VTGO for PocketPC API
Release 2.0

Programmer's Reference Guide

Corporate Headquarters

IP blue Software Solutions
111 Town Square Place, Suite 540
Jersey City, NJ 07310, USA
http://www.ipblue.com
Phone: (212) 485-1200
Fax: (212) 485-1380

Introduction

Welcome to IP blue VTGO for PocketPC Application Programming Interface (API).

IP blue VTGO softphone is a Pocket PC softphone for the Cisco IP Telephony environment. VTGO APIs allow third party developers to "voice-enable" their applications. Using VTGO APIs, client application can request VTGO softphone make telephone calls, accept calls, transfer calls, send DTMF digits, etc.

About This Guide

This guide is indented for developers who write PocketPC applications and want to integrate with VTGO softphone. It contains detailed descriptions of VTGO API interfaces and provides guidelines for using these interfaces. This guide also contains examples for eVC and HTML page, although VTGO API can be used with any application tools that supports standard DLLs or COM (eVB, VB.NET, C#, scripting languages, others).

API Reference

APIs have been designed to be easy to use, allowing the developer to concentrate on the function of the application rather than being bogged down in technical details concerning the APIs.

Since VTGO softphone is providing all telephony functionality, softphone has to run on the Pocket PC device for third party application to be able to use it. A subset of API functions makes it easy to control VTGO (RunProgram, ShowProgram, ExitProgram), while another subset allows to request VTGO application to perform telephony commands (MakeCall, AcceptCall, others).

APIs are implemented as a standard DLL (VTGO_Proxy.dll) and COM DLL (VTGO_Proxy_Com.dll). Function callback mechanism is implemented for notifications (telephony events) in standard DLL. This means that languages that do not support callbacks (e.g. scripting languages, eVB) will not be able to receive events.

COM version of VTGO APIs is implemented in VTGO_Proxy_Com.dll. COM dll exposes Phone object offers the same functions as standard DLL. Please see sample code at the end of this document.

Functions

MakeCall

Declaration	**long MakeCall(BSTR** *sPhone***)**
Parameters	*sPhone* - Specifies the destination phone number
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to make call to specified phone number.

AcceptCall

Declaration	**long AcceptCall()**
Parameters	None
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to accept incoming phone call. Has no effect if there is no incoming phone call.

EndCall

Declaration	**long EndCall()**
Parameters	None
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to end active call.

TransferCall

Declaration	**long TransferCall(BSTR** *sDestination***)**
Parameters	*sDestination* – Specifies the destination phone number to transfer call to
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to perform one-step transfer of the active call to the specified destination number. Has no effect if there is no phone call in progress.

SendDTMFDigits

Declaration	**long SendDTMFDigits(BSTR** *sDigits***)**
Parameters	*sDigits* **–** DTMF digits to send
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to send specified digits (DTMF tones) to the switch. sDigits parameter can be 1 or more characters from this list: 0-9, a-z, A-Z, #, and *. Characters will be converted to dialable digits before sending (e.g. "a" will be sent as 2, "d" as 3, etc).

HoldCall

Declaration	**long HoldCall()**
Parameters	None
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to put active call on hold. Has no effect if there is no active call in progress.

ResumeCall

Declaration	**long ResumeCall()**
Parameters	None
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to resume call that was placed on hold.

IgnoreCall

Declaration	**long IgnoreCall()**
Parameters	None
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Causes softphone to "ignore" incoming call, meaninig ringing tone, vibration and/or notification ballons will be turned off. However, softphone user will still be able to answer this call.

RunProgram

Declaration	**long RunProgram()**
Parameters	None
Returned Value	0 if success, 1 if failure
Remarks	Starts softphone application. Fails if sofphone application executable (VTGO.exe) can not be found on the device. If softphone was found in running state, returns with success.

EndProgram

Declaration	**long EndProgram()**
Parameters	None
Returned Value	0 if success, 1 if failure
Remarks	Shuts down softphone application. If there are any phone calls in progress, they will be disconnected. Has no effect if softphone is not running.

