SIP SERVER SDK v2.0

TECHNICAL DOCUMENTATION

VERSION 1.1

CONTENTS

INTRODUCTION AND QUICK START

The VaxTele COM *(Component Object Model)* component *VaxTeleServerSIP.dll*. It is a collection of functions and events that allows you to develop SIP (sessions initiation protocol) based SERVER, IP-PBX, IVR and other IP-Telephony related services.

The VaxTele COM component *(VaxTeleServerSIP.dll)* functions and events enables the rapid development of SIP based servers and allows you to use it in your call centers as telephony server, in your offices as virtual PBX to interconnect two or more remote offices, to provide call routing services, to provide IP-Telephony PC-to-phone services and many more.

EXPORTED FUNCTIONS

Initialize()

The *Initialize()* function initializes VaxTele SIP Server COM component. It allocates SIP listen port and starts listening for incoming SIP requests and triggers the COM *(Component Object Model)* events accordingly. It also allocates RTP port(s) for voice streaming.

Syntax

```
boolean Initialize(
                DomainName,
                SIPListenIP,
                SIPListenPort,
                RTPListenIP,
                RTPListenPort
                )
```

Parameters

DomainName (string)

The value of DomainName parameter is internally used to create SIP URI(s). It is also used as “realm” field in SIP packets during the authentication of SIP clients *(softphone, hardphone, ATA etc)* and call processing.

DomainName parameter can be blank/empty. If its value is blank/empty string then SIPListenIP parameter value is use in SIP authentication process and to create SIP URI(s).

e-g: if value of DomainName is sip.vaxtele.com then VaxTele creates SIP URI *sip:username@sip.vaxtele.com*
But if value of DomainName is “” empty string and SIPListenIP value is 10.3.5.66 then VaxTele creates SIP URI *sip:username@10.3.5.66*

Note: It is not necessary that DomainName should point to an IP address.

SIPListenIP (string)

Specifies the IP on which VaxTele listen for incoming SIP requests. It can be the IP address assigned to the computer on which VaxTele COM integrated SIP server is running.

SIPListenPort (integer)

Specifies the port number on which you want your SIP server to receive SIP requests. Standard SIP listen port is 5060

RTPListenIP (string)

Specifies the IP address on which your VaxTele integrated SIP server receives voice streams. If multiple IP addresses are assigned to the computer on which you are running SIP server then SIPListenIP and RTPListenIP can be different.

RTPListenPort (integer)

Specifies the start port number for the range of RTP ports allocation. Pass -1 and then VaxTele randomly choose a RTP port and allocates to the call for voice streaming.

Otherwise allocation starts from the parameter value and increments for even number. For RTP compliance, RTP port number must be an even number.

If RTPListenPort = –1 then

Call-1	3828
Call-2	4042
Call-3	8922

Value is –1 then random RTP port number assigns to each call.

If RTPListenPort = 2234

Call-1	2236
Call-2	2238
Call-3	2240

Note: According to the RFC 2327, the value of listen port must be in the range of 1024 to 65535 inclusive, for RTP compliance it should be an even number.

Return Value

If the function fails, the return value is zero and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
Initialize(“”, “192.168.0.3”, 5060, “192.168.0.3”, -1)
Initialize(“sip.vaxtele.com”, “10.18.0.3”, 5060, “10.18.0.3”, -1)
Initialize(“vaxtele.com”, “209.237.151.17”, 5060, “209.237.151.18”, -1)
```

See Also

UnInitialize(), GetVaxObjectError()

UnInitialize()

The *UnInitialize()* function simply un-allocates all the resources previously allocated by *Initialize()* function.

Syntax

```
UnInitialize()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
UnInitialize()
```

See Also

Initialize()

SetLicenseKey()

After you make the order, we deliver a license key to our customers to unlock the trial version of VaxTele COM component.

Use that license key by calling *SetLicenseKey()* exported function and trial version starts working as registered version without expiry and trial period limitations.

Syntax

```
SetLicenseKey(LicenseKey)
```

Parameters

LicenseKey (string)
Specifies the License key.

Return Value

No return value.

Example

