



## **PC Best Networks VoIP Recorder V2 SDK Reference**

For version 2.22

## Index:

1	Introduction .....	4
2	SDK Programming Guide .....	5
2.1	Prerequisite for SDK .....	5
2.2	Network Requirements.....	6
2.3	Components and Files .....	8
2.4	Initialization code.....	9
3	API Reference .....	15
3.1	InitCapture .....	15
3.2	FreeCapture .....	16
3.3	GetNICCount .....	16
3.4	GetNICName.....	17
3.5	GetNICDescription .....	17
3.6	SetNIC.....	18
3.7	GetNICIPCount.....	18
3.8	GetNICIP .....	19
3.9	SetNICByIP.....	19
3.10	SetLicenseKey.....	20
3.11	SetLicenseMAC .....	20
3.12	IsLicensed .....	21
3.13	GetLicTo .....	21
3.14	ValidateLicence .....	22
3.15	SetChannelCount.....	22
3.16	SetAudioRootFolder .....	23
3.17	SetAudioFileFormat.....	23
3.18	SetFilter     New in 2.05 .....	24
3.19	SetFilterType     use SetFilter instead.....	24
3.20	SetFilterCount     use SetFilter instead.....	25
3.21	SetFilterItem     use SetFilter instead .....	25
3.22	SetExclude     New in 2.05.....	26
3.23	SetExcludeType     use SetExclude instead.....	27
3.24	SetExcludeCount     use SetExclude instead .....	27
3.25	SetExcludeItem     use SetExclude instead.....	28
3.26	SetMaxWavSeconds.....	29
3.27	SetProtocol .....	29
3.28	SetPort.....	30
3.29	StartCapture.....	30
3.30	StopCapture.....	31
3.31	GetChanxxxxx functions.....	31
3.32	OnCallOffered event .....	32
3.33	OnCallConnected event.....	32
3.34	OnCallEnd event .....	33
3.35	OnCallAudioStream event .....	34
3.36	EnableAudioStreamEvent deprecated, for OCX only .....	35
3.37	GetWavFileName .....	35
3.38	GetXMLFileName .....	36
3.39	SetRecording .....	36
3.40	SetRTPPBXCount .....	37
3.41	SetRTPPBXAddr.....	37
3.42	SetRTPEXtenCount .....	37
3.43	SetRTPEXten .....	38

## PC Best Networks VoIP Recorder V2 SDK Reference

3.44	SetIgnorePossibleSameCall .....	38
3.45	SetNoAudioSeconds .....	39
3.46	SetPCAPFile .....	39
3.47	EnableRTSrv .....	40
3.48	SetMaxDays .....	40
3.49	SetLogLevel .....	41
3.50	SetLogFileName .....	41
3.51	Log .....	41
3.52	SetUsePacketTime .....	42
3.53	SetPauseDTMFKey deprecated .....	42
3.54	SetPauseDTMFStr.....	43
3.55	SetRecordOnlyAfterAnswer .....	43
3.56	StartChanRecording .....	43
3.57	StopChanRecording .....	44
3.58	GetSysProcessedPacketCount .....	44
3.59	SetRecordCallLegs.....	45
3.60	SetRecordStereo .....	45
3.61	PauseRecording .....	46
3.62	SetPauseOption .....	46
3.63	RecordSIPHeadersInXML .....	46
3.64	SetExtenPattern .....	47
3.65	GetChanByCallID .....	47
3.66	GetChanBySIPID .....	48
3.67	GetChanByPChargingVector.....	48
3.68	GetChanByPAssertedIdentity .....	49
3.69	GetChanByRemotePartyID .....	49
3.70	SetDumpFile .....	49
3.71	SetDumpFileMaxSize .....	50
3.72	EnableCiscoBIBRecording .....	51
3.73	SetCiscoBIBRecordingTrunkName .....	51
3.74	SetCiscoBIBIncomingTrunkName.....	52
3.75	SetCiscoBIBOutgoingTrunkName.....	52
3.76	EnableSaveCallPcapFile .....	53
3.77	GetPcapDriverType .....	53
3.78	GetPcapDriverTypeStr .....	54
3.79	EnableAddingDateAndTimeInAudio    new .....	54
3.80	SetAddingDateAndTimeInAudioText    new .....	55
4	Setup Packet Filter .....	55

# **1 Introduction**

PC Best Networks provides NO.1 Windows VoIP development kits to business customers.

The VoIP Recorder SDK(also called SIP Recorder SDK) is one part of VoIP Recorder software, presenting programming interfaces to customers who want to embed a recorder in their VoIP enabled software, like Call Center applications, CRM applications or Call Management applications.

The recorder SDK has 3 interfaces for programmers: C(DLL), .NET(C# or VB.NET) and ActiveX. You can choose the program language according to your preference. There are some samples in SDK package for the interfaces.

Our contact information for support:

Email: [support@pcbest.net](mailto:support@pcbest.net)

## 2 SDK Programming Guide

### 2.1 Prerequisite for SDK

**Please install Npcap or WinPcap first** in order to use SDK. This software is used by SDK to sniff network traffic.

For Windows XP/2003/Vista/2008/Win7/2008R2/Win8 (x86 and x64) users, you can use WinPcap, but For Windows 7, 8 10+ users, please download Npcap driver instead.

The WinPcap project has ceased development and is no longer maintained. We recommend using **Npcap** instead for all Windows version 7+.

Also, please install x86 version of [Visual C++ Redistributable for Visual Studio 2015](#) first on your machine, in order to run the recorder.

*Npcap download:*

<https://nmap.org/npcap/>

The screenshot shows the Npcap website with a sidebar on the left containing links like 'Site News', 'Advertising', and 'About/Contact'. The main content area has a heading 'Loopback Adapter for you.' followed by a list of features:
 

- Loopback Packet Injection:** Npcap is also able to send loopback packets using the Winsock Kernel (WSK) technique. User-level software such as Nping can just send the packets out using Npcap Loopback Adapter just like any other adapter. Npcap then does the magic of removing the packet's Ethernet header and injecting the payload into the Windows TCP/IP stack.
- Libpcap API:** Npcap uses the excellent Libpcap library, enabling Windows applications to use a portable packet capturing API that is also supported on Linux and Mac OS X. While WinPcap was based on LibPcap 1.0.0 from 2009, Npcap includes the latest Libpcap release along with improvements that we also contribute back upstream to Libpcap.
- WinPcap compatibility:** For applications that don't yet make use of Npcap's advanced features, Npcap can be installed in "WinPcap Compatible Mode." This will replace any existing WinPcap installation. If compatibility mode is not selected, Npcap can coexist alongside WinPcap; applications which only know about WinPcap will continue using that, while other applications can choose to use the newer and faster Npcap driver instead.

 Below this, there is a section titled 'Downloading and Installing Npcap Free Edition' which states: 'The free version of Npcap may be used (but not externally redistributed) on up to 5 systems (free license details). It may also be used on unlimited systems where it is only used with Nmap and/or Wireshark. Simply run the executable installer. The full source code for each release is available, and developers can build their apps against the SDK. The improvements for each release are documented in the Npcap Changelog.'
 A red circle highlights the following list:
 

- Npcap 1.00 installer for Windows 7/2008R2, 8/2012, 8.1/2012R2, 10/2016, 2019 (x86 and x64)
- Npcap SDK 1.00 (ZIP).
- Npcap 1.00 debug symbols (ZIP).
- Npcap 1.00 source code (ZIP).

 At the bottom, it says: 'The latest development source is in our Github source repository. Windows XP and earlier are not supported; you can use WinPcap for these versions.'

*WinPcap download:*

<https://www.winpcap.org/install/default.htm>

## 2.2 Network Requirements

You don't need additional hardware if you only want to record calls for the local computer (using VoIP Recorder V2 to record VoIP software on the same computer).

**You must have a network switch that supports bidirectional 'port mirroring' (also called 'port mapping' or the existence of a 'span port'). Most business grade switches do have this feature.**

**Home or Small office can choose:**

**TP-LINK TL-SG105E**

[http://www.tp-link.com/en/products/details/cat-41\\_TL-SG105E.html](http://www.tp-link.com/en/products/details/cat-41_TL-SG105E.html)

**Netgear GS105E**

<https://www.youtube.com/watch?v=kCSRgbEMkWs>

**Or you must have network hub to connect your SIP VoIP devices if you want to record other devices.** Why? Because all Ethernet traffic passes through all hub ports, so the PC runs VoIP Recorder V2 can sniff the network and record other devices.

See articles:

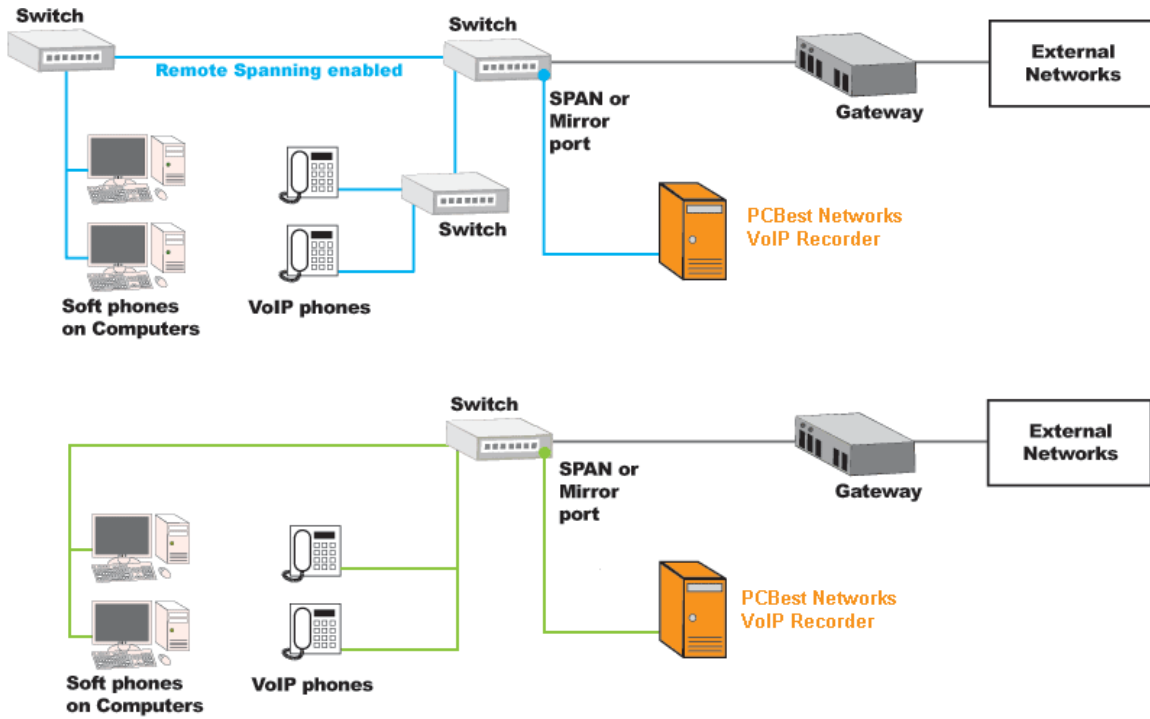
[http://ask-leo.com/whats\\_the\\_difference\\_between\\_a\\_hub\\_a\\_switch\\_and\\_a\\_router.html](http://ask-leo.com/whats_the_difference_between_a_hub_a_switch_and_a_router.html)

