

```

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 sb54_CSAI.MiscFormComponents;
using SB54_CSAI.TrackObjects;

namespace SB54_CSAI.ClientTrackList
{
    #region uctClientTrackList CLASS

    /// <summary>
    /// uctClientTrackList 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 uctClientTrackList : System.Windows.Forms.UserControl
    {
        #region Instance fields

        private System.Windows.Forms.Panel oPnlFilesBottom;
        private System.Windows.Forms.Panel oPnlFilesTop;
        private System.Windows.Forms.ToolTip toolTip1;
        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 ArrayList supportedFiles = new ArrayList();
        private string hoveredItem="";
        private bool ChangingTrackOrder=false;
        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 System.Windows.Forms.Panel oPnlBottomTitle;
        private System.Windows.Forms.Label oLblTracks;
        private System.Windows.Forms.Label oLblPlayer;

```

```

private System.Windows.Forms.ImageList oImgAddType;
private System.Windows.Forms.Label oLblAddType;
private System.Windows.Forms.PictureBox oPicREW;
private System.Windows.Forms.PictureBox oPicPlay;
private System.Windows.Forms.PictureBox oPicFF;
private System.Windows.Forms.PictureBox oPicStop;
private System.Windows.Forms.Button oBtnSendFiles;
private System.Windows.Forms.ImageList oImgList16;
private System.Windows.Forms.ContextMenu oContMenuSendFiles;
private System.Windows.Forms.MenuItem oMenuItemAdd;
private System.Windows.Forms.MenuItem oMenuItemClear;
private System.Windows.Forms.Panel oPnlControls;
private int bottomSelectedTrackItem;
/// <summary>
/// RemoteOperationsHandler delegate for the Remote Operations
/// </summary>
public delegate void RemoteOperationsHandler(object sender, RemoteEventArgs e);
/// <summary>
/// Raised if the user initiates a Remote operation for the Re-MP3 Server</see>
/// </summary>
public event RemoteOperationsHandler RemoteOperations;
private TrackResolver trResolve = new TrackResolver();
private string currTrack;
private int CurrentTrackIndex=-1;

#endregion
#region Public Constructor
/// <summary>
/// Constructor, simply initalises all GUI components and setup all
/// the supported files using the setUpSupportedFiles() method
/// </summary>
public uctClientTrackList()
{
    // 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 / Properties

/// <summary>
/// gets or sets the current track. This property will be set by the
/// <see cref="sb54_CS.AI.ClientApp.FrmClient">ReMP3 Client. </see>to indicate
/// that the <see cref="sb54_CS.AI.remoteInterfaces.RemotingObject">RemotingObject </see>
see>
/// remoting object has received a new track from the MediaPlayer, that may be in
this
/// ReMP3 client list
/// </summary>
public string currentTrackName
{
    get
    {
        return currTrack;
    }
    set
    {
        //see if the ReMP3 client actually has this track name in its playlist
        checkToSeeIfThisTrackExists(value);
        //redraw the tracklist items, as the tracklist may now need to show a newly
        //selected track in BOLD
        redrawTrackItems();
        //store the trackname
        currTrack = value;
    }
}
}