```
SetLicenseKey("LicenseKey")
Initialize("", "192.168.0.3", 5060, "192.168.0.3", -1)
```

See Also

Initialize(), GetVaxObjectError()

GetVaxObjectError()

Call this function, to get the error for the last operation that failed.

Syntax

```
integer GetVaxObjectError()
```

Parameters

No parameters.

Return Value

Error code number.

200	Fail to initialize RTP Socket.
201	Fail to allocate RTP port or RTP listen IP is not correct.
202	Fail to create RTP task manager.
203	Fail to start media thread.
204	Fail to create RTP communication manager.
205	Unable to use RTP communication manager.
206	Unable to open file.
207	Unable to read file.
208	Wave file format is not correct, please use 8000Hz, 16bit, mono uncompressed wave file.
209	Invalid play wave ID.
210	Fail to start recording manager.
211	Call is not in the voice session.
212	Fail to initialize VaxTele object.
213	Error to create SDP body.
214	Error to create INVITE request.
215	Error to initialize SIP communication layer.
216	Error to open SIP listen port or SIP listen IP is not correct.
217	Error to create SIP task manager.
218	Error to create REGISTER request.
219	Error to create UN-REGISTER request.
220	Error to create BYE request.
221	Error to create CANCEL request.
222	Call-Id is not valid or call does not exist.
223	Session-Id is not valid or session does not exist.
224	Login does not exist.
225	Login length is not correct.
226	Line does not exist.
227	Cann't send SIP response.
228	SIP response code is not valid, please check SIP RFC 3261.
229	Error to create SIP message.

230	Line already exist.
231	Reg-Id is not correct or does not exist.
232	Error to create SIP reponse.
233	User is not registered.
234	Invalid codec OR there is no codec found for voice streaming.
235	Invalid DTMF type.
236	Other party does not support RFC2833 DTMF digits.
237	Error to create REFER request to transfer the call.
238	Error to create event handler.
239	Invalid license key.
240	Invalid digit for DTMF.
241	Invalid proxy URI.
242	Line is not registered.

Example

```
SetLicenseKey("LicenseKey")

Result = Initialize("", "192.168.0.3", 5060, "192.168.0.3", -1)
if(Result == 0) GetVaxObjectError()
```

See Also

Initialize(), SetLicenseKey()

AddUser()

The *AddUser()* function is use to add users to the SIP Server developed by using VaxTele COM component.

VaxTele COM *(VaxTeleServerSIP.dll)* component internally creates a list of users to process the SIP registration and call requests.

When a user is added by calling *AddUser()* function then its login and password can be used in any SIP based softphone or hardphone to connect and register to your SIP Server developed by using VaxTele COM component.

Generally, all SIP clients *(softphone, hardphone, ATA, wifi SIP phone)* require these settings to connect to any SIP server.

- User Name
- Display Name
- Authorization Login
- Authorization Password
- Domain/Realm
- SIP proxy

e-g:

```
Initialize("sip.vaxtele.com", "192.168.0.3", 5060, "192.168.0.3", -1)
AddUser("9090", "123")
```

User Name	9090
Display Name	9090
Authorization Login	9090
Authorization Password	123
Domain/Realm	sip.vaxtele.com
SIP proxy	192.168.0.3

Add these settings in any SIP softphone and then that softphone will register to your SIP server without any problem.

Syntax

```
boolean AddUser(
                UserLogin,
                LoginPwd
                )
```

Parameters

UserLogin (string)
String that specifies the user's login.

LoginPwd (string)
String that specifies the password of user's login.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SetLicenseKey("LicenseKey")
Initialize("sip.vaxtele.com", "192.168.0.3", 5060, "192.168.0.3", -1)

Result = AddUser("9090", "123")
if(Result == 0) GetVaxObjectError()
```

See Also

RemoveUser(), DialCallToUser(), GetVaxObjectError()

RemoveUser()

VaxTele COM component internally creates list of users for authentication and registration purposes. *RemoveUser()* function is used to remove any user added by the previous call to *AddUser()* function.

Syntax

```
RemoveUser(UserLogin)
```

Parameters

UserLogin (string)
String that specifies the user's login.

Return Value

No return value.

Example

```
RemoveUser("9092")
RemoveUser("Jason")
```

See Also

AddUser(), DialCallToUser()

AuthClientRegister()

SIP clients *(softphone, hardphone, ATA, wifi phone)* send SIP REGISTER request to connect and register to SIP server(s).