ShowProgram

Declaration	**long ShowProgram(bool** *bShow***)**
Parameters	*bShow* – show or hide softphone
Returned Value	0 if success, 1 if failure (if softphone was not running)
Remarks	Shows or hides running softphone application.

IsProgramRunning

Declaration	**bool IsProgramRunning()**
Parameters	None
Returned Value	True if softphone is running, False if not.
Remarks	Detects if softphone is currently running on the device.

AllowSoftphonePopup

Declaration **long AllowSoftphonePopup(bool** *bAllow***)**

Parameters *bShow* – allow or disallow softphone popup

Returned Value 0 if success, 1 if failure

Remarks By default, softphone will popup (come to foreground) on incoming call or when call is connected. 3rd party application might want to keep the softphone GUI hidden all the time. By calling AllowSoftphonePopup function and passing false parameter, API client will ensure that softphone will come to foreground on its own.

GetState

Declaration **long GetState()**

Parameters None

Returned Value 0 - softphone is not running
1 - softphone is running, but not registered with CallManager
2 - softphone is running and is in idle state (no phone calls in progress)
3 – outlbound call has been originated and is ringing on called party device
4 – incoming call is ringing on the device
5 – call is in progress

Remarks Queries softphone for telephony state.

RegisterCallbackFunction

Declaration **long RegisterCallbackFunction (*pCallbackFunc)**

Parameters Pointer to third party application's callback function that will be called by API DLL when it needs to notify application about softphone's event

Returned Value 0 if success, 1 if failure

Remarks Callback function must be declared in third party app as a static function with the following parameters:

```
void CallbackFunction (CallbackData_t CBData,
                       void *pContext)
```

where CallbackData_t type is declared as

```
typedef struct
{
    DWORD  m_dwProcessID;
    DWORD  m_dwEventID;
    char   m_szDN[128];
    char   m_szCallID[128];
    char   m_szParam1[128];
    char   m_szParam2[128];
    char   m_szParam3[128];
    DWORD  m_dwParam1;
    DWORD  m_dwParam2;
    DWORD  m_dwParam3;
} CallbackData_t;
```

and pContext is optional context pointer.

See Proxy_Tester_eVC sample application for more details.

Events

The following events are reported to third party application via m_dwEventID parameter in callback function:

Event	Event ID	Description
EVENT_APP_STARTED	1	Softphone started
EVENT_APP_CLOSED	2	Softphone closed
EVENT_REGISTERED	3	Softphone registered with server
EVENT_UNREGISTERED	4	Softphone unregistered
EVENT_ONHOOK	5	Softphone went onhook
EVENT_OFFHOOK	6	Softphone went offhook
EVENT_RINGING	7	Incoming call is ringing
EVENT_ALERTING	8	Outbound call has been originated
EVENT_CONNECTED	9	Call has been established
EVENT_DISCONNECTED	10	Call has disconnected
EVENT_HELD	11	Active call has been placed on hold
EVENT_RESUMED	12	Call has been resumed
EVENT_ONE_STEP_TRANSFER	13	One-step transfer has been initiated
EVENT_TRANSFER_INITIATED	14	Two-step (consultative) transfer has been initiated
EVENT_TRANSFER_ COMPLETE	15	Two-step transfer has been complete
EVENT_TRANSFER_CANCELLED	16	Two-step transfer has been cancelled
EVENT_CONFERENCE_INITIATED	17	Conference call has been initiated
EVENT_CONFERENCE_COMPLETE	18	Conference call has been complete
EVENT_CONFERENCE_CANCELLED	19	Conference call has been cancelled
EVENT_CALL_PARKED	20	Active call has been parked

Some of the events are accompanied with additional data. For example, m_szCallID parameter will contain call reference ID for telephony-related events. m_szParam1, m_szParam2, m_szParam3 will contain event-specific information, e.g. caller ID, caller name, etc. See Proxy_Tester_eVC sample application for more details.

Installation and Distribution

VTGO for PocketPC software exposes API 2.0 interface starting with version 2.1.0.85. If softphone version 2.1.0.85 or later is installed on PocketPC device, all necessary support files have been also installed.