[http://www.cisco.com/en/US/products/hw/switches/ps708/products\\_tech\\_note09186a008015c612.shtml#support](http://www.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a008015c612.shtml#support)

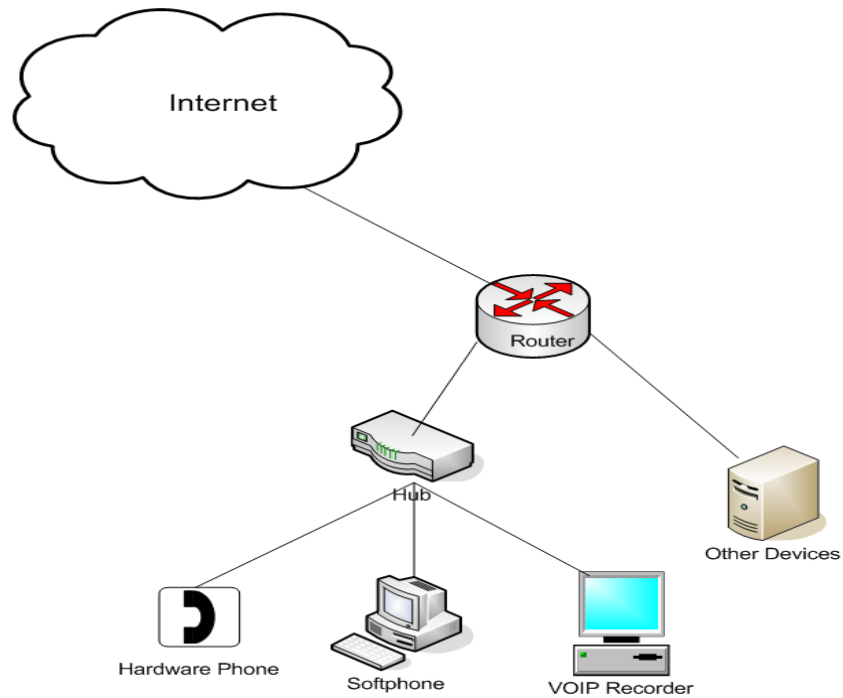
If you have any questions or problems about network hardware issues, please feel free to contact us by <http://www.pcbest.net/contact.php>

### Typical Networks:

## PC Best Networks VoIP Recorder V2 SDK Reference



OR:



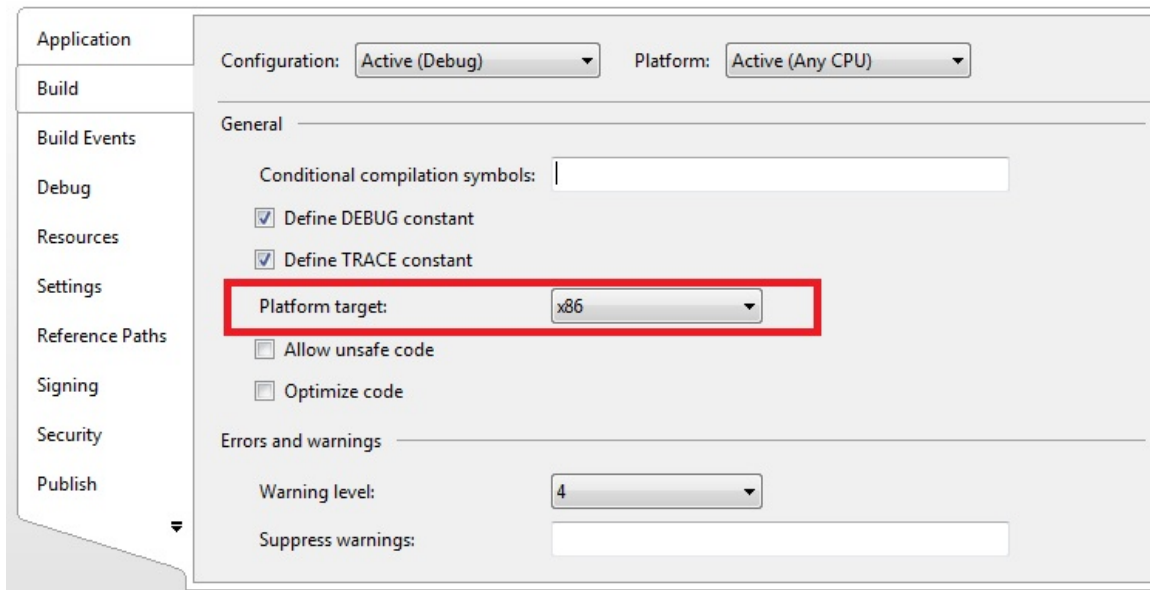
## 2.3 Components and Files

**For C/C++:** VRAPI.dll and all other dlls in SDK/bin folder need to be included into your project.

**For C#, VB.NET:** VRAPI.DLL and VRAPIASM.dll, and all other dlls in SDK/bin folder.

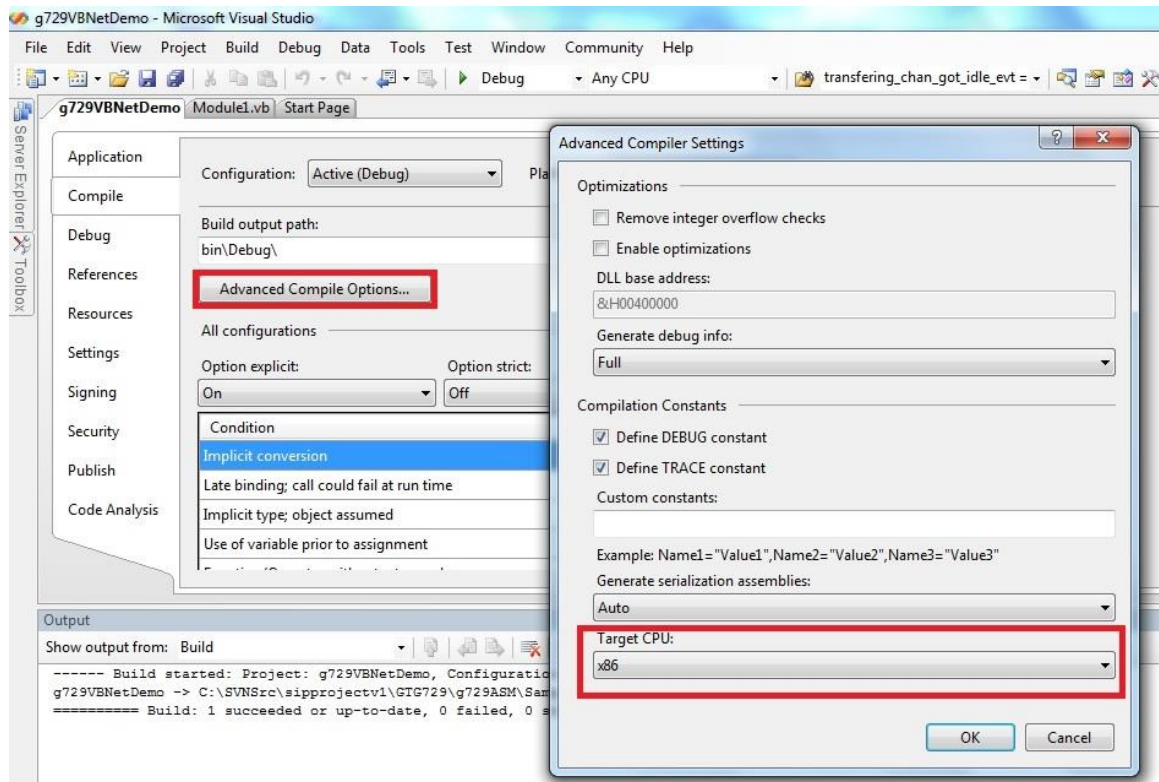
Because VRAPI.DLL and VRAPIASM.dll are 32bit library, you will need to set your .net project target to x86.

For C#:



For VB.NET:





**For ActiveX:** VR2Ctrl.ocx need to be deployed into the destination machine. You may need GTAPI.dll, GTAUDIOCODEC.dll, GTSIPAPI.dll, GTAPIASM.dll with your exe together to the destination machine, for extended feature like Realtime listening, or for PCBest VR2 distribution.

**VR2Ctrl.ocx** is the major ActiveX file for ActiveX interface programming. After you installed VoIP Recorder MSI file, this ocx should be already registered into your Windows system. (You can also register this ActiveX control by cmd: Regsvr32 vr2ctrl.ocx).