When VaxTele receives REGISTER request then it triggers *OnClientRegister()* event. To start the authentication process, call *AuthClientRegister()* function.

Call *AuthClientRegister()* function and then VaxTele;

1. Starts SIP authentication process.
2. Completes authentication process.
3. Then accepts registration (REGISTER) request.

Please see **SIP PHONE REGISTRATION FLOW** (Page. 105) for more details.

Syntax

```
boolean AuthClientRegister(RegId)
```

Parameters

RegId (integer)
Identifies the unique Id of SIP registration request.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnClientRegister(UserLogin, Domain, RegId)
{
        AuthClientRegister(RegId)
}
```

See Also

AcceptClientRegister(), RejectClientRegister(), OnClientRegister()

AcceptClientRegister()

During the SIP registration process, SIP clients *(softphone, hardphone)* send *(REGISTRATION)* requests. SIP servers authorize login and password and then accept the registration request.

In VaxTele, it receives registration request then VaxTele triggers *OnClientRegister()* event. Call *AcceptClientRegister()* function in that event simply accepts the registration request without authorizing the login and password.

Actually, call to *AcceptClientRegister()* function skips the login authorization process and accepts the registration request.

Please see **SIP PHONE REGISTRATION FLOW** (Page. 105) for more details.

Syntax

```
boolean AcceptClientRegister(RegId)
```

Parameters

RegId (integer)
Identifies the unique Id of SIP registration request.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnClientRegister(UserLogin, Domain, RegId)
{
	AcceptClientRegister(RegId)
}
```

See Also

AuthClientRegister(), RejectClientRegister(), OnClientRegister()

RejectClientRegister()

VaxTele receive registration requests from SIP clients *(softphone, hardphones)* and triggers *OnClientRegister()* event. *RejectClientRegister()* can be used to reject any registration request.

Syntax

```
boolean RejectClientRegister(RegId)
```

Parameters

RegId (integer)
Identifies the unique Id of SIP registration request.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnClientRegister(UserLogin, Domain, RegId)
{
        RejectClientRegister(RegId)
}
```

See Also

AuthClientRegister(),AcceptClientRegister(), OnClientRegister()

AddLine()

VaxTele is a SIP *(session initiation procotol)* based SIP server. It can also be connected to other SIP servers like *(asterisk, openSER, cisco call manager, Nortel PBX, quantum gateway)* by using SIP protocol and then it can send/receive call requests to those SIP servers and devices.

For example, if you want to make VaxTele work with asterisk SIP server then create a user login in asterisk and add that user account settings in VaxTele as **LINE** by calling *AddLine()* function.

DialCallToLine() function can be used to send call requests to asterisk. VaxTele triggers *OnIncomingCallFromLine()* event for incoming call requests from asterisk.

Besides that two VaxTele servers can also be connected with *AddLine()* function.

Another use of *AddLine()* function is to provide user to PSTN call feature.

There are many SIP based IP-Telephony Service Providers (ITSP) are available on then internet, some of them are;

- www.broadvoice.com
- www.voipvoip.com
- www.inphonex.com
- www.verizon.com
- www.voxbone.com

Buy IP-Telephony service from them and then they provide SIP account settings, simply use those SIP account settings by calling VaxTele's *AddLine()* function and then dial calls by using *DialCallToLine()* function and receive calls by using *OnIncomingCallFromLine()* event.

> *NOTE: ITSP provides SIP account settings, first test those settings directly by using any softphone. Dial and receives phone calls with softphone, just to make sure that settings are working properly and then use those settings in VaxTele.*

So, *AddLine()* function with some other functions can be used to connect, register, dial and receive calls from external third party SIP server(s).

1. RegisterLine()
2. OnIncomingCallFromLine()
3. UnRegisterLine()
4. DialCallToLine()

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 113) for more details.

Syntax