```

```

    /// <summary>
    /// Raises the RemoteOperations event for the uctClientTrackList user control. Any
users of the
    /// uctClientTrackList may now subscribe to this event using a
RemoteOperationsHandler
    /// delegate.
    /// </summary>
    /// <param name="e">The <see cref="SB54_CSAI.ClientTrackList.RemoteEventArgs">Remote
control event args</see></param>
    public virtual void OnRemoteOperations(RemoteEventArgs e)
    {
        if (RemoteOperations != null)
        {
            // Invokes the delegates.
            RemoteOperations(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>
    /// Clears the track list of all items
    /// </summary>
    public void clearList()
    {
        oLstFiles.Items.Clear();
    }

    /// <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
    /// </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[] dirFiles = arDirFiles.ToArray(typeof(System.String)) as
        String[];

        //make recursive call based on new files retrieved for this
        directory
            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 ClientTrackList " +
        "\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(uctClientTrackList));
    this.oPnlFilesBottom = new System.Windows.Forms.Panel();
    this.oPnlControls = new System.Windows.Forms.Panel();
    this.oPicREW = new System.Windows.Forms.PictureBox();
    this.oPicPlay = new System.Windows.Forms.PictureBox();
    this.oPicFF = new System.Windows.Forms.PictureBox();
    this.oPicStop = new System.Windows.Forms.PictureBox();
    this.oLblAddType = new System.Windows.Forms.Label();
    this.oBtnSendFiles = new System.Windows.Forms.Button();
    this.oImgList16 = new System.Windows.Forms.ImageList(this.components);
    this.oPnlBottomTitle = new System.Windows.Forms.Panel();
    this.oLblPlayer = new System.Windows.Forms.Label();
    this.oLblTracks = new System.Windows.Forms.Label();
    this.oBtnLoadTracklist = new System.Windows.Forms.Button();
    this.oBtnSaveTracklist = new System.Windows.Forms.Button();
    this.oBtnClear = new System.Windows.Forms.Button();
    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.toolTip1 = new System.Windows.Forms.ToolTip(this.components);
}

```

```

this.oLstFiles = new System.Windows.Forms.ListBox();
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.oMenuItemAdd = new System.Windows.Forms.MenuItem();
this.oMenuItemClear = new System.Windows.Forms.MenuItem();
this.oSaveFileBrowser1 = new System.Windows.Forms.SaveFileDialog();
this.oImgAddType = new System.Windows.Forms.ImageList(this.components);
this.oContMenuSendFiles = new System.Windows.Forms.ContextMenu();
this.oPnlFilesBottom.SuspendLayout();
this.oPnlControls.SuspendLayout();
this.oPnlBottomTitle.SuspendLayout();
this.oPnlFilesFill.SuspendLayout();
this.SuspendLayout();
//
// 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.oPnlControls);
this.oPnlFilesBottom.Controls.Add(this.oPnlBottomTitle);
this.oPnlFilesBottom.Controls.Add(this.oBtnLoadTracklist);
this.oPnlFilesBottom.Controls.Add(this.oBtnSaveTracklist);
this.oPnlFilesBottom.Controls.Add(this.oBtnClear);
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(0, 584);
this.oPnlFilesBottom.Name = "oPnlFilesBottom";
this.oPnlFilesBottom.Size = new System.Drawing.Size(1008, 104);
this.oPnlFilesBottom.TabIndex = 3;
//
// oPnlControls
//
this.oPnlControls.Controls.Add(this.oPicREW);
this.oPnlControls.Controls.Add(this.oPicPlay);
this.oPnlControls.Controls.Add(this.oPicFF);
this.oPnlControls.Controls.Add(this.oPicStop);
this.oPnlControls.Controls.Add(this.oLblAddType);
this.oPnlControls.Controls.Add(this.oBtnSendFiles);
this.oPnlControls.Dock = System.Windows.Forms.DockStyle.Right;
this.oPnlControls.Location = new System.Drawing.Point(576, 24);
this.oPnlControls.Name = "oPnlControls";
this.oPnlControls.Size = new System.Drawing.Size(432, 80);
this.oPnlControls.TabIndex = 26;
//
// oPicREW
//
this.oPicREW.Image = ((System.Drawing.Image)(resources.GetObject("oPicREW.Image"
)));
this.oPicREW.Location = new System.Drawing.Point(8, 24);
this.oPicREW.Name = "oPicREW";
this.oPicREW.Size = new System.Drawing.Size(40, 40);
this.oPicREW.TabIndex = 21;
this.oPicREW.TabStop = false;
this.toolTip1.SetToolTip(this.oPicREW, "Rewind the ReMP3 Server tracklist by 1
track");
this.oPicREW.Click += new System.EventHandler(this.oPicREW_Click);
//
// oPicPlay
//
this.oPicPlay.Image = ((System.Drawing.Image)(resources.GetObject("oPicPlay.
Image")));
this.oPicPlay.Location = new System.Drawing.Point(56, 24);
this.oPicPlay.Name = "oPicPlay";