## 2.4 Initialization code

C/C++:

```
#include "VRAPI.h"
```

```
VR_InitCapture(0); //Init Driver
```

```
for (int i = 0; i < VR_GetNICCount(); i++)
{
```

```
m_cbNIC.AddString(VR_GetNICDescription(i)); //Use VR_GetNICDescription to get the
NIC name, then set it into GUI List Control
}
```

```
m_btnStart.EnableWindow(TRUE); //GUI control status setting
m_btnStop.EnableWindow(FALSE); //GUI control status setting
```

```
if(VR_GetNICCount() > 0)
    m_cbNIC.SetCurSel(0); //GUI control status setting
else
    m_btnStart.EnableWindow(FALSE); //GUI control status setting
```

```
//Then allow user to choose NIC
VR_SetNIC(m_cbNIC.GetCurSel()); //Set chose NIC
```

```
//If using pcap file instead of NIC
//VR_SetPCAPFile("C:\\temp\\projects\\Recorder2\\PCapSamples\\Skinny\\Cisco-
Skinny-CCM-5.pcap");
```

```
VR_SetChannelCount(MAX_CHAN_NUM); //Set Channel Number
VR_SetAudioRootFolder("c:\\temp\\VR2"); //Set Root folder of recorder wav files
```

```
//log
VR_SetLogLevel(4);
VR_SetLogFileName("C:\\temp\\VR2\\VR2.txt");
```

```
//Protocol: SIP, H323, RTP, MGCP, and SCCP or SKINNY or SKINNY_CCM71 or
SCCP_CCM71, IAX2
VR_SetProtocol("SIP"); //Set Recording Protocol
//VR_SetPort(2000);
```

```
//Set Callback function below
VR_SetCB_Call_Offered(cb_call_offered);
VR_SetCB_Call_Connected(cb_call_connected);
VR_SetCB_Call_Idle(cb_call_idle);
```

```
VR_StartCapture(); //Start Capturing
```

```
Free Driver:
VR_StopCapture();
VR_FreeCapture();
```

**.NET(C# or VB.NET):**

```
VRAPIASM.VRAPIEnv.InitCapture(0); //Init VR2 Driver
```

```
short nicCount = VRAPIASM.VRAPIEnv.GetNICCount(); //Get total NIC count

if (nicCount <= 0)
{
    MessageBox.Show("No Network Interface!");
    btnStartAndStop.Enabled = false;
}
else
{
    for (short i = 0; i < nicCount; i++)
    {
        string s = i.ToString() + " : " + VRAPIASM.VRAPIEnv.GetNICName(i) + "
- " + VRAPIASM.VRAPIEnv.GetNICDescription(i);
        cbNIC.Items.Add(s); //Set NIC names into list for selecting
    }
    cbNIC.SelectedIndex = 0;
}

//After user choose NIC, then
VRAPIASM.VRAPIEnv.SetAudioRootFolder(tbRoot.Text); //Set Root Folder of
recorder file

//<!-- 0 = default(.wav), 1 = mp3, 2 = gsm -->
VRAPIASM.VRAPIEnv.SetAudioFileFormat(1);

//log
VRAPIASM.VRAPIEnv.SetLogFileName("c:\\temp\\VoIP-Recorder-Log.txt");
VRAPIASM.VRAPIEnv.SetLogLevel(4);

//how many channels to open
VRAPIASM.VRAPIEnv.SetChannelCount(Convert.ToInt32(tbChanNum.Text));
//Set Channel Number

//SIP, SCCP or SKINNY, RTP, H323, IAX2, UNISTIM, MGCP
//NOTE: only SIP, SCCP, RTP, and MGCP work so far
VRAPIASM.VRAPIEnv.SetProtocol("SIP"); //Set Protocol

VRAPIASM.VRAPIEnv.SetNIC(Convert.ToInt16(cbNIC.SelectedIndex)); //Set
NIC index

//Set Callback
VRAPIASM.VRAPIEnv.SetCB_Call_Offered(m_pCallOffered);
VRAPIASM.VRAPIEnv.SetCB_Call_Connected(m_pCallConnected);
VRAPIASM.VRAPIEnv.SetCB_Call_Idle(m_pCallIdle);
VRAPIASM.VRAPIEnv.SetCB_Call_DTMF(m_pCallDTMF);
```

## PC Best Networks VoIP Recorder V2 SDK Reference

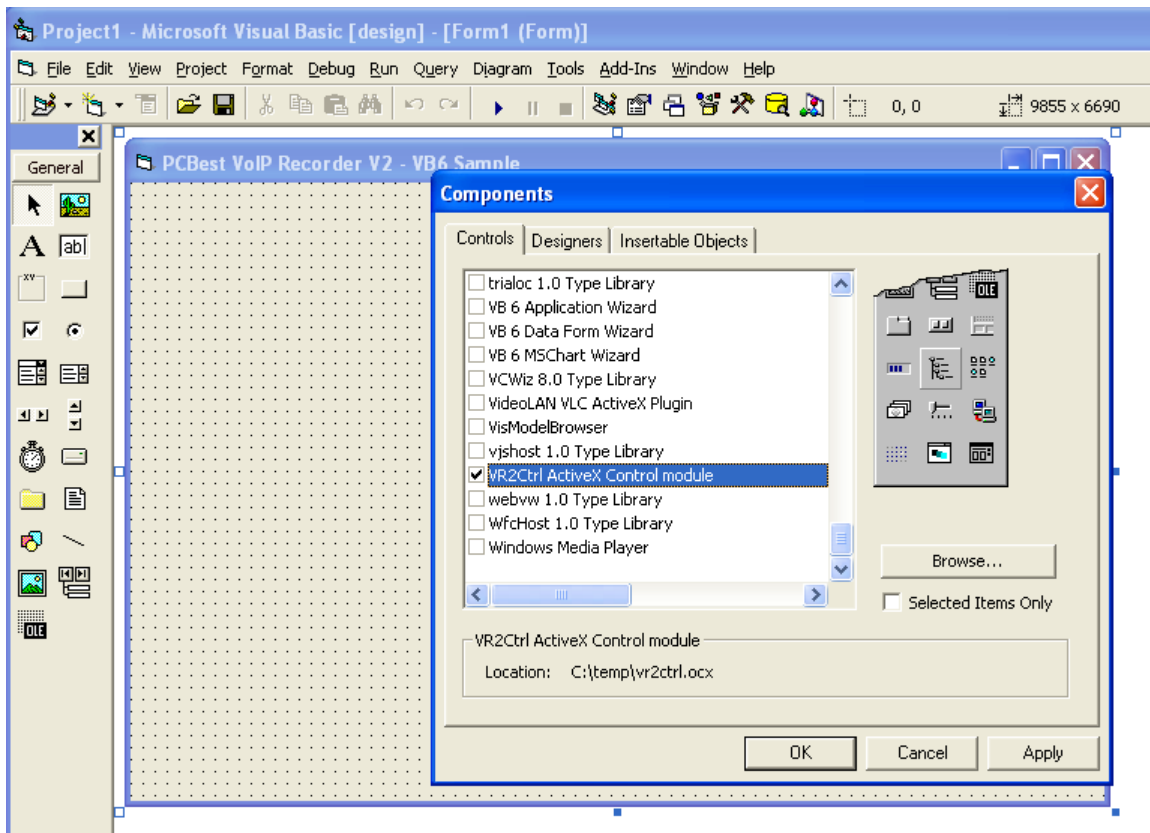
```
VRAPIASM.VRAPIEnv.StartCapture(); //Start Capturing
```

Free VR2 Driver:

```
VRAPIASM.VRAPIEnv.StopCapture();  
VRAPIASM.VRAPIEnv.FreeCapture();
```

### ActiveX:

The following sample code is in vb6. You can find the full code in vb6 folder.  
Drag the ActiveX control into your Form or Dialog.



Assume its name is SIPRecorder1.

- a. In the **Form\_Load** function, you should initialize the control by using the following code

```
Dim i As Integer  
SIPRecorder1.InitCapture 0 //Invoke InitCapture to initialize the ActiveX object  
For i = 1 To SIPRecorder1.GetNICCount
```

```
Combo1.AddItem SIPRecorder1.GetNICDescription(i - 1)
Next i //assume that Combo1 is the ComboBox control to hold NIC list
Combo1.ListIndex = 0 //Set the first one as default selected item
Command1.Enabled = True //"Start Capturing" button is enabled
Command2.Enabled = False //"Stop Capturing" button is disabled
```

```
IsCapturing = False //Boolean to indicate if it is capturing(recording)
```

### **b. In the Form\_Unload function, you free the resource**

```
If IsCapturing Then //if the Boolean indicate it is capturing, then stop it first
    SIPRecorder1.StopCapture
End If
```

```
SIPRecorder1.FreeCapture //use FreeCapture function to free the resource
```

### **c. Event when “Start Capturing” clicked.**

```
ChanNum = Int(Text1.Text) //Get the recording channel numbers from text edit. You
can hardcode it to the actual number according to your license
```

```
SIPRecorder1.SetChannelCount ChanNum //Then set the channel number into object
SIPRecorder1.SetNIC Combo1.ListIndex //Set which NIC you want to work on
SIPRecorder1.SetAudioRootFolder Text2.Text //Set audio root folder to record
SIPRecorder1.SetLogLevel 4 //Set log level to 4. If you don't want the log, set it to 0
SIPRecorder1.SetLogFileName "c:\recorder.txt" //set log file name
VR2Ctrl1.SetProtocol "SIP" //SIP protocol
```

```
SIPRecorder1.StartCapture //Invoke StartCapture function to start
Command1.Enabled = False //Disable “Start Capturing” button
Command2.Enabled = True //Enable “Stop Capturing” button
```

```
IsCapturing = True //change the global boolean to indicate capturing
```

The following code is for GUI grid to show channels

```
MSFlexGrid1.Rows = ChanNum + 1
MSFlexGrid1.Cols = 5
```

```
MSFlexGrid1.Col = 0
For i = 1 To ChanNum
    MSFlexGrid1.Row = i
    MSFlexGrid1.Text = Str(i)
Next i
```

```
MSFlexGrid1.Row = 0
MSFlexGrid1.Col = 1
```

```
MSFlexGrid1.Text = "Caller"  
MSFlexGrid1.ColWidth(1) = 2000  
MSFlexGrid1.Row = 0  
MSFlexGrid1.Col = 2  
MSFlexGrid1.Text = "Callee"  
MSFlexGrid1.ColWidth(2) = 2000  
MSFlexGrid1.Row = 0  
MSFlexGrid1.Col = 3  
MSFlexGrid1.Text = "Status"  
MSFlexGrid1.ColWidth(3) = 2000  
MSFlexGrid1.Row = 0  
MSFlexGrid1.Col = 4  
MSFlexGrid1.Text = "Audio"  
MSFlexGrid1.ColWidth(4) = 2000
```

**d. Event when “Stop Capturing” clicked.**

```
SIPRecorder1.StopCapture //Invoke the StopCapture to stop  
IsCapturing = False //change the boolean
```

```
Command1.Enabled = True //change two buttons status  
Command2.Enabled = False
```

```
'reset SIP recorder1 for next time use  
SIPRecorder1.FreeCapture  
SIPRecorder1.InitCapture
```