```
boolean AddLine(
                LineId,
                UserName,
                Login,
                LoginPwd,
                DisplayName,
                DomainRealm,
                ProxyURI,
                OutboundProxy
                )
```

Parameters

LineId (integer)
: Any unique value to identify the LINE in other line related functions and events.

UserName (string)
: Specifies the user name provided by ITSP (IP-Telephony service provider) or third party SIP server.

Login (string)
: Specifies the login provided by ITSP or third party SIP server.

LoginPwd (string)
: Specifies the password provided by ITSP or third party SIP server.

DisplayName (string)
: Provided by ITSP or third party SIP server.

DomainRealm (string)
: Specifies the DomainRealm value provided by ITSP or third party SIP server.

ProxyURI (string)
: Specifies the SIP Proxy provided by ITSP or third party SIP server.

OutboundProxy (string)
: Specifies the outbound Proxy provided by ITSP or third party SIP server.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
AddLine(1, “9034”, “9034”, “3245”, “”, “sip.abc.com”, “192.168.0.3”, “”)
AddLine(2, “8034”, “88034”, “Hello”, “John”, “abc.com”, “10.18.0.3”, “”)

DialCallToLine(1, “001914600518”, “32401”, 20)
```

See Also

RemoveLine(), RegisterLine(), DialCallToLine(), OnIncomingCallFromLine(), UnRegisterLine()

RemoveLine()

Function can be used to remove any LINE previously added by the call to *AddLine()* function, generally *RemoveLine()* function removes the line identified by LineId parameter.

Syntax

```
boolean RemoveLine(LineId)
```

Parameters

LineId (integer)
Unique value to identify the LINE.

Return Value

No return value.

Example

```
RemoveLine(1)
RemoveLine(2)
```

See Also

AddLine()

RegisterLine()

The LINE added by previous call to *AddLine()* function can be registered to the external third party SIP server or service provider (ITSP) with *RegisterLine()* function.

Syntax

```
boolean RegisterLine(LineId)
```

Parameters

LineId (integer)
Unique value to identify the LINE.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
RegisterLine(1)
RegisterLine(2)
```

See Also

UnRegisterLine(), AddLine(), OnLineRegisterSuccess()

UnRegisterLine()

If a LINE, which is already registered to the external third party SIP server by the previous call to *RegisterLine()*function then *UnRegisterLine()* can be used to unregister that LINE.

Syntax

```
boolean UnRegisterLine(LineId)
```

Parameters

LineId (integer)
Unique value to identify the LINE.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
UnRegisterLine(1)
UnRegisterLine(2)
```

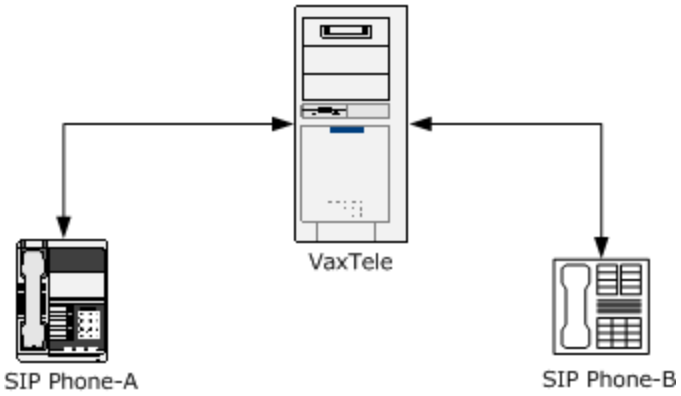
See Also

RegisterLine(), AddLine(), OnLineUnRegisterSuccess()

DialCallToUser()

The *DialCallToUser()* function is use to create and send call requests to a SIP client, who is already registered to VaxTele integrated SIP Server.

Here is the scenario to start a call session between two clients.



1. SIP phone-A sends call request to VaxTele.
2. VaxTele triggers *OnIncomingCallFromUser()* event.
3. Call *DialCallToUser()* function.
4. SIP phone-B starting ringing for incoming call.
5. SIP phone-B sends SIP provisional responses (Trying, ringing etc) to VaxTele.
6. VaxTele triggers *OnProvisionalResponse()*event.
7. Call *SendResponse()* to forward those SIP responses to SIP Phone-A.
8. SIP phone-B accepts the call and sends call-accepted response to VaxTele.
9. VaxTele triggers *OnDialCallConnected()*.
10. Call AcceptCall() function.
11. SIP phone-A call Connected.
12. Both SIP clients (SIP-phones) are connected and having voice conversation.