```

```

        this.oPicPlay.Size = new System.Drawing.Size(40, 40);
        this.oPicPlay.TabIndex = 22;
        this.oPicPlay.TabStop = false;
        this.toolTip1.SetToolTip(this.oPicPlay, "Play the current tracklist at the ReMp3
Server");
        this.oPicPlay.Click += new System.EventHandler(this.oPicPlay_Click);
        //
        // oPicFF
        //
        this.oPicFF.Image = ((System.Drawing.Image)(resources.GetObject("oPicFF.Image"
)));
        this.oPicFF.Location = new System.Drawing.Point(104, 24);
        this.oPicFF.Name = "oPicFF";
        this.oPicFF.Size = new System.Drawing.Size(40, 40);
        this.oPicFF.TabIndex = 23;
        this.oPicFF.TabStop = false;
        this.toolTip1.SetToolTip(this.oPicFF, "Fastforward the ReMP3 Server tracklist by
1 track");
        this.oPicFF.Click += new System.EventHandler(this.oPicFF_Click);
        //
        // oPicStop
        //
        this.oPicStop.Image = ((System.Drawing.Image)(resources.GetObject("oPicStop.
Image")));
        this.oPicStop.Location = new System.Drawing.Point(160, 24);
        this.oPicStop.Name = "oPicStop";
        this.oPicStop.Size = new System.Drawing.Size(40, 40);
        this.oPicStop.TabIndex = 24;
        this.oPicStop.TabStop = false;
        this.toolTip1.SetToolTip(this.oPicStop, "Stop the Audio playback at the ReMp3
Server");
        this.oPicStop.Click += new System.EventHandler(this.oPicStop_Click);
        //
        // oLblAddType
        //
        this.oLblAddType.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)))
;
        this.oLblAddType.Location = new System.Drawing.Point(216, 8);
        this.oLblAddType.Name = "oLblAddType";
        this.oLblAddType.Size = new System.Drawing.Size(208, 24);
        this.oLblAddType.TabIndex = 20;
        this.oLblAddType.Text = "How should tracks be added to Server";
        this.oLblAddType.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
        //
        // oBtnSendFiles
        //
        this.oBtnSendFiles.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
        this.oBtnSendFiles.ForeColor = System.Drawing.Color.Black;
        this.oBtnSendFiles.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
        this.oBtnSendFiles.ImageIndex = 0;
        this.oBtnSendFiles.ImageList = this.oImgList16;
        this.oBtnSendFiles.Location = new System.Drawing.Point(232, 32);
        this.oBtnSendFiles.Name = "oBtnSendFiles";
        this.oBtnSendFiles.Size = new System.Drawing.Size(88, 24);
        this.oBtnSendFiles.TabIndex = 25;
        this.oBtnSendFiles.Text = "Send Files";
        this.oBtnSendFiles.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
        this.toolTip1.SetToolTip(this.oBtnSendFiles, "Select how the tracks should be
added to the ReMP3 Server");
        this.oBtnSendFiles.Click += new System.EventHandler(this.oBtnSendFiles_Click);
        //
        // oImgList16
        //
        this.oImgList16.ColorDepth = System.Windows.Forms.ColorDepth.Depth24Bit;
        this.oImgList16.ImageSize = new System.Drawing.Size(16, 16);
        this.oImgList16.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgList16.ImageStream")));
        this.oImgList16.TransparentColor = System.Drawing.Color.Transparent;
        //
        // oPnlBottomTitle
        //
        this.oPnlBottomTitle.BackColor = System.Drawing.Color.Navy;