Third party developers may not re-distribute VTGO API DLLs with their applications. VTGO API DLLs and supporting files must be installed by VTGO installation program to ensure compatibility and proper functionality. Softphone setup installs VTGO_Proxy.dll and VTGO_Proxy_COM.dll files to Windows folder, where they can be found by third party applications.

Sample Code

Following is a snippet of an eVC program that makes a call to 212-555-1212. Notice that this sample is checking to see if softphone program is running before attempting to make a phone call:

```
typedef long (CALLBACK* LPFNDLLFUNC_MAKECALL)(BSTR);
typedef bool (CALLBACK* LPFNDLLFUNC_IS_PROGRAM_RUNNING)();

void CMyDlg::MakeCall(void)
{

      long nRetval;
      HINSTANCE hDLL;
      LPFNDLLFUNC_MAKECALL lpfnMakeCall;
      LPFNDLLFUNC_IS_PROGRAM_RUNNING lpfnIsProgramRunning;

      hDLL = LoadLibrary(_T("VTGO_Proxy.dll"));

      if (hDLL != NULL)
      {
            lpfnMakeCall = (LPFNDLLFUNC_MAKECALL)
                  GetProcAddress(hDLL, _T("MakeCall"));

            lpfnIsProgramRunning=(LPFNDLLFUNC_IS_PROGRAM_RUNNING)
                  GetProcAddress(hDLL, _T("IsProgramRunning"));

            bool bProgramRunning = lpfnIsProgramRunning();
            if (!bProgramRunning)
            {
                  AfxMessageBox(_T("Softphone is not running"));
                  FreeLibrary(hDLL);
                  return;
            }

            CString sPhone = _T("2122221212");
            nRetval = lpfnMakeCall(sPhone);
            if (nRetval != 0)
            {
                  AfxMessageBox(_T("Failed to MakeCall"));
            }

            FreeLibrary(hDLL);
      }
      else
      {
            AfxMessageBox(_T("Failed to load dll"));
      }
}
```

Softphone COM APIs are "safe for scripting" and thus allow softphone integration with scripting languages. For example, a web pages running in Pocket IE can control the softphone and make phone calls, which is illustrated in the following sample:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function RunProgram()
{
  var oPhone = new ActiveXObject("ProxyCom.Phone.1");
  if (!oPhone)
    return false;
  oPhone.RunProgram();
  return true;
}

function EndProgram()
{
  var oPhone = new ActiveXObject("ProxyCom.Phone.1");
  if (!oPhone)
    return false;
  oPhone.ExitProgram();
  return true;
}

function MakeCall()
{
  var oPhone = new ActiveXObject("ProxyCom.Phone.1");
  if (!oPhone)
    return false;
  if (oPhone.IsProgramRunning())
  {
    var sDestination = document.txtDestination.value;
    oPhone.MakeCall(sDestination);
    return true;
  }
  else
  {
    alert("Softphone is not running");
    return false;
  }
}

function IsPhoneRunning()
{
  var oPhone = new ActiveXObject("ProxyCom.Phone.1");
  if (!oPhone)
    return false;
  if (oPhone.IsProgramRunning())
    return true;
  else
    return false;
}
```

```
function ShowProgram(bShow)
{
  var oPhone = new ActiveXObject("ProxyCom.Phone.1");
  if (!oPhone)
    return false;

  if (oPhone.IsProgramRunning())
  {
    oPhone.ShowProgram(bShow);
    return true;
  }
  else
  {
    alert("Softphone is not running");
    return false;
  }
}
</SCRIPT>
</HEAD>


<BODY>
<H3>Softphone Integration Sample</H3>

Destination
<input name="txtDestination" size="8">
<input type="button" value="     Dial     "
onClick="javascript:MakeCall();">

<br>
<br>
<input type="button" value=" Start Program "
onClick="javascript:RunProgram();">
<input type="button" value="  End Program  "
onClick="javascript:EndProgram();">

<br>
<br>
<input type="button" value=" Show Program "
onClick="javascript:ShowProgram(true);">
<input type="button" value=" Hide Program "
onClick="javascript:ShowProgram(false);">

</BODY>
</HTML>
```