MRS SDK for .net Applications

Introduction

MRS, Media Relay System, is a client/server communications system for delivering video, audio, location and various events from capture devices to monitor applications. The system is usually built around a centralized server that controls permissions and directs the traffic between the devices and the monitors. This document describes the MRS CLI (Common Language Interface) DLL version 6.0.0.1.

Minimum Requirements

In order to develop a .net application with the MRS CLI, the developer needs a Windows PC with .net 2.0 (or higher) and Microsoft Visual Studio 2005 (or higher). The application must be a 32 bit application and must be distributed with the following DLL files:

```
MRS_CLI.dll
StateReportDLL.dll
```

MRS Classes

Class Name	Description
ICaptureDeviceStatus	An interface for handling device status changes and retrieving details about the current status.
IDisconnectable	An interface for disconnecting a client/server connection.
IDownloadControl	An interface for control and notifications of media file download.
IMediaRecorderConfiguration	An interface for configuring media recording parameters.
IMonitor	An interface for receiving events from connected devices.
Management	The management class for connection and access to online device properties.
SMediaFile	A structure that gives information about a media file as part of a media files list.

ICaptureDeviceStatus Interface Properties

Property Name	Description
ID	Device identifier.
IMEI	IMEI of the internal modem.
Name	Device name.

VideoChannelName[index]	Video channel name for a specific channel index.
VideoSignalState[index]	Video signal state for a specific channel index.

IDisconnectable Interface Methods

Method Name	**Description**
Disconnect()	Disconnect from server.

IDownloadControl Interface Events

Event Name	**Description**
TransportCompletion	Download is done.
TransportCancelation	Download was aborted.
TransportProgress(name,done,total,rate)	Download progress for a specific file.

IDownloadControl Interface Methods

Method Name	**Description**
Close()	Abort download or just release object after download is complete.
GetLocalAbsoluteDirectory()	Returns the local absolute directory where the media files are stored.
GetLocalRelativeDirectory()	Returns the local relative directory (to the downloads root) where the media files are stored.

IDownloadControl Interface Properties

Property Name	**Description**
Status	Download status.

IMediaRecorderConfiguration Interface Properties

Property Name	**Description**
RecordingRootFolder	The local root folder path for recordings.
DownloadsRootFolder	The local root folder path for downloads.

IMonitor Interface Events

All device events carry the device ID as their first parameter to the identify the source device.

Event Name	Description
BatteryStatusChange (devid, flags, voltage, partial, minutes)	New battery status report.
DeviceConnection (devid)	A new device has just connected
DeviceDisconnection (devid)	A device has just disconnected
DeviceStatus (devstatus)	A device has some properties updated, such as its name. This event comes almost immediately after device connection, but may come several times during its connection.
DirectConnection (index)	A new direct connection (client-server). Use this index to access files on the server.
DirectDisconnection (index)	A direct connection has been lost.
FileList (devid, htype, scat, dir, files, freespace)	A new media file list query reply.
InputPortValueChange (devid, index, value)	One of the input ports of a device changed its value.
LocationChange (devid, lat, long, height, dtDateTime)	A device has a new location update. The date/time parameter indicates the UTC time, `DateTime.MinValue` if unknown.
MediaSectionNotFound (devid, htype, scat, reqid)	A media file was not found when cloning a media section.
NewMediaSection (devid, htype, scat, stime, dir, name, ext, reqid)	A new media section was cloned from an existing media file.
OutputPortValueChange (devid, index, value)	One of the output ports of a device changed its value.
SatellitesCount(devid, count)	Number of navigation satellites received by the GNSS receiver.
SignalQuality(devid, signal)	Signal strength measurement, signal value is between 0 and 255.
VideoDetectionState (devid, channel, state)	Camera video signal detection state changed.
VisualMotionDetection (devid, channel, detected)	Visual motion detection state changed.

IMonitor Interface Methods

Method Name	Description
GetChannelRecordingStatus (devid,	Finds live recording status of a channel in a device

Method Name	Description
chtype, chindex, manual, monitor_recid, server_recid, device_recid)	and returns the identifiers of live recordings to allow their stop.
RequestAbstractMediaSection (devid, role, scat, stime, srcid, vidch, audch, reqid, nsec)	Requests cloning a media section from a media file on a server or on a device with a given start time, duration, source device, video and audio channels. It triggers a NewMediaSection or MediaSectionNotFound event if the device and server support this request (5.4.1.1).
RequestMediaDownload (devid, htype, scat, dir, name)	Requests downloading a media file from a device or server, given directory and base name. It returns an IDownloadControl interface for download control and notifications.
RequestRecordingsList (devid, role, scat, dir)	Requests a recordings list from a device or server, given a directory. It triggers a FileList event.
StartRecording (devid, role, vidch, audch)	Starts recording a video and/or an audio channel from a device on the device storage or server storage or monitor storage.
StopRecording (devid, role, recid)	Stops a recording identified by recid on a device, server or monitor.

Management Class Methods

Method Name	Description
Connect(server,user,pass,secure)	Starts a connection to a server, returns an IDisconnectable object reference.
GetDeviceModel(devid)	Returns the model name for a device with a given ID.
GetDeviceName(devid)	Returns the device name for a device with a given ID.
GetInputPortValue(devid,index,value)	Returns input port value for a device with a given ID and port index.
MediaRecorderConfiguration()	Returns IMediaRecorderConfiguration interface for configuring storage management parameters.
Monitor()	Starts monitoring events. Returns IMonitor interface.
Name(app,prod)	Sets application and product name for the monitor application.
ReleaseResources()	Releases resources consumed by the system before quitting.
StartCommunications()	Starts general communications.
StartStorageThread()	Starts storage thread, good for recordings and downloads.
StartWindowsSockets()	Starts general socket communications for the application

	(static).

Initialization Sequence

The following initialization sequence is taken from the MRS_DotNetMonitor C# example project:

```
// Definition
  MRS.Management management = null;
  MRS.IDisconnectable disconnectable = null;
  MRS.IMonitor monitor = null;


// Initialization
   MRS.Management.StartWindowsSockets();
   management = new MRS.Management();
   management.Name("MRS DotNet Monitor Example", "MRS SDK");

   MRS.IMediaRecorderConfiguration mrcRecConfig = management.MediaRecorderConfiguration();
   String
     strUserDir = Environment.GetFolderPath(Environment.SpecialFolder.Personal),
     strMRSDir = Path.Combine(strUserDir, "MRS"),
     strRecDir = Path.Combine(strMRSDir, "Recordings"),
     strDldsDir = Path.Combine(strMRSDir, "Downloads");

   Directory.CreateDirectory(strRecDir);
   Directory.CreateDirectory(strDldsDir);
   mrcRecConfig.RecordingRootFolder = strRecDir;
   mrcRecConfig.DownloadsRootFolder = strDldsDir;

   monitor = management.Monitor();
   management.StartCommunications();
   management.StartStorageThread();


// Connection
  private void Connect()
  {
   if (disconnectable != null)
     disconnectable.Disconnect();
   disconnectable = management.Connect(cfConnect.m_strAddress, cfConnect.m_strUserName,
cfConnect.m_strPassword, cfConnect.m_bSecure);
  }
```

Termination Sequence

The following termination sequence is taken from the MRS_DotNetMonitor C# example project:

```
private void CloseApp()
  {
   if (disconnectable != null)
   {
     disconnectable.Disconnect();
     disconnectable = null;
   }
   management.ReleaseResources();
   management = null;
   monitor = null;
   Environment.Exit(0);
  }
```