**e. Add a timer to trigger event every second**

*This code is for updating the channel status in GUI grid. If you are not showing customers recording status real-time, you don't need this code*

```
Dim i As Integer  
Dim ChanStatus As Integer  
If IsCapturing And ChanNum > 0 Then  
    For i = 1 To ChanNum  
        MSFlexGrid1.Row = i  
        ChanStatus = SIPRecorder1.GetChanStatus(i - 1)  
        If ChanStatus > 0 Then 'not idle  
            MSFlexGrid1.Col = 1  
            MSFlexGrid1.Text = SIPRecorder1.GetChanCallerID(i - 1)  
            MSFlexGrid1.Col = 2  
            MSFlexGrid1.Text = SIPRecorder1.GetChanCalleeID(i - 1)
```

```

MSFlexGrid1.Col = 3
If SIPRecorder1.GetChanStatus(i - 1) = 1 Then
    MSFlexGrid1.Text = "Connecting"
Else
    If SIPRecorder1.GetChanStatus(i - 1) = 2 Then
        MSFlexGrid1.Text = "Connected"
    End If
End If
MSFlexGrid1.Col = 4
MSFlexGrid1.Text = SIPRecorder1.GetChanAudioFileName(i - 1)
Else
    'idle
    MSFlexGrid1.Col = 1
    If MSFlexGrid1.Text <> "" Then 'need to update screen because there is text on
the row
        MSFlexGrid1.Col = 1
        MSFlexGrid1.Text = ""
        MSFlexGrid1.Col = 2
        MSFlexGrid1.Text = ""
        MSFlexGrid1.Col = 3
        MSFlexGrid1.Text = ""
        MSFlexGrid1.Col = 4
        MSFlexGrid1.Text = ""
    End If
End If
Next i
End If

```

## 3 API Reference

### 3.1 InitCapture

**Description:** Initialize the VR2 driver. What it does is it gets the machine's NIC information. You should use this method first before you use other methods of SDK.

**Format:**

C/C++: void VR\_InitCapture()

.NET: void InitCapture ()

OCX: bool InitCapture ()

**Parameters:**

None

**Return:**

None

**Sample code:**

Ocx.InitCapture

## 3.2 FreeCapture

**Description:** Free the resource, corresponding to InitCapture function.

**Format:**

C/C++: void VR\_FreeCapture()

.NET: void FreeCapture ()

OCX: bool FreeCapture ()

**Parameters:**

None

**Return:**

None

**Sample code:**

Ocx.FreeCapture

## 3.3 GetNICCount

**Description:** Get number of NIC(Network Interface Card, also called Ethernet card) of the computer.

**Format:**

C/C++: short VR\_GetNICCount();

.NET: short GetNICCount();

OCX: short GetNICCount();

**Parameters:**

None

**Return:**



The number of NIC.

**Sample code:**

```
int cnt = VR_GetNICCount();
```

### 3.4 GetNICName

**Description:** Get the name of NIC by index.

**Format:**

C/C++: `const char* VR_GetNICName(short idx);`

.NET: `string GetNICName(short idx);`

OCX: `BSTR GetNICName(short idx);`

**Format:**

`string GetNICName(short Index)`

**Parameters:**

Index: NIC index. Based on 0, to GetNICCount()-1

**Return:**

The name of NIC.

**Sample code:**

```
Ocx. GetNICName(0) //Get the name of first NIC
```

### 3.5 GetNICDescription

**Description:** Get the description of NIC by index.

**Format:**

C/C++: `const char* VR_GetNICDescription(short idx);`

.NET: `string GetNICDescription(short idx);`

OCX: `BSTR GetNICDescription(short idx);`

**Parameters:**

Index: NIC index. Based on 0, to GetNICCount()-1

**Return:**

The description of NIC.

**Sample code:**

```
GetNICDescription(0) //Get the description of first NIC
```

## 3.6 SetNIC

**Description:** Set the index of NIC to capture the traffic

**Format:**

C/C++: void VR\_SetNIC(short idx);

.NET: void SetNIC(short idx);

OCX: void SetNIC(short idx);

**Parameters:**

Index: NIC index. Based on 0, to GetNICCount()-1

**Return:**

none

**Sample code:**

```
Ocx.SetNIC(0) //Use the first NIC
```

## 3.7 GetNICIPCount

**Description:** Get how many IP addresses associated with the idx NIC

**Format:**

C/C++: short VR\_GetNICIPCount(short idx);

.NET: short GetNICIPCount(short idx);

OCX: long GetNICIPCount(long idx);

**Parameters:**

Index: NIC index. Based on 0, to GetNICCount()-1

**Return:**

Number of IPs

**Sample code:**

```
int IPCnt = GetNICIPCount(0);
```

## 3.8 GetNICIP

**Description:** Get the IP addresses associated with the idx NIC

**Format:**

C/C++: `const char* VR_GetNICIP(short idxNIC, short idxIP);`

.NET: `string GetNICIP(short idxNIC, short idxIP);`

OCX: `BSTR GetNICIP(long idxNIC, long idxIP);`

**Parameters:**

idxNIC: NIC index. Based on 0, to GetNICCount()-1

idxIP: IP index. Based on 0, to GetNICIPCount(idxNIC)-1

**Return:**

IP address of NIC

**Sample code:**

```
string ipAddr = GetNICIPCount(0, 0);
```

## 3.9 SetNICByIP

**Description:** Set the IP address of NIC to capture the traffic. This function has same effect as SetNIC, but with different parameter.

**Format:**

C/C++: `void VR_SetNICByIP(const char* ipAddr);`

.NET: `void SetNICByIP(string ipAddr);`

OCX: `void SetNICByIP(BSTR ipAddr);`

**Parameters:**

ipAddr: the IP address of Ethernet Card(NIC), like "192.168.1.100".

**Return:**

none

**Sample code:**

```
Ocx. SetNICByIP ("192.168.1.100".) //Use the NIC which has IP address
"192.168.1.100".
```

## 3.10 SetLicenseKey

**Description:** Set the license key

**Format:**

C/C++: void VR\_SetLicenseKey(const char\* s);

.NET: void SetLicenseKey(string s);

OCX: void SetLicenseKey(BSTR s);

**Parameters:**

s: License key you get from PCBest Networks

**Return:**

none

**Sample code:**

```
Ocx.SetLicenseKey("ABCD-EFGH-....")
```

## 3.11 SetLicenseMAC

**Description:** Set the MAC address of NIC which license key should associate with.

**Format:**

C/C++: void VR\_SetLicenseMAC(const char\* s);

.NET: void SetLicenseMAC (string s);

OCX: void SetLicenseMAC (BSTR s);

**Parameters:**

s: MAC address of your wired NIC. If you are using USB key, please set the USB key driver's letter, like "e:", or "f:".

**Return:**

none

**Sample code:**

```
Ocx.SetLicenseKey("98-A1-DC-E2-6B-75");
```

```
USB sample: Ocx.SetLicenseKey("g:");
```

## 3.12 IsLicensed

**Description:** Get if your license is valid. Use this function after StartCapture is invoked.

**Format:**

C/C++: int VR\_IsLicensed();

.NET: int IsLicensed();

OCX: int IsLicensed();

**Parameters:**

none

**Return:**

1: licensed

0: not licensed

**Sample code:**

## 3.13 GetLicTo

**Description:** If your license is valid by calling above IsLicensed function, you can call this function to get license information.

**Format:**

C/C++: const char\* VR\_GetLicTo();

.NET: string GetLicTo();

OCX: BSTR GetLicTo();

**Parameters:**

none

**Return:**

License information.

**Sample code:**

## 3.14 ValidateLicence

**Description:** Call this function to validate a license key for the local machine or for USB key. You can use this function to develop your own license validation process application.

**Format:**

C/C++: `const char* VR_ValidateLicence(const char *sKey, unsigned int numChan, const char* appVersion, const char* sMacAddr);`  
.NET: `string ValidateLicence(string sKey, uint numChan, string appVersion, string sMacAddr);`  
OCX: `BSTR ValidateLicence(BSTR sKey, ULONG numChan, BSTR appVersion, BSTR sMacAddr);`

**Parameters:**

**sKey:** the license key supplied by PC Best Networks

**numChan:** How many channels the key is for, or the application plans to open and use

**appVersion:** leave it blank("")

**sMacAddr:** if you wish the license key is associated to a specifical MAC address of local, please specify. Format is like "98-A1-DC-E2-6B-75". Otherwise leave it blank. If you are using USB key, please set this to USB driver letter, like "e:", or "f:".

**Return:**

License information with the license file path generated.

**Sample code:**

```
ValidateLicence("ABCDE-FGHIJ-KLMNO-PQRST-UVWXY", 4, "", "");
ValidateLicence("ABCDE-FGHIJ-KLMNO-PQRST-UVWXY", 4, "", "g:"); //USB
ValidateLicence("ABCDE-FGHIJ-KLMNO-PQRST-UVWXY", 4, "", "98-A1-DC-E2-6B-75"); //On NIC MAC address 98-A1-DC-E2-6B-75.
```

## 3.15 SetChannelCount

**Description:** Set the number of channels to open for recording

**Format:**

C/C++: `void VR_SetChannelCount(int cnt);`  
.NET: `void SetChannelCount(int cnt);`  
OCX: `void SetChannelCount(long cnt);`

**Parameters:**

cnt: the number of channels

**Return:**

none

**Sample code:**

Ocx. SetChannelCount (4) //4 simultaneous channels

## 3.16 SetAudioRootFolder

**Description:** Set the root folder of recording audio files

**Format:**

C/C++: void VR\_SetAudioRootFolder(const char\* RootFolder);

.NET: void SetAudioRootFolder(string RootFolder);

OCX: void SetAudioRootFolder(BSTR RootFolder);

**Parameters:**

RootFolder: the root folder

**Return:**

none

**Sample code:**

Ocx. SetAudioRootFolder("c:\temp") //c:\temp as recording root folder

## 3.17 SetAudioFileFormat

**Description:** Set recording audio as wav or mp3

**Format:**

C/C++: void VR\_SetAudioFileFormat(int fmt);

.NET: void SetAudioFileFormat(int fmt);

OCX: void SetAudioFileFormat(int fmt);

**Parameters:**

fmt: 0 = default(.wav), 1 = mp3

**Return:**

none

**Sample code:**

```
VRAPIASM.VRAPIEnv.SetAudioFileFormat(1); //Set recording as mp3
```

### 3.18 SetFilter **New in 2.05**

**Description:** Set or remove a filter. This function is developed to replace the following three functions: SetFilterType, SetFilterCount, SetFilterItem.