```

```

        this.oPnlBottomTitle.Controls.Add(this.oLblPlayer);
        this.oPnlBottomTitle.Controls.Add(this.oLblTracks);
        this.oPnlBottomTitle.Dock = System.Windows.Forms.DockStyle.Top;
        this.oPnlBottomTitle.Location = new System.Drawing.Point(0, 0);
        this.oPnlBottomTitle.Name = "oPnlBottomTitle";
        this.oPnlBottomTitle.Size = new System.Drawing.Size(1008, 24);
        this.oPnlBottomTitle.TabIndex = 18;
        //
        // oLblPlayer
        //
        this.oLblPlayer.Dock = System.Windows.Forms.DockStyle.Right;
        this.oLblPlayer.Font = new System.Drawing.Font("Tahoma", 14.25F, System.Drawing.
FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.oLblPlayer.ForeColor = System.Drawing.Color.White;
        this.oLblPlayer.Location = new System.Drawing.Point(592, 0);
        this.oLblPlayer.Name = "oLblPlayer";
        this.oLblPlayer.Size = new System.Drawing.Size(416, 24);
        this.oLblPlayer.TabIndex = 19;
        this.oLblPlayer.Text = "Re-MP3 Server : Media Player Operations";
        //
        // oLblTracks
        //
        this.oLblTracks.Dock = System.Windows.Forms.DockStyle.Left;
        this.oLblTracks.Font = new System.Drawing.Font("Tahoma", 14.25F, System.Drawing.
FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
        this.oLblTracks.ForeColor = System.Drawing.Color.White;
        this.oLblTracks.Location = new System.Drawing.Point(0, 0);
        this.oLblTracks.Name = "oLblTracks";
        this.oLblTracks.Size = new System.Drawing.Size(280, 24);
        this.oLblTracks.TabIndex = 18;
        this.oLblTracks.Text = "Local Track List Operations";
        //
        // oBtnLoadTracklist
        //
        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(144, 56);
        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(72, 56);
        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, 56);

```

```

        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);
        //
        // 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(216, 56);
    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(248, 56);
    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, 56);
    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.Location = new System.Drawing.Point(0, 0);
    this.oPnlFilesTop.Name = "oPnlFilesTop";
    this.oPnlFilesTop.Size = new System.Drawing.Size(1008, 104);
    this.oPnlFilesTop.TabIndex = 4;
    //
    // 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(1008, 580);
        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);
        //
        // 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(0, 0);
        this.oPnlFilesFill.Name = "oPnlFilesFill";
        this.oPnlFilesFill.Size = new System.Drawing.Size(1008, 584);
        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;
    //
    // oMenuItemAdd
    //
    this.oMenuItemAdd.Index = 0;
    this.oMnuIcons.SetMenuIcons(this.oMenuItemAdd, "2");
    this.oMenuItemAdd.OwnerDraw = true;
    this.oMenuItemAdd.Text = "Add to existing files";
    this.oMenuItemAdd.Click += new System.EventHandler(this.oMenuItemAdd_Click);
    //
    // oMenuItemClear
    //
    this.oMenuItemClear.Index = 1;
    this.oMnuIcons.SetMenuIcons(this.oMenuItemClear, "3");
    this.oMenuItemClear.OwnerDraw = true;
    this.oMenuItemClear.Text = "Clear existing files 1st";
    this.oMenuItemClear.Click += new System.EventHandler(this.oMenuItemClear_Click);
    //
    // oImgAddType
    //
    this.oImgAddType.ColorDepth = System.Windows.Forms.ColorDepth.Depth32Bit;
    this.oImgAddType.ImageSize = new System.Drawing.Size(16, 16);
    this.oImgAddType.ImageStream = ((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("oImgAddType.ImageStream")));
    this.oImgAddType.TransparentColor = System.Drawing.Color.Transparent;
    //
    // oContMenuSendFiles
    //
    this.oContMenuSendFiles.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
this.oMenuItemAdd,
this.oMenuItemClear});
    //
    // uctClientTrackList
    //
    this.Controls.Add(this.oPnlFilesFill);
    this.Controls.Add(this.oPnlFilesTop);
    this.Controls.Add(this.oPnlFilesBottom);
    this.Name = "uctClientTrackList";
    this.Size = new System.Drawing.Size(1008, 688);
    this.oPnlFilesBottom.ResumeLayout(false);
    this.oPnlControls.ResumeLayout(false);
    this.oPnlBottomTitle.ResumeLayout(false);
    this.oPnlFilesFill.ResumeLayout(false);
    this.ResumeLayout(false);
}
#endregion
#region Private Methods
#region GUI User interactions

/// <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)
{
    //redraw the tracklist items, as the tracklist may now need to show a newly
    //selected track in BOLD
    redrawTrackItems();
}

/// <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>