So, *DialCallToUser()* is used to initiate and send call requests to registered SIP clients.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details

Syntax

```
integer DialCallToUser(
                        CallFrom,
                        ToUserLogin,
                        CodecList,
                        TimeOut
                        )
```

Parameters

CallFrom (string)

Specifies the from user's login-Id. Callee (ToUserLogin) receives the call request and shows CallFrom value as caller-Id.

ToUserLogin (string)

Specifies the user login-Id to which *DialCallToUser()* sends call request. ToUserLogin should be a registered user.

CodecList (string)

String value that specifies the list of codecs to be added in the call request.

Supported codecs:

0 = GSM
1 = iLBC
2 = G711 A-Law
3 = G711 U-Law
4 = G729

"03" specifies that only GSM & G711u are supported to the call voice stream and GSM priority is higher.

"4213" specifies that supported codec for the call request initiated by *DialCallToUser()* function are G729, G711a, iLBC and G711u. Highest priority codec is 4 (G729) and lowest priority codec is 3 (G711u).

TimeOut (integer)

Specifies the time in seconds. If the other party to which call request is sent does not send any SIP response within TimeOut interval of time then time out occurs and VaxTele triggers *OnDialCallTimeOut()* event.

For example, DialCallToUser() sends call request to a softphone and waits for SIP response. Suppose softphone is under network problem and does not send any SIP response then time out on VaxTele end occurs and VaxTele triggers *OnDialCallTimeOut()* event.

Note: SIP works on UDP protocol. UDP provides an unreliable service and packets may arrive out of order or go missing without notice. That's why SIP server waits for SIP response from other party.

Return Value

If the function succeeds, the return value is unique Call-Id, otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
CallId = DialCallToUser(“Ext2098”, “Ext2043”, “32401”, 20)
if(CallId == -1) GetVaxObjectError()
```

See Also

DialCallToLine(), DialCallToURI(), OnDialCallConnected(), OnDialCallTimeOut()

DialCallToLine()

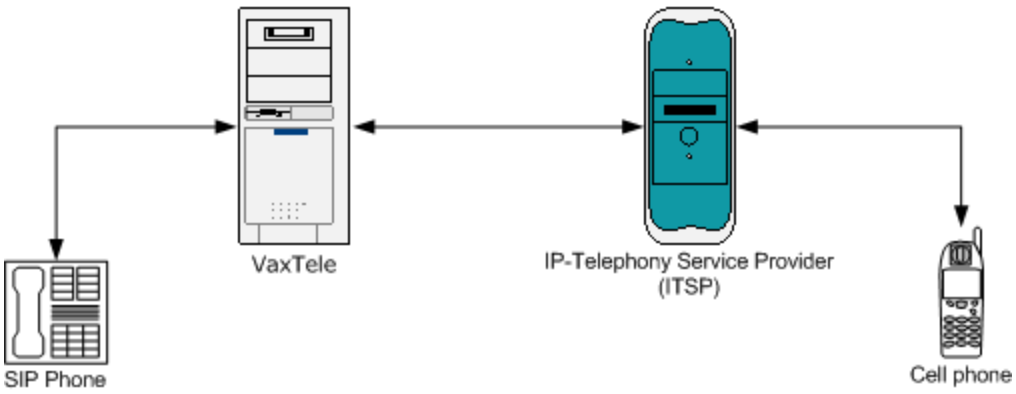
The *DialCallToLine()* function create and send call requests to LINE(s) added by previous call to *AddLine()* function.

Generally *DialCallToUser()* fuction along with other functions & events like;

- AddLine()
- RegisterLine()
- OnIncomingCallFromLine()
- OnIncomingCallFromUser()
- RemoveLine()
- UnRegisterLine()

Can be used to send and receive call requests to other external SIP server(s).

Suppose if you want that user who is using SIP phone (softphone or hardphone) could call to mobile and other telephone numbers then here is the scenario;



1. Buy a SIP account from any IP-Telephony service provider, there are many ITSP available on internet some of them are;

    - www.voxbone.com
    - www.voipvoip.com
    - www.verizon.com
    - www.broadvoice.com
    - www.inphonex.com

2. Add ITSP account settings as LINE by using *AddLine()* function
3. Add that account by using *AddUser()* function.
4. User using SIP phone sends call request.
5. VaxTele triggers *OnIncomingCallFromUser()* event.
6. In that event call *DialCallToLine()* function and send call request to ITSP.

7. PSTN or mobile phone starts ringing.
8. Callee pick up the call.
9. VaxTele triggers *OnDialCallConnected()* event.
10. In that event call *AcceptCall()* function.
11. Call connected and both user using SIP phone and person using mobile or PSTN phone can have voice conversation.

Note: Connect any softphone directly to ITSP by using the SIP account settings provided by ITSP.
Dial and receive phone calls just to make sure that account settings are working properly. Then use those settings with VaxTele.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 113) for more details.

Syntax