**Format:**

C/C++: void VR\_SetFilter(const char\* s, int act);

NET: void SetFilter(string s, int act);

OCX: void SetFilter(BSTR s, int act);

**Parameters:**

s == Filter string. Can be IP Address, MAC address, or call id.

MAC address following this format: AA-AA-AA-AA-AA-AA.

act == 1: add, 0:remove

**Return:**

none

**Sample code:**

```
SetFilter ("192.168.1.1", 1) //add IP filter 192.168.1.1
```

```
SetFilter ("A7-B8-9C-66-F3-40", 0) //remove MAC filter A7-B8-9C-66-F3-40
```

```
SetFilter("8396772", 1) //Add call id filter 8396772
```

### 3.19 SetFilterType **use SetFilter instead**

**Description:** Set the filter type if you want filter enabled

**Format:**

C/C++: void VR\_SetFilterType(int FilterType);

.NET: void SetFilterType(int FilterType);

OCX: void SetFilterType(long FilterType);

**Parameters:**

FilterType:

0 == IP Addr,



1 == ID for SIP, please only give the username of SIP address, not full <sip:xxx> string.  
For example, for SIP address <sip:1234@abc.com>, you set 1234 only here.  
You can use \* from any string, and ? for any character. See the sample below.  
2 = MAC address

**Return:**

none

**Sample code:**

```
SetFilterType(1) //set filter by call id(or username)
SetFilterCount(1)
SetFilterItem(0, "980*") //Filter out any calls which number have prefix 980. ONLY
record those calls.
```

## 3.20 SetFilterCount **use SetFilter instead**

**Description:** Set how many filters you want

**Format:**

C/C++: void VR\_SetFilterCount(int FilterNumbers);  
.NET: void SetFilterCount(int FilterNumbers);  
OCX: void SetFilterCount(int FilterNumbers);

**Parameters:**

FilterNumbers: number of filters. Default it is 0, means no filter. Max value is 1024.

**Return:**

none

**Sample code:**

```
Ocx. SetFilterCount (2) //set 2 filters
```

## 3.21 SetFilterItem **use SetFilter instead**

**Description:** Set the filter item

**Format:**

C/C++: void VR\_SetFilterItem(int Index, const char\* Filter);  
.NET: void SetFilterItem(int Index, string Filter);  
OCX: void SetFilterItem(int Index, string Filter);

**Parameters:**

Index: Index of filter, from 0 to SetFilterCount-1.

Filter: filter string

**Return:**

none

**Sample code:**

```
//Samples for IP address filter
ocx.SetFilterType(0)
Ocx.SetFilterCount (2) //set 2 filters
Ocx.SetFilterItem(0, "192.168.1.100");
Ocx.SetFilterItem(1, "192.168.1.101");
```

```
//Samples for call-id filter
ocx.SetFilterType(1)
Ocx.SetFilterCount (2) //set 2 filters
Ocx.SetFilterItem(0, "101");
Ocx.SetFilterItem(1, "102");
```

```
//Samples for MAC
ocx.SetFilterType(2)
Ocx.SetFilterCount (2) //set 2 filters
Ocx.SetFilterItem(0, "09-32-A4-F3-E2-43");
Ocx.SetFilterItem(1, "67-D2-CE-91-5A-B2");
```

## 3.22 SetExclude **New in 2.05**

**Description:** Set or remove an exclude. This function is developed to replace the following three functions: SetExcludeType, SetExcludeCount, SetExcludeItem.

**Format:**

C/C++: void VR\_SetExclude(const char\* s, int act);

NET: void SetExclude(string s, int act);

OCX: void SetExclude(BSTR s, int act);

**Parameters:**

s == Exclude string. Can be IP Address, MAC address, or call id.

MAC address following this format: AA-AA-AA-AA-AA-AA.

act == 1: add, 0:remove

**Return:**

none

**Sample code:**

```
SetExclude ("192.168.1.1", 1) //Excluding IP Exclude 192.168.1.1
SetExclude ("A7-B8-9C-66-F3-40", 0) //Remove excluding MAC address A7-B8-9C-66-F3-40
SetExclude ("8396772", 1) //Excluding call id 8396772
```

### 3.23 SetExcludeType **use SetExclude instead**

**Description:** Set the exclude type if you want exclude enabled. Exclude list allows you set a list of addresses you **don't** want to record.

**Format:**

C/C++: void VR\_SetExcludeType(int ExcludeType);  
.NET: void SetExcludeType(int ExcludeType);  
OCX: void SetExcludeType(long ExcludeType);

**Parameters:**

ExcludeType:

0 == IP Addr,

1 == ID for SIP, please only give the username of SIP address, not full <sip:xxx> string.  
For example, for SIP address <sip:1234@abc.com>, you set 1234 only here.

You can use \* from any string, and ? for any character. See the sample below.

2 = MAC address

**Return:**

none

**Sample code:**

```
SetExcludeType (1) //set filter by call id(or username)
SetExcludeCount(1)
SetExcludeItem(0, "980*") //Exclude any calls which number have prefix 980. DO NOT
record those calls.
```

### 3.24 SetExcludeCount **use SetExclude instead**

**Description:** Set how many filters you want

**Format:**

C/C++: void VR\_SetExcludeCount (int ExcludeNumbers);  
.NET: void SetExcludeCount (int ExcludeNumbers);  
OCX: void SetExcludeCount (int ExcludeNumbers);

**Parameters:**

ExcludeNumbers: number of Excludes. Default it is 0, means no filter. Max value is 1024.

**Return:**

none

**Sample code:**

```
SetExcludeCount (2) //set 2 filters
```

### 3.25 SetExcludeItem **use SetExclude instead**

**Description:** Set the exclude item

**Format:**

C/C++: void VR\_SetExcludeItem (int Index, const char\* Excluder);  
.NET: void SetExcludeItem (int Index, string Excluder);  
OCX: void SetExcludeItem (int Index, string Excluder);

**Parameters:**

Index: Index of filter, from 0 to SetExcludeCount-1.

Excluder: exclude string

**Return:**

none

**Sample code:**

```
//Samples for IP address filter
```

```
ocx.SetExcludeType(0)
```

```
Ocx.SetExcludeCount (2) //set 2 filters
```

```
Ocx. SetExcludeItem (0, "192.168.1.100");
```

```
Ocx. SetExcludeItem (1, "192.168.1.101");
```

```
//Samples for call-id filter
```

```
ocx. SetExcludeType (1)
```

```
Ocx. SetExcludeCount (2) //set 2 filters
```

```
Ocx. SetExcludeItem (0, "101");
```

```
Ocx. SetExcludeItem (1, "102");
```

```
//Samples for MAC
```

```
ocx. SetExcludeType (2)
Ocx. SetExcludeCount (2) //set 2 filters
Ocx. SetExcludeItem (0, "09-32-A4-F3-E2-43");
Ocx. SetExcludeItem (1, "67-D2-CE-91-5A-B2");
```

## 3.26 SetMaxWavSeconds

**Description:** Set the max length of wav file in seconds

**Format:**

```
C/C++: void VR_SetMaxWavSeconds(int Seconds);
.NET: void SetMaxWavSeconds(int Seconds);
OCX: void SetMaxWavSeconds(int Seconds);
```

**Parameters:**

Seconds: in seconds

**Return:**

none

**Sample code:**

## 3.27 SetProtocol

**Description:** Set the VoIP protocol type to record.

**Format:**

```
C/C++: void VR_SetProtocol(const char* sProtocolName);
.NET: void SetProtocol(string sProtocolName);
OCX: void SetProtocol(BSTR sProtocolName);
```

**Parameters:**

sProtocolName: Can be SIP, SCCP or SKINNY or SKINNY\_CCM7 or SCCP\_CCM7, RTP, H323, IAX2, UNISTIM, MGCP.  
It supports multiple protocol types, like you can set 'SIP|SCCP|MGCP|H323'

**Return:**

none

**Sample code:**

## 3.28 SetPort

**Description:** Set the port number that working with the protocol. In the most of cases, you don't need to use this function, as VoIP Recorder engine can distinguish it. Just in case VoIP recorder cannot detect the call, you can use this function to set substandard protocol port.

**Format:**

C/C++: void VR\_SetPort(unsigned short PortNum);

.NET: void SetPort(ushort PortNum);

OCX: void SetPort(long PortNum);

**Parameters:**

PortNum 0 - 65535

**Return:**

none

**Sample code:**

## 3.29 StartCapture

**Description:** Start to capture network traffic(start recording).

**Format:**

C/C++: int VR\_StartCapture();

.NET: int StartCapture();

OCX: int StartCapture();

**Parameters:**

none

**Return:**

1: success

0: failed

**Sample code:**

ocx.StartCapture

### 3.30 StopCapture

**Description:** Stop capturing(recording)

**Format:**

C/C++: int VR\_StopCapture ();

.NET: int StopCapture ();

OCX: int StopCapture ();

**Parameters:**

1: success

0: failed

**Return:**

boolean

**Sample code:**

ocx.StopCapture

### 3.31 GetChanxxxx functions

**Description:** Get the channel status functions

**Format:**

int **GetChanStatus**(long ChanIndex) //return 0=idle, 1=connecting, 2=connected

string **GetChanCallerID**(long ChanIndex) //return caller's user id

string **GetChanCallerIP**(long ChanIndex) //return caller's ip

int **GetChanCallerPort**(long ChanIndex) //return caller's port

string **GetChanCalleeID**(long ChanIndex) //return callee's user id

string **GetChanCalleeIP**(long ChanIndex) //return callee's ip

int **GetChanCalleePort**(long ChanIndex) //return callee's port

string **GetChanUniqueID**(long ChanIndex) //return channel's unique id for this call

string **GetChanAudioFileName**(long ChanIndex) //return audio file name for this call

int **GetChanAudioFileNumber**(long ChanIndex) //return the number of audio files for this call

int **GetChannelCount**() //return the total number of channels which have been opened

int **GetTotalCallCount**(); //how many calls in total

int **GetSuccessfulCallCount**(); //how many calls got connected

**Parameters:**

ChanIndex: Channel index, based on 0.

**Return:**

**Sample code:**

## 3.32 OnCallOffered event

**Description:** A new call detected, but it is not connected yet

**Format:**

```
typedef void (VR_CALLBACK *VR_CB_call_offered)(int ChanIndex, const
char* CallerIP, const char* CallerID, const char* CalleeIP, const char*
CalleeID, unsigned int CallTime, const char* UniqueID, const char*
AudioFile, int CallDir);
```

C/C++: void VR\_SetCB\_Call\_Offered(VR\_CB\_call\_offered p);

.NET: void SetCB\_Call\_Offered(VR\_CB\_call\_offered p);

```
OCX: void OnCallOffered(int ChanIndex, const char* CallerIP, const char*
CallerID, const char* CalleeIP, const char* CalleeID, unsigned int
CallTime, const char* UniqueID, const char* AudioFile, int CallDir);
```

**Parameters:**

ChanIndex: Channel index, based on 0.

