag. So
simply check to
            //see if FF is allowed. If it is this means the current track is
```

```
finished so end of stream
    //must be true
    if (allowFF)
    {
        //do do the FF, and de-latch the ff request flag
        cmdFF();
        allowFF=false;
    }
    break;
}
default:
    break;
}
}

/// <summary>
/// Event handler that occurs embedded WindowsMediaPlayer control media item changes
.
/// This event is used to check the current Media player position against the
current
/// media player MediaItem duration. If they are equal, then the MediaItem must be
finished
/// play back (End Of Stream) so set a global flag that allows/disabled the FF
command
/// which is used within the oMMPlayer_PlayStateChange event
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMMPlayer_MediaChange(object sender, AxWMPLib.
_WMPOCXEvents_MediaChangeEvent e)
{
    //get current position, and duration
    double currPos = oMMPlayer.Ctlcontrols.currentPosition;
    double duration = oMMPlayer.currentMedia.duration;
    //if they are equal give or take a few milliseconds
    if ( Math.Ceiling(currPos).Equals(Math.Ceiling(0.0)))
    {
        //allow FF to get next track, which is used by the oMMPlayer_PlayStateChange
event
        allowFF=true;
    }
    else
    {
        allowFF=false;
    }
}

/// <summary>
/// Event handler that occurs embedded WindowsMediaPlayer control OpenStateChange
changes.
/// Attempt to make the player play
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oMMPlayer_OpenStateChange(object sender, AxWMPLib.
_WMPOCXEvents_OpenStateChangeEvent e)
{
    // Simply try and change the state of the player to play
    oMMPlayer.Ctlcontrols.play();
}

#endregion
#region Helper Methods

/// <summary>
/// Causes the track list items to be redrawn
/// </summary>
private void redrawTrackItems()
{
    //then cause the listbox to do a repaint, which will then correctly BOLD the
//currently selected track
    oLstFiles.Invalidate(oLstFiles.Bounds);
}
```

```

        oLstFiles.Update();
    }

    /// <summary>
    /// Informs the embedded WindowsMediaPlayer control to play a new
    /// track based on the next available track list item.
    /// If the track list items are being re-ordered this method will
    /// be ignored
    /// </summary>
    private void getNewTrack()
    {
        //is there an index selected
        if (oLstFiles.SelectedIndex != -1)
        {
            //if there is NOT a reorder occurring
            if (! ChangingTrackOrder)
            {
                //Informs the WindowsMediaPlayer control to play the next track
                try
                {
                    //get new track name
                    string file = ((MP3ListItem)oLstFiles.SelectedItem).URL.ToString;
                }
                //if the file is on line, use it, otherwise do FF
                if (trResolve.checkFileIsOnline(file))
                {
                    //store the index of new track
                    CurrentTrack = oLstFiles.SelectedIndex;
                    //play the new track
                    oMMPPlayer.URL = file;
                    oMMPPlayer.Ctlcontrols.play();
                    //create a new Scan event for event subscribers, START OF
                    CurrentMediaTrackEventArgs ea = new
                    CurrentMediaTrackEventArgs(file);
                    OnCurrentMediaTrack(ea);
                }
                else
                {
                    cmdFF();
                }
            }
            catch (InvalidCastException iex)
            {
                //this should not happen, it is working, but sometimes throws
                //InvalidCastException, even though its working fine doing the
                //above
            }
            catch (Exception ex)
            {
                MessageBox.Show("Problem with track selection uctMediaPlayer " +
                    "\r\n\r\n\r\n" + ex.Message,
                    "Track Selection ERROR", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
    }
}

/// <summary>
/// Moves an item (at the index specified by the input parameter)
/// within the track list down by 1 position.
/// </summary>
/// <param name="item">Representing the item within the track list that
/// the user requires to be moved down by 1 position</param>
private void moveItemDown(int item)
{

```

```
        //set the global flag to state that order is being changed
        ChangingTrackOrder = true;
        //do a sanity check, then do the reordering moving the item down by 1
        if (item == oLstFiles.Items.Count - 1)
            return;
        oLstFiles.Items.Insert(item + 2, oLstFiles.Items[item].ToString());
        oLstFiles.Items.RemoveAt(item);
        ChangingTrackOrder = false;
        oLstFiles.SelectedIndex = (item + 1);
    }

    /// <summary>
    /// Moves an item (at the index specified by the input parameter)
    /// within the track list up by 1 position.
    /// </summary>
    /// <param name="item">Representing the item within the track list that
    /// the user requires to be moved up by 1 position</param>
    private void moveItemUp(int item)
    {
        //set the global flag to state that order is being changed
        ChangingTrackOrder = true;
        //do a sanity check, then do the reordering moving the item up by 1
        if (item == 0)
            return;
        oLstFiles.Items.Insert(item - 1, oLstFiles.Items[item].ToString());
        oLstFiles.Items.RemoveAt(item + 1);
        ChangingTrackOrder = false;
        oLstFiles.SelectedIndex = (item - 1);
    }