```
integer DialCallToLine(
                        LineId,
                        DialNo,
                        CodecList,
                        TimeOut
                       )
```

Parameters

LineId (integer)
Unique value to identify the LINE.

DialNo (string)
Specifies the dial number.

CodecList (string)
String value that specifies the list of codecs to be added in the call request.

Supported codecs:

0 = GSM
1 = iLBC
2 = G711 A-Law
3 = G711 U-Law
4 = G729

"132" specifies that iLBC, G711u and G711a are supported for the voice session and in the call request iLBC priority is highest and G711a is at lowest priority.

"02134" specifies that supported codec for the call request initiated by *DialCallToUser()* function are GSM, iLBC, G711u and G729.

In the call request, Highest priority codec is 0 (GSM) and lowest priority codec is 4 (G729).

TimeOut (integer)

Specifies the time in seconds. If the other party to which call request is sent does not send any SIP response within TimeOut interval of time then time out occurs and VaxTele triggers *OnDialCallTimeOut()* event.

For example, *DialCallToLine()* sends call request and waits for SIP response. Suppose other party is not available due to power or network failure and does not send any SIP response then time out on VaxTele end occurs and VaxTele triggers *OnDialCallTimeOut()* event.

Note: SIP works on UDP protocol. UDP provides an unreliable service and packets may arrive out of order or go missing without notice. That's why SIP server waits for SIP response from other party.

Return Value

If the function succeeds, the return value is unique Call-Id, otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
CallId = DialCallToLine(1, "001914600518", "32401", 20)
if(CallId == -1) GetVaxObjectError()
```

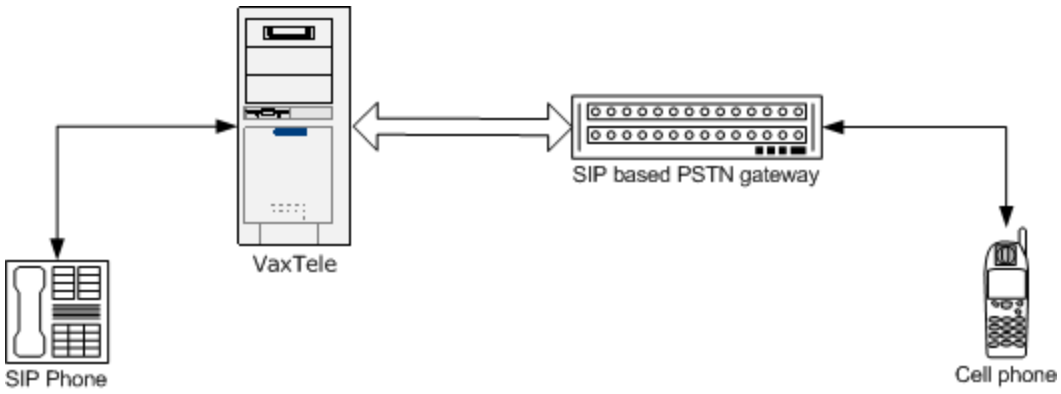
See Also

DialCallToUser(), DialCallToURI(), OnDialCallConnected()