CallerIP: IP address of caller

CallerID: ID of caller

CalleeIP: IP address of callee

CalleeID: ID of callee

CallTime: Time of the call

UniqueID: UniqueID for the call

AudioFile: audio file name to record the call

CallDir: 0 = Inbound call, 1 = Outbound call

**Return:**

**Sample code:**

## 3.33 OnCallConnected event

**Description:** A call is connected



## Format:

```
typedef void (VR_CALLBACK *VR_CB_call_connected)(int ChanIndex, const char* CallerIP, const char* CallerID, const char* CalleeIP, const char* CalleeID, unsigned int InitTime, unsigned int ConnectTime, const char* UniqueID, const char* AudioFile, int CallDir);
```

C/C++: void VR\_SetCB\_Call\_Connected(VR\_CB\_call\_connected p);

.NET: void SetCB\_Call\_Connected(VR\_CB\_call\_connected p);

OCX: void OnCallConnected(int ChanIndex, const char\* CallerIP, const char\* CallerID, const char\* CalleeIP, const char\* CalleeID, unsigned int InitTime, unsigned int ConnectTime, const char\* UniqueID, const char\* AudioFile, int CallDir);

## Parameters:

ChanIndex: Channel index, based on 0.

CallerIP: IP address of caller

CallerID: ID of caller

CalleeIP: IP address of callee

CalleeID: ID of callee

InitTime: Time when the call starts

ConnectTime: Time when the call get connected

UniqueID: UniqueID for the call

AudioFile: audio file name to record the call

CallDir: 0 = Inbound call, 1 = Outbound call

## Return:

## Sample code:

## 3.34 OnCallEnd event

**Description:** A call ends.

## Format:

```
typedef void (VR_CALLBACK *VR_CB_call_idle)(int ChanIndex, const char* CallerIP, const char* CallerID, const char* CalleeIP, const char* CalleeID, unsigned int InitTime, unsigned int ConnectTime, unsigned int EndTime, const char* UniqueID, const char* AudioFile, int AudioFileNum, int Reason, int CallDir, const char* sDTMF, int Codec);
```

C/C++: void VR\_SetCB\_Call\_Idle(VR\_CB\_call\_idle p);

.NET: void SetCB\_Call\_Idle(VR\_CB\_call\_idle p);

OCX: void OnCallIdle(int ChanIndex, const char\* CallerIP, const char\* CallerID, const char\* CalleeIP, const char\* CalleeID, unsigned int

```
InitTime, unsigned int ConnectTime, unsigned int EndTime, const char*
UniqueID, const char* AudioFile, int AudioFileNum, int Reason, int
CallDir, const char* sDTMF, int Codec);
```

### Parameters:

ChanIndex: Channel index, based on 0.

CallerIP: IP address of caller

CallerID: ID of caller

CalleeIP: IP address of callee

CalleeID: ID of callee

InitTime: Time when the call starts

ConnectTime: Time when the call get connected

EndTime: Time of call ends.

UniqueID: UniqueID for the call

AudioFile: audio file name to record the call

AudioFileNum: the number of audio files

Reason: Reason call ends. 0 = normal. 401 or 407 = unauthorized

CallDir: 0 = Inbound call, 1 = Outbound call

sDTMF: DTMF key pressed

Codec: Audio Codec used. Caller Codec: Codec&0xFF, Callee Codec:  
(Codec&0xFF00)>>8

**Return:** none

## 3.35 OnCallAudioStream event

**Description:** Event to retrieve the audio data of a call. **Note: you must call EnableAudioStreamEvent to enable this feature, if you are using OCX interface.**

### Format:

```
typedef void (VR_CALLBACK *VR_CB_call_audio_buffer)(int ChanIndex,
const char* UniqueID, int Direction, const char* Buff, int BuffSize);
```

C/C++: void VR\_SetCB\_Call\_Audio\_Buffer(VR\_CB\_call\_audio\_buffer p);

.NET: void SetCB\_Call\_Audio\_Buffer(VR\_CB\_call\_audio\_buffer p);

OCX: void OnCallAudioStream(int ChanIndex, const char\* UniqueID, int  
Direction, const char\* Buff, int BuffSize);

### Parameters:

ChanIndex: Channel index, based on 0.

UniqueID: Call unique ID

Direction: 0 = caller to callee, 1 = callee to caller

Buff: Audio buffer address

BuffSize: Audio buffer size

**Return:** none

### 3.36 EnableAudioStreamEvent deprecated, for OCX only

**Description:** Enable OnCallAudioStream event.

**Format:**

C/C++: use VR\_SetCB\_Call\_Audio\_Buffer instead

.NET: use SetCB\_Call\_Audio\_Buffer instead

OCX: void EnableAudioStreamEvent(LONG Enabled)

**Parameters:**

Enabled: 1 = enable, 0 = disable

**Return:** none

### 3.37 GetWavFileName

**Description:** Callback function allows you to set your own wav file full path.

**Format:**

```
typedef const char* (VR_CALLBACK *VR_CB_get_wav_file_name1)(int  
ChanIndex, const char* UniqueID, const char* CallerID, const char*  
CalleeID);
```

C/C++: void VR\_SetCB\_GetWAVFileName1(VR\_CB\_get\_wav\_file\_name1 p)

.NET: void SetCB\_GetWAVFileName(VR\_CB\_get\_wav\_file\_name p)

**Parameters:**

ChanIndex: channel number

UniqueID: VoIP SDK generated unique string for the call

CallerID: caller id

CalleeID: called id

**Return:**

The full path of wav file. If it is ""(null string), SDK won't generate wav file.

### 3.38 GetXMLFileName

**Description:** Callback function allows you to set your own xml file full path.

**Format:**

```
typedef const char* (VR_CALLBACK *VR_CB_get_xml_file_name)(int
ChanIndex, const char* UniqueID, const char* CallerID, const char*
CalleeID);
```

C/C++: void VR\_SetCB\_GetXMLFileName(VR\_CB\_get\_xml\_file\_name p);

.NET: void SetCB\_GetXMLFileName(VR\_CB\_get\_xml\_file\_name p);

**Parameters:**

ChanIndex: channel number

UniqueID: VoIP SDK generated unique string for the call

CallerID: caller id

CalleeID: called id

**Return:**

The full path of xml file. If it is ""(null string), SDK won't generate xml file.

### 3.39 SetRecording

**Description:** Set if you want to record the call. Default it is enabled. If you set false, the VoIP Recorder will only save xml call info file, but not record conversation into wav files. You can use StartRecording, StopRecording method to start and stop recording on a channel manually.

**Format:**

C/C++: void VR\_SetRecording(int EnableRecording);

.NET: int SetRecording(int EnableRecording);

OCX: int SetRecording(int EnableRecording);

**Parameters:**

EnableRecording: 1 = true(enabled), default value      0 = false

**Return:** none

### 3.40 SetRTPPBXCount

**Description:** If recording protocol is RTP, this function is used to tell how many PBX IP addresses. Call this function first, then call VR\_SetRTPPBXAddr to set each PBX's IP address.

**Format:**

C/C++: void VR\_SetRTPPBXCount(int cnt);

.NET: void SetRTPPBXCount(int cnt);

OCX: void SetRTPPBXCount(long cnt);

**Parameters:**

cnt: Count of IP PBX.

**Return:** none

### 3.41 SetRTPPBXAddr

**Description:** Use this function to set each VoIP PBX's IP address when working on RTP recording protocol.

**Format:**

C/C++: void VR\_SetRTPPBXAddr(int idx, const char\* ip);

.NET: void SetRTPPBXAddr(int idx, string ip);

OCX: void SetRTPPBXAddr(int idx, BSTR ip);

**Parameters:**

idx: index of VoIP PBX, from 0 - SetRTPPBXCount(int cnt)-1.

PBXAddr: IP address of VoIP PBX.

### 3.42 SetRTPEXtenCount

**Description:** Set extension count which work with VoIP PBX by using recording model RTP.

**Format:**

C/C++: void VR\_SetRTPEXtenCount(int cnt)

.NET: void SetRTPEXtenCount(int cnt)

OCX: void SetRTPEXtenCount(int cnt)

**Parameters:**

cnt: the total number of RTP extensions. Max value is 1024.

**Return:** none

### 3.43 SetRTPEXten

**Description:** Map RTP extension's IP address with extension number, and user name.

**Format:**

C/C++: void VR\_SetRTPEXten(int Index, const char\* Name, const char\* ExtenNumber, const char\* IPAddr)

.NET: void SetRTPEXten(int Index, string Name, string ExtenNumber, string IPAddr)

OCX: void SetRTPEXten(int Index, BSTR Name, BSTR ExtenNumber, BSTR IPAddr)

**Parameters:**

Index: index of extension, from 0.

Name: extension's name, like "Bob", "Mike".

ExtenNumber: extension's phone number, like "101", "205".

IPAddr: IP address of this extension.

**Return:** none

### 3.44 SetIgnorePossibleSameCall

**Description:** Set if recorder ignores possible same call. In some IP PBX environments, one call may have two legs. For example, an incoming call reached PBX, then PBX transferred the call to an extension. Recorder will try to recognize this call by comparing two legs caller and called ID. If the IDs are the same, then the recorder will consider it as one call if this feature is enabled.

**Format:**

C/C++: void VR\_SetIgnorePossibleSameCall(int Enabled)

.NET: void SetIgnorePossibleSameCall(int Enabled)

OCX: void SetIgnorePossibleSameCall(long Enabled)

**Parameters:**

Enabled: 0 = false(default value), 1 = true

**Return:** none

## 3.45 SetNoAudioSeconds

**Description:** Set if VoIP recorder will discontinue recording if there is no RTP audio data for the specific maximum seconds.

**Format:**

C/C++: void VR\_SetNoAudioSeconds(int MaxSeconds)

.NET: void SetNoAudioSeconds(int MaxSeconds)

OCX: void SetNoAudioSeconds(long MaxSeconds)

**Parameters:**

MaxSeconds: 0 = this feature is off (default value), > 0 the maximum seconds if there is no RTP audio data

**Return:** none

## 3.46 SetPCAPFile

**Description:** Set a Wireshark .pcap file as input NIC device. VoIP recorder will read captured frames from .pcap file instead of from network interface card.