```

```

e) private void oLstFiles_DragEnter(object sender, System.Windows.Forms.DragEventArgs
    {
        // 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;
    }

    /// <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)
    {
        // transfer the filenames to a string array
        string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
        //add the dragged files to track list
        addFiles(files);
    }

    /// <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
        topSelectedTrackItem = oLstFiles.SelectedIndices[0];
        if (topSelectedTrackItem == 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;
        bottomSelectedTrackItem = oLstFiles.SelectedIndices[oLstFiles.SelectedIndices.
Count-1];
        //if at bottom, return
        if (bottomSelectedTrackItem == 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();
    }

    /// <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;
                //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;
            }
        }
    }

    /// <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" +
        "all the available tracks that were present in the XML file",
        "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_CS.AI.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.MouseEventArgs
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)
{
    oLstFiles.Items.Clear();
}

/// <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();
    }
}
#endregion
#region GUI look and feel methods

/// <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 == CurrentTrackIndex)
        {
            //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_CS.AI.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();
    }
}
#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>
/// Looks for the trackName string within the list of tracks, and if found

```

```
/// stores the index at which the trackName was found, so that item can be
/// painted BOLD to show it is the current track
/// </summary>
/// <param name="trackName">A string representing the new trackname</param>
private void checkToSeeIfThisTrackExists(string trackName)
{
    //look through all items, and see if the track exists in the list of tracks
    for(int i=0;i<oLstFiles.Items.Count;i++)
    {
        MP3ListItem li = (MP3ListItem)oLstFiles.Items[i];
        if ( li.URL.Equals(trackName))
        {
            //it does exist, so store its index
            CurrentTrackIndex=i;
        }
    }
}

/// <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 isSupportedFiled(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 Audio 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 getSupportesFiles()
    {
        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
#region RemoteControl operations
/// <summary>
/// Calls the cmdREW of the <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">media
player control</see>
/// control that is hosted by the <see cref="sb54_CSAI.ServerApp.FrmMain">ServerApp
FrmMain</see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPicREW_Click(object sender, System.EventArgs e)
{
    RemoteEventArgs ea = new RemoteEventArgs(RemoteOperationType.REW);
    OnRemoteOperations(ea);
}

/// <summary>
/// Calls the cmdFF of the <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">media
player control</see>
/// control that is hosted by the <see cref="sb54_CSAI.ServerApp.FrmMain">ServerApp
FrmMain</see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPicFF_Click(object sender, System.EventArgs e)
{
    RemoteEventArgs ea = new RemoteEventArgs(RemoteOperationType.FF);
    OnRemoteOperations(ea);
}

/// <summary>
/// Calls the cmdSTOP of the <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">media
player control</see>
/// control that is hosted by the <see cref="sb54_CSAI.ServerApp.FrmMain">ServerApp
FrmMain</see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPicStop_Click(object sender, System.EventArgs e)
{
    RemoteEventArgs ea = new RemoteEventArgs(RemoteOperationType.STOP);
    OnRemoteOperations(ea);
}

/// <summary>
/// Calls the cmdPLAY of the <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">media
player control</see>
/// control that is hosted by the <see cref="sb54_CSAI.ServerApp.FrmMain">ServerApp
FrmMain</see>
/// </summary>
/// <param name="sender">The source object that raised the event</param>
/// <param name="e">The event arguments</param>
private void oPicPlay_Click(object sender, System.EventArgs e)
{
    RemoteEventArgs ea = new RemoteEventArgs(RemoteOperationType.PLAY);
    OnRemoteOperations(ea);
}

/// <summary>
/// Shows a send files menu