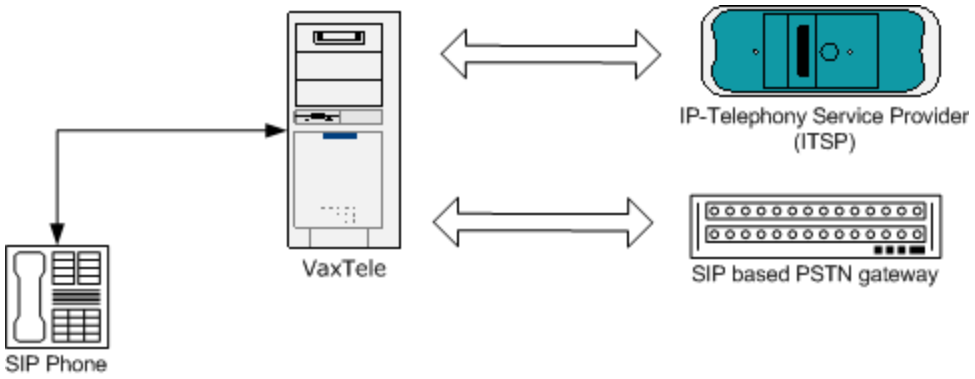
DialCallToURI()

In some cases you may require to send call requests directly to the IP of other SIP devices and SIP servers then in such cases *DialCallToURI()* can be used.

1. During VaxTele to PSTN gateway communication if your PSTN gateway is configured in a way that it receives SIP call request directly on IP then *DialCallToURI()* can be used to send SIP requests directly to PSTN gateway.



2. In call routing and termination services, service providers buy routes and forward your call traffic to those routes directly to route's SIP servers and SIP devices.
DialCallToURI() can also be used in such case.



So, *DialCallToURI()* with other functions and events;

- OnIncomingCallAnonymous()
- OnDialCallConnected()

Can be used for direct IP to IP SIP communication.

Please see **CONNECT VAXTELE WITH PSTN NETWORK** (Page. 108) for more details.

Syntax

```
integer DialCallToURI(
                        FromURI,
                        ToURI,
                        ProxyURI,
                        Login,
                        LoginPwd,
                        CodecList,
                        TimeOut
                        )
```

Parameters

FromURI (string)
: Specifies From URI in SIP call request.
sip:username@VaxTele-Listen-IP
Username in From URI appears as caller-Id on receving end.

ToURI (string)
: Specifies To URI in SIP call request.
sip:username@Destination-IP
Username in To URI appears as dial number on receving end.

ProxyURI (string)
: Specifies the IP of the SIP server or SIP devices to which you want to send the call request.

Login (string)
: Specifies the login if any otherwise leave it blank.

LoginPwd (string)
: Specifies the password if any otherwise leave it blank.

CodecList (string)
: String value that specifies the list of codecs to be added in the call request.

  Supported codecs:

  0 = GSM
  1 = iLBC
  2 = G711 A-Law
  3 = G711 U-Law
  4 = G729

  “132” specifies that iLBC, G711u and G711a are supported for the voice session and in the call request iLBC priority is highest and G711a is at lowest priority.

"02134" specifies that supported codec for the call request initiated by *DialCallToUser()* function are GSM, iLBC, G711u and G729.
In the call request, highest priority codec is 0 (GSM) and lowest priority codec is 4 (G729).

TimeOut (integer)

Specifies the time in seconds. If the other party to which call request is sent does not send any SIP response within TimeOut interval of time then time out occurs and VaxTele triggers *OnDialCallTimeOut()* event.

For example, *DialCallToURI()* sends call request and waits for SIP response. Suppose other party is not available due to power or network failure and does not send any SIP response then time out on VaxTele side occurs and VaxTele triggers *OnDialCallTimeOut()* event.

Note: SIP works on UDP protocol. UDP provides an unreliable service and packets may arrive out of order or go missing without notice. That's why SIP server waits for SIP response from other party.