**Format:**

C/C++: void VR\_SetPCAPFile(const char\* PCapFileName)

.NET: void SetPCAPFile(string PCapFileName)

OCX: void SetPCAPFile(BSTR PCapFileName)

**Parameters:**

PCapFileName: The full path of .pcap file.

**Return:** none

### 3.47 EnableRTSrv

**Description:** If enable realtime audio listening server. Defaultly it is not enabled. If you enable this option, you must edit VR2RTSrv.ini in the same folder. Recorder will setup a SIP server, and listening on the SIP port. Any call to the Recorder's SIP server, and the To address is the channel number it want to listen at. For example, if the remote user want to listen to the conversation on the first channel, then it should call to <sip:1@recorder's-ip>.

VR2 recorder default uses the 'VR2RTSrv.ini' file for configuration of the SIP server. You can find this file in SDK\bin folder. Please ship this file in the same folder of your exe. In the most of cases, you won't need to change this file, but if you do need the SIP server in recorder to work on a different port, please change this line:

```
#SIP Port, default 5060
gtsrv.sip.ip.port = 5060
```

If you change the default SIP port, on your SIP softphone, you will need to include the port as well. For example, in above case, please change the SIP address to <sip:1@recorder's-ip:**5070**> where 5070 is the new SIP port you set in INI file.

**Format:**

C/C++: void VR\_EnableRTSrv(int Enabled)

.NET: void EnableRTSrv(int Enabled)

OCX: void EnableRTSrv(LONG Enabled)

**Parameters:**

Enabled: 1 = enable, 0 = disable. Defaultly it is disabled.

**Return:** none

### 3.48 SetMaxDays

**Description:** Set how many days of audio files saving in audio root folder. Default it is 0 days, means no clean up. If you want VoIP recorder clean up automatically, and delete wav files older than a specific days, please use this function to set.

**Format:**

C/C++: void VR\_SetMaxDays(int days)

.NET: void SetMaxDays(int days)

OCX: void SetMaxDays(LONG days)

**Parameters:**

days: the number of days.



**Return:** none

## 3.49 SetLogLevel

**Description:** Set SDK log level.

**Format:**

C/C++: void VR\_SetLogLevel(short Level)

.NET: void SetLogLevel(short Level)

OCX: void SetLogLevel(short Level)

**Parameters:**

Level: 0 = disabled, 1 = error, 2 = warning, 3 = debug, 4 = info, 5 = full

**Return:** none

## 3.50 SetLogFileName

**Description:** Set SDK log file name.

**Format:**

C/C++: void VR\_SetLogFileName(const char\* fn)

.NET: void SetLogFileName(string fn)

OCX: void SetLogFileName(BSTR fn)

**Parameters:**

fn: log file name of full path.

**Return:** none

## 3.51 Log

**Description:** Log a trace into SDK log file.

**Format:**

C/C++: void VR\_Log(int LogLevel, const char\* LogInfo)

.NET: void Log(int LogLevel, string LogInfo)

OCX: void Log(int LogLevel, BSTR LogInfo)

**Parameters:**

LogLevel: Log level of trace info, 1 = error, 2 = warning, 3 = debug, 4 = info, 5 = full

LogInfo: text of trace

**Return:** none

## 3.52 SetUsePacketTime

**Description:** if using packet time instead of local machine's time when writing call info. Default it is false(not use packet time, use local time).

**Format:**

C/C++: void VR\_SetUsePacketTime(int b);

.NET: void SetUsePacketTime(int b);

OCX: void SetUsePacketTime(long b);

**Parameters:**

b: 1 = use packet time 0 = use local time(default value)

**Return:**

## 3.53 SetPauseDTMFKey **deprecated**

**Description:** Pause/Resume recording when receiving a DTMF key happened during a call

**Format:**

C/C++: void VR\_API VR\_SetPauseDTMFKey(char c);

.NET: void SetPauseDTMFKey(char c);

OCX: void SetPauseDTMFKey(char c);

**Parameters:**

c: DTMF character. Like '\*', '#', '3',..... Set it to space(' ') to disable this feature. Space(' ') is also default.

**Return:**

null

### 3.54 SetPauseDTMFStr

**Description:** Pause/Resume recording when receiving a DTMF string during a call

**Format:**

C/C++: void VR\_API VR\_SetPauseDTMFStr(const char\* c);

.NET: void SetPauseDTMFKey(string c);

OCX: void SetPauseDTMFKey(BSTR c);

**Parameters:**

c: DTMF string. Like '123\*', '456#', '3'. Set it to empty string("") to disable this feature. empty string("") is also default.

**Return:**

null

### 3.55 SetRecordOnlyAfterAnswer

**Description:** For SIP recorder, if only record after call is connected. Set this to true, it will not record early media before call is answered, like ring tone,....

**Format:**

C/C++: void VR\_API VR\_SetRecordOnlyAfterAnswer (int b);

.NET: void SetRecordOnlyAfterAnswer (bool b);

OCX: void SetRecordOnlyAfterAnswer (long b);

**Parameters:**

b: 1 = true, 0 = false

**Return:**

null

### 3.56 StartChanRecording

**Description:** This function provides a way for application to start recording on a channel.

Note: in default, recorder SDK will automatically record the call on the channel. If you would like to start recording by your own, you can call SetRecording(false) function first

after calling StartCapturing. Then the recording won't start until you call this StartChanRecording function.

**Format:**

C/C++: void VR\_API VR\_StartChanRecording (int ch, const char\* audio\_filename);

.NET: void StartChanRecording (int ch, string audio\_filename);

OCX: void StartChanRecording (long ch, BSTR audio\_filename);

**Parameters:**

ch: channel index

**Return:**

null

**Sample:**

```
void OnCallOffered(int ChanIndex, String CallerIP, String CallerID,
String CalleeIP, String CalleeID, unsigned int CallTime, String
UniqueID, String AudioFile, int CallDir)
{
    StartChanRecording(ChanIndex, "c:\\recordingfolder\\abc.wav");
}
```

### 3.57 StopChanRecording

**Description:** This function provides a way for application to stop recording on a channel.

**Format:**

C/C++: void VR\_API VR\_StopChanRecording (int ch);

.NET: void StopChanRecording (int ch);

OCX: void StopChanRecording (long ch);

**Parameters:**

ch: channel index

**Return:**

null

### 3.58 GetSysProcessedPacketCount

**Description:** Call this function periodly to check if the recorder core engine is still processing.

**Format:**

C/C++: unsigned int VR\_API VR\_GetSysProcessedPacketCount();

.NET: uint GetSysProcessedPacketCount();  
OCX: ulong GetSysProcessedPacketCount();

**Parameters:**

null

**Return:**

null

## 3.59 SetRecordCallLegs

**Description:** If create separated wav files for each leg of the call.

**Format:**

C/C++: void VR\_API VR\_SetRecordCallLegs(int b);  
.NET: void SetRecordCallLegs(int b);  
OCX: void SetRecordCallLegs(int b);

**Parameters:**

b: 1 = true(save each leg's audio as a new wav file), 0 = false

**Return:**

number of packets to be processed.

## 3.60 SetRecordStereo

**Description:** Set if the recording wav file is stereo. (Left channel is caller and right channel is callee). Currently only support stereo in WAV file format.

**Format:**

C/C++: void VR\_API VR\_SetRecordStereo(int b);  
.NET: void SetRecordStereo(int b);  
OCX: void SetRecordStereo(int b);

**Parameters:**

b: 1 = true(stereo), 0 = false(mono, default)

**Return:**

### 3.61 PauseRecording

**Description:** Pause recording temporarily on a channel

**Format:**

C/C++: void VR\_API VR\_PauseRecording(int ch, int pause);

.NET: void VR\_PauseRecording(int ch, int pause);

OCX: void PauseRecording(int ch, int pause);

**Parameters:**

ch: channel index

pause: 1 = pause, 0 = resume

**Return:**

null

### 3.62 SetPauseOption

**Description:** Set pause options. Call this function after StartCapture function.

**Format:**

C/C++: void VR\_SetPauseOption(int ch, int opt)

.NET: void SetPauseOption(int ch, int opt)

OCX: void SetPauseOption(LONG ch, LONG opt)

**Parameters:**

ch: channel index

opt:

0 = not insert blank/silence audio in wav for paused period, not auto-pause(pause the call when it starts)

1 = insert blank/silence audio in wav for paused period

2 = auto-pause + not insert blank/silence audio in wav for paused period

3 = auto-pause + insert blank/silence audio in wav for paused period

**Return:**

null

### 3.63 RecordSIPHeadersInXML

**Description:** Set the extra SIP headers required to write into XML call file

**Format:**

C/C++: void VR\_API VR\_RecordSIPHeadersInXML(const char\* headers);

.NET: void VR\_RecordSIPHeadersInXML(string headers);

OCX: void RecordSIPHeadersInXML(BSTR headers);

**Parameters:**

**headers:** SIP headers. Can set multiple headers and seperated by ';'. Sample: "From;To;P-Asserted-Identity;P-Charging-Vector".

Full SIP Headers Supported:

From;To;Call-Id;Contact;P-Asserted-Identity;P-Charging-Vector;Remote-Party-ID;Reason;Date;User-Agent;Accept-Language;Refer-To

**Return:**

null

## 3.64 SetExtenPattern

**Description:** Set pattern for extension numbers so recorder can know which phone number is extension numbers.

**Format:**

C/C++: void VR\_API VR\_SetExtenPattern(const char\* patterns);

.NET: void VR\_SetExtenPattern(string patterns);

OCX: void SetExtenPattern(BSTR patterns);

**Parameters:**

patterns: seperated by ';', \* for any length of string, and ? for one digit.

Sample: 1??? means 4 characters, and begining with 1.

Sample: 9\* means starting with 9, any length.

**Return:**

null

## 3.65 GetChanByCallID

**Description:** Get channel id by VR2 unique call id.

**Format:**

C/C++: int VR\_API VR\_GetChanByCallID(const char\* call\_id);

.NET: int VR\_GetChanByCallID(string call\_id);  
OCX: long GetChanByCallID(BSTR call\_id);

**Parameters:**

call\_id: VR2 unique call id.

**Return:**

channel id, from 0. if not found, return -1.