```

```

    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oBtnSendFiles_Click(object sender, System.EventArgs e)
    {
        Button b = (Button)sender;
        oContMenuSendFiles.Show(b, new Point(0,0));
    }

    /// <summary>
    /// Calls the sendFiles method with RemoteOperationType.ADD_FILES
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oMenuItemAdd_Click(object sender, System.EventArgs e)
    {
        sendFiles(RemoteOperationType.ADD_FILES);
    }

    /// <summary>
    /// Calls the sendFiles method with RemoteOperationType.CLEAR_ADD_FILES
    /// </summary>
    /// <param name="sender">The source object that raised the event</param>
    /// <param name="e">The event arguments</param>
    private void oMenuItemClear_Click(object sender, System.EventArgs e)
    {
        sendFiles(RemoteOperationType.CLEAR_ADD_FILES);
    }

    /// <summary>
    /// Causes the <see cref="sb54_CSAI.ServerApp.FrmMain">ServerApp FrmMain</see> to
    call the
    /// addFiles() method of the <see cref="SB54_CSAI.MediaPlayerControl.uctMediaPlayer"
    >Embedded Media Player</see>
    /// control. Depending on the rType instruction the server track will either be
    cleared (rType = CLEAR_ADD_FILES)
    /// or the server tracks will have these new tracks added (rType = ADD_FILES)
    /// </summary>
    /// <param name="rType">The <see cref="RemoteOperationType">remote operation type</
    see></param>
    private void sendFiles(RemoteOperationType rType)
    {
        //get a list of files to send to server
        String[] files = new string[oLstFiles.Items.Count];
        for (int i=0; i < oLstFiles.Items.Count; i++)
        {
            files[i] = ((MP3ListItem)oLstFiles.Items[i]).URL.ToString();
        }
        //raise event
        RemoteEventArgs ea = new RemoteEventArgs(files,rType);
        OnRemoteOperations(ea);
    }
    #endregion
    #endregion
}
#endregion

#region Public Enumerations
/// <summary>
/// Defines the operation type of the current Remote Operation
/// </summary>
public enum RemoteOperationType
{
    /// <summary>
    /// Rewind operation
    /// </summary>
    REW,
    /// <summary>
    /// Fastforward operation
    /// </summary>
    FF,
    /// <summary>
    /// Play operation

```

```

    /// </summary>
    PLAY,
    /// <summary>
    /// Stop operation
    /// </summary>
    STOP,
    /// <summary>
    /// Add Files operation
    /// </summary>
    ADD_FILES,
    /// <summary>
    /// Add Files but clears existing operation
    /// </summary>
    CLEAR_ADD_FILES
}
#endregion
#region RemoteEventArgs CLASS
/// <summary>
/// Provides the RemoteEventArgs used with the <see cref="SB54_CSAI.ClientTrackList.
uctClientTrackList">
/// Client Track List</see> RemoteOperations event. The RemoteOperations event is used
to add files to the
/// <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">media player control</see>
/// </summary>
public class RemoteEventArgs : EventArgs
{
    #region Instance Fields
    //instance fields
    private string[] files;
    private RemoteOperationType opType = RemoteOperationType.PLAY;
    #endregion
    #region Public Constructors
    /// <summary>
    /// Constructs a new RemoteEventArgs object using the parameters provided, used for
    /// ADD_FILES,CLEAR_ADD_FILES operation
    /// </summary>
    /// <param name="files">An array of files to use</param>
    /// <param name="opType"><see cref="SB54_CSAI.ClientTrackList.RemoteOperationType">
The current Remote operation</see></param>
    public RemoteEventArgs(string[] files,RemoteOperationType opType)
    {
        this.files = files;
        this.opType = opType;
    }

    /// <summary>
    /// Constructs a new RemoteEventArgs object using the parameters provided, used for
    /// REW,FF,STOP,PLAY operations
    /// </summary>
    /// <param name="opType"><see cref="SB54_CSAI.ClientTrackList.RemoteOperationType">
The current Remote operation</see></param>
    public RemoteEventArgs(RemoteOperationType opType)
    {
        this.opType = opType;
    }
    #endregion
    #region Public Methods/Properties
    /// <summary>
    /// Returns an array of string representing the playable files that should be
    /// added to the <see cref="SB54_CSAI.MediaPlayer.uctMediaPlayer">
    /// media player</see>
    /// </summary>
    /// <returns>An array of string representing the playable files</returns>
    public string[] PlayableFiles
    {
        get { return files;}
    }

    /// <summary>
    /// Gets the current RemoteOperationType of this event argument
    /// </summary>
    public RemoteOperationType OperationType
    {

```

```
        get
        {
            return this.opType;
        }
    }
#endregion
}
#endregion
}
```