    /// <summary>
    /// Check to see if the input string parameter is contained within the
    /// internal supportedFiles collection
    /// </summary>
    /// <param name="fileExtension">The current file extension</param>
    /// <returns>True if the current file extension is one that is contained within
    /// the internal supportedFiles collection, False otherwise</returns>
    private bool isSupportedFile(string fileExtension)
    {
        return supportedFiles.Contains(fileExtension.ToLower());
    }

    /// <summary>
    /// Add all the allowable extensions to the internal supportedFiles collection
    /// </summary>
    private void setUpSupportedFiles()
    {
        //add Window Audio Files
        supportedFiles.Add(".wav");
        supportedFiles.Add(".snd");
        supportedFiles.Add(".au");
        supportedFiles.Add(".aif");
        supportedFiles.Add(".aifc");
        supportedFiles.Add(".aiff");
        supportedFiles.Add(".wma");
        supportedFiles.Add(".mp3");

        //add Window Video Files
        supportedFiles.Add(".asf");
        supportedFiles.Add(".wm");
        supportedFiles.Add(".wma");
    }

    /// <summary>
    /// Provides a string that represents of all supported files
    /// </summary>
    /// <returns>String representation of all supported files within the
    /// internal supportedFiles collection</returns>
    private string getSupportedFiles()
    {
        string items = "";
    }
}
```

```

        foreach (string s in supportedFiles)
        {
            items += s + "\n";
        }
        return items;
    }

    /// <summary>
    /// Asks the user for a folder to add to track list
    /// </summary>
    private void AddFolder()
    {
        // Show the FolderBrowserDialog.
        DialogResult result = oFldrBrowser1.ShowDialog(this);

        //did you pressed ok on dialog
        if( result == DialogResult.OK )
        {
            string folderName = oFldrBrowser1.SelectedPath;
            ArrayList allFiles = new ArrayList();
            //store all files and directories from SelectedPath
            string[] files = Directory.GetFiles(folderName);
            for (int i=0;i<files.Length;i++)
            {
                allFiles.Add(files[i]);
            }

            string[] dirs = Directory.GetDirectories(folderName);
            for (int i=0;i<dirs.Length;i++)
            {
                allFiles.Add(dirs[i]);
            }
            string[] allFilesAndDir = allFiles.ToArray(Type.GetType("System.String")) as string[];
            //add all files and folder to track list
            addFiles(allFilesAndDir);
        }
    }

    /// <summary>
    /// Asks the user for a file(s) to add to track list
    /// </summary>
    private void AddFiles()
    {
        oFileBrowser1.Title = "Please select which tracks to load";
        oFileBrowser1.Filter = "All files (*.*)|*.*";
        oFileBrowser1.FilterIndex = 1;
        oFileBrowser1.Multiselect = true;
        oFileBrowser1.FileName="";

        // Show the FileBrowserDialog
        DialogResult result = oFileBrowser1.ShowDialog(this);
        //did you pressed ok on dialog
        if( result == DialogResult.OK )
        {
            //add all files to track list
            string[] files = oFileBrowser1.FileNames;
            addFiles(files);
        }
    }

    #endregion
    #endregion
}
#endregion
#region CurrentMediaTrackEventArgs CLASS
/// <summary>
/// Provides the CurrentMediaTrackEventArgs used with the <see cref="SB54_CSAI.
MediaPlayerControl.uctMediaPlayer">
/// Media Player</see> currentMediaTrack event. The currentMediaTrack event is used by
the

```

```
/// <see cref="SB54_CSAI.ServerApp.frmMain">ReMP3 server</see> to inform the ReMP3 client application
/// that there is a new track available
/// </summary>
public class CurrentMediaTrackEventArgs : EventArgs
{
    #region Instance Fields
    //Instance fields
    private readonly string track="";
    #endregion
    #region Public Constructor
    /// <summary>
    /// Constructs a new CurrentMediaTrackEventArgs object using the parameters provided
    /// </summary>
    /// <param name="foldername">The currently scanned folder</param>
    public CurrentMediaTrackEventArgs(string track)
    {
        this.track = track;
    }
    #endregion
    #region Public Methods/Properties

    /// <summary>
    /// Returns the currently scanned folder
    /// </summary>
    /// <returns>Returns the current media track name</returns>
    public string TrackName
    {
        get { return this.track;}
    }
    #endregion
}
#endregion
}
```