## 3.66 GetChanBySIPID

**Description:** Get channel id by SIP Call-ID. If you have another application got the SIP Call-ID header from sip calls, then it can notify to record about if this call should record or not.

**Format:**

C/C++: int VR\_API VR\_GetChanBySIPID (const char\* sip\_id);  
.NET: int VR\_GetChanBySIPID(string sip\_id);  
OCX: long GetChanBySIPID(BSTR sip\_id);

**Parameters:**

sip\_id: sip call id in Call-Id header.

**Return:**

channel id, from 0. if not found, return -1.

## 3.67 GetChanByPChargingVector

**Description:** Get channel id by SIP P-Charging-Vector.

**Format:**

C/C++: int VR\_API VR\_GetChanByPChargingVector (const char\* s);  
.NET: int VR\_GetChanByPChargingVector(string s);  
OCX: long GetChanByPChargingVector(BSTR s);

**Parameters:**

s: P-Charging-Vector value.



**Return:**

channel id, from 0. if not found, return -1.

### 3.68 GetChanByPAssertedIdentity

**Description:** Get channel id by SIP P-Asserted-Identity

**Format:**

C/C++: int VR\_API VR\_GetChanByPAssertedIdentity (const char\* s);

.NET: int VR\_GetChanByPAssertedIdentity(string s);

OCX: long GetChanByPAssertedIdentity (BSTR s);

**Parameters:**

s: P-Asserted-Identity value.

**Return:**

channel id, from 0. if not found, return -1.

### 3.69 GetChanByRemotePartyID

**Description:** Get channel id by SIP Remote-Party-ID

**Format:**

C/C++: int VR\_API VR\_GetChanByRemotePartyID(const char\* s);

.NET: int VR\_GetChanByRemotePartyID(string s);

OCX: long GetChanByRemotePartyID(BSTR s);

**Parameters:**

s: Remote-Party-ID value.

**Return:**

channel id, from 0. if not found, return -1.

### 3.70 SetDumpFile

**Description:** Use this function to switch the recorder to dump mode. It will save all traffic into the dump file you specified. It also works with the filter function, like SetFilter or SetExclude to filter specific packets. The other functions are still same to use, in order to start recording. Like call SetNIC to specify which NIC you would like to record, call StartCapture to start capture. But in this mode, recorder won't record the call at all. The filter method in chapter 4 also works for this dump function.

**Format:**

C/C++: void VR\_API VR\_SetDumpFile(const char\* filename);  
.NET: void SetDumpFile(string filename);  
OCX: long SetDumpFile(BSTR filename);

**Parameters:**

filename: the full path of the filename you would like to dump the data into. Usually it is a "\*.pcap" file type, which can be used and open in Wireshark.

**Return:**

null

**Sample:**

SetDumpFile("c:\\temp\\abc.pcap");

### 3.71 SetDumpFileMaxSize

**Description:** Set max size of the dump file. In a long run system, it will dump to file name to original-01.pcap, original-02.pcap,..... once reached the max size.

**Format:**

C/C++: void VR\_API VR\_SetDumpFileMaxSize(int maxSize);  
.NET: void SetDumpFile(int maxSize);  
OCX: long SetDumpFile(int maxSize);

**Parameters:**

maxSize: in bytes. 1K = 1024, 1M = 1048576, 100M = 104857600

**Return:**

null

**Sample:**

SetDumpFileMaxSize(1048576000); //Set max file size to 1000 MB

**BIB recording functions:**

## 3.72 EnableCiscoBIBRecording

**Description:** Enable Cisco BIB Recording

**Format:**

C/C++: void VR\_API VR\_EnableCiscoBIBRecording()

.NET: void EnableCiscoBIBRecording()

OCX: void EnableCiscoBIBRecording()

**Parameters:**

**Return:**

null

**Sample:**

## 3.73 SetCiscoBIBRecordingTrunkName

**Description:** Set Cisco BIB Recording Trunk Name. In SIP INVITE To header, To: <sip:7777@192.168.1.170>, 7777 is the recording trunk name/DID, which marks SIP call is for BIB recording. You might setup multiple BIB recording trunks, separate with ';'.

**Format:**

C/C++: void VR\_API VR\_SetCiscoBIBRecordingTrunkName(const char\* trunkName);

.NET: void SetCiscoBIBRecordingTrunkName(string trunkName)

OCX: void SetCiscoBIBRecordingTrunkName(BTSTR trunkName)

**Parameters:**

**Return:**

null

**Sample:**

```
SetCiscoBIBRecordingTrunkName("7777;6666");
```

### 3.74 SetCiscoBIBIncomingTrunkName

**Description:** Set Cisco BIB Incoming Trunk Name. If multiple names, separate with ';'.  
A sample in SIP INVITE:

```
From: <sip:1004@192.168.1.146;x-nearend;x-refci=20482684;x-  
nearendclusterid=StandAloneCluster;x-nearenddevice=SEP0090F5DE91C9;x-  
nearendaddr=1004;x-farendrefci=20482683;x-farendclusterid=StandAloneCluster;x-  
farenddevice=PCBest_Telekom_PBX;x-farendaddr=3551234>;tag=1835~4b10e8a8-  
eb98-46b1-b2a2-036e5a422eac-20482688
```

In this case, PCBest\_Telekom\_PBX marked by x-farenddevice is the incoming trunk name

**Format:**

C/C++: void VR\_API VR\_SetCiscoBIBIncomingTrunkName(const char\* trunkName);

.NET: void SetCiscoBIBIncomingTrunkName(string trunkName)

OCX: void SetCiscoBIBIncomingTrunkName(BTSTR trunkName)

**Parameters:**

**Return:**

null

**Sample:**

```
SetCiscoBIBIncomingTrunkName ("PCBest_Telekom_PBX ");
```

### 3.75 SetCiscoBIBOutgoingTrunkName

**Description:** Set Cisco BIB Outgoing Trunk Name. If multiple names, separate with ';'.  
A sample in SIP INVITE:

```
From: <sip:1004@192.168.1.146;x-nearend;x-refci=20482701;x-  
nearendclusterid=StandAloneCluster;x-nearenddevice=SEP0090F5DE91C9;x-  
nearendaddr=1004;x-farendrefci=20482702;x-farendclusterid=StandAloneCluster;x-  
farenddevice=TEST_TRUNK;x-farendaddr=176>;tag=1847~4b10e8a8-eb98-46b1-  
b2a2-036e5a422eac-20482706
```

In this case, TEST\_TRUNK marked by x-farenddevice is the outgoing trunk name

**Format:**

C/C++: void VR\_API VR\_SetCiscoBIBOutgoingTrunkName(const char\* trunkName);  
.NET: void SetCiscoBIBOutgoingTrunkName string trunkName)  
OCX: void SetCiscoBIBOutgoingTrunkName(BTSTR trunkName)

**Parameters:****Return:**

null

**Sample:**

SetCiscoBIBOutgoingTrunkName ("TEST\_TRUNK");

### 3.76 EnableSaveCallPcapFile

**Description:** Enable generating a Wireshark compatible pcap file for each call. The file name will be the same as wav file, but the extension of file is .pcap in the same folder of wav.

**Format:**

C/C++: void VR\_API VR\_EnableSaveCallPcapFile();  
.NET: void EnableSaveCallPcapFile()  
OCX: void EnableSaveCallPcapFile()

**Parameters:****Return:**

null

**Sample:**

### 3.77 GetPcapDriverType

**Description:** Get pcap driver type installed in the system. We know for Windows, the original driver for sniffing the network packet is WinPcap, but since Win10, Npcap is taken over to be the standard network capture driver. You can use this function to determine what kind of driver is installed. If there is none, you will need to ask clients to install the driver first.

**Format:**

C/C++: int VR\_API VR\_GetPcapDriverType();

.NET: int GetPcapDriverType ()

OCX: LONG GetPcapDriverType ()

**Parameters:**

**Return:**

0 = No pcap driver is installed in the system

1 = WinPcap driver is installed in the system

2 = Npcap driver is installed in the system

**Sample:**

## 3.78 GetPcapDriverTypeStr

**Description:** Get pcap driver in string description.

**Format:**

C/C++: const char\* VR\_API VR\_GetPcapDriverTypeStr();

.NET: string GetPcapDriverTypeStr ()

OCX: BSTR GetPcapDriverTypeStr ()

**Parameters:**

**Return:**

"" = No pcap driver is installed

"WinPcap..." = WinPcap driver is installed

"Npcap..." = Npcap driver is installed

## 3.79 EnableAddingDateAndTimeInAudio      new

**Description:** Enable adding an announcement in recording audio, to indicate the date and time for the recorded call.

**Format:**

C/C++: void VR\_API VR\_EnableAddingDateAndTimeInAudio(int enabled);  
.NET: void EnableAddingDateAndTimeInAudio(int enabled)  
OCX: void EnableAddingDateAndTimeInAudio(int enabled)

**Parameters:**

enabled: 0 = disabled, 1 = add at the beginning, 2 = add at the end

**Return:**

### 3.80 SetAddingDateAndTimeInAudioText new

**Description:** How to announce the date and time in recording. If it is "", default it is "Call recorded %s. Audio of recorded call: ". Recorder will use current system date and time to replace %s.

**Format:**

C/C++: void VR\_API VR\_SetAddingDateAndTimeInAudioText (const char\* fmt);  
.NET: void SetAddingDateAndTimeInAudioText(string fmt)  
OCX: void SetAddingDateAndTimeInAudioText(BTSTR fmt)

**Parameters:**

fmt: the string to announce. For example: "This recording was made at %s, over California 2 system at LA data center"

**Return:**

## 4 Setup Packet Filter

Sometimes you might need to set a filter in the driver to only look at the specific packets because forwarding the whole network's traffic into one port leads to a lot for VoIP Recorder to process.

Create a file named 'pcap\_filter.txt' and put it under same folder of VoIP recorder's exe and dll files.

You can refer to two samples over there already: pcap\_filter\_rtp\_only.txt and pcap\_filter\_sip\_rtp.txt.

Also refer to the following links about how to set filters:

## PC Best Networks VoIP Recorder V2 SDK Reference

<http://www.tcpdump.org/manpages/pcap-filter.7.html>

<http://wiki.wireshark.org/CaptureFilters>

[http://docs.nimsoft.com/prodhelp/en\\_US/Probes/Catalog/net\\_traffic/1.3/index.htm?toc.htm?1925170.html](http://docs.nimsoft.com/prodhelp/en_US/Probes/Catalog/net_traffic/1.3/index.htm?toc.htm?1925170.html)