Return Value

If the function succeeds, the return value is unique Call-Id, otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
DialCallToURI("sip:40@10.4.2.88", "sip:001800555777", "10.43.22.7",
"admin",   "adminpwd", "023", 20)

CallId = DialCallToURI("sip:335362@10.4.2.88",
          "sip:001800555777", "10.43.22.7", "", "", "1023", 20)

if(CallId == -1) GetVaxObjectError()
```

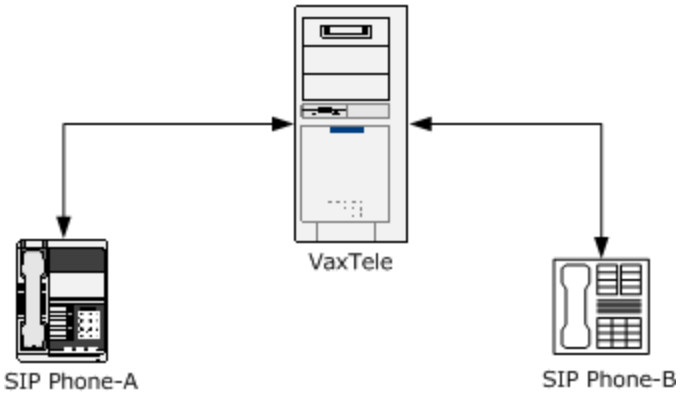
See Also

DialCallToUser(), DialCallToLine(), OnDialCallConnected(), GetVaxObjectError()

SendResponse()

During the SIP communication *SendResponse()* function is used to send/forward the SIP response (trying, ringing, sessions progress etc).

SIP-PHONE TO SIP-PHONE CALL



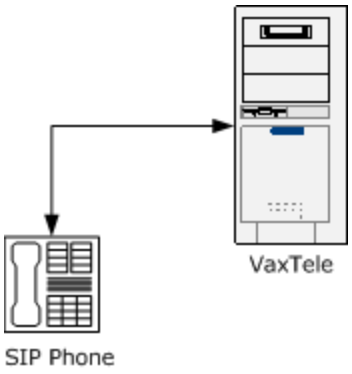
1. SIP phone-A dials a phone call and sends call request to VaxTele
2. *OnIncomingCallFromUser()* event triggers, your VaxTele integrated application receives that event and dials call to SIP phone-B by calling *DialCallToUser()* function.
3. SIP phone-B receives the call request and it starts ringing for incoming call and the SIP phone-B sends SIP response “180, Ringing” to VaxTele.
4. VaxTele triggers *OnProvisionalResponse()* event and your VaxTele integrated SIP Server application captures that event and forward “180, Ringing” response to SIP phone-A by call *SendResponse()* function.

```
OnProvisionalResponse()
{
        SendResponse()
}
```

5. SIP phone-B receives response and alerts user that SIP phone-A is ringing.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106)
for more details.

SIP-PHONE CALL REJECTION



1. SIP phone-A dials a phone call and sends call request to VaxTele *OnIncomingCallFromUser()* event triggers, your VaxTele integrated application receives that event.
2. Call *SendResponse(404, "Not Allowed")* and VaxTele sends response "404, Not Allowed" to SIP phone.

```
OnIncomingCallFromUser()
{
        SendResponse(404, "Not Allowed", "3201")
}
```

LIST OF ALL SIP RESPONSES (SIP RFC 3261)

Provisional responses 1xx			
100	Trying	180	Ringing
181	Call Is Being Forwarded	182	Queued
183	Session Progress		
Redirection 3xx			
300	Multiple Choices	301	Moved Permanently
302	Moved Temporarily	305	Use Proxy
380	Alternative Service		
Request Failure 4xx			
400	Bad Request	401	Unauthorized
402	Payment Required	403	Forbidden
404	Not Found	405	Method Not Allowed
406	Not Acceptable	407	Proxy Authentication Required

408	Request Timeout	410	Gone
413	Request Entity Too Large	414	Request-URI Too Long
415	Unsupported Media Type	416	Unsupported URI Scheme
420	Bad Extension	421	Extension Required
423	Interval Too Brief	480	Temporarily Unavailable
481	Call/Transaction Does Not Exist	482	Loop Detected
483	Too Many Hops	484	Address Incomplete
485	Ambiguous	486	Busy Here
487	Request Terminated	488	Not Acceptable Here
491	Request Pending	493	Undecipherable
Server Failure 5xx			
500	Server Internal Error	501	Not Implemented
502	Bad Gateway	503	Service Unavailable
504	Server Time-out	505	Version Not Supported
513	Message Too Large		
Global Failures 6xx			
600	Busy Everywhere	603	Decline
604	Does Not Exist Anywhere	606	Not Acceptable

- For 1xx VaxTele triggers *OnProvisionalResponse()*
- For 3xx VaxTele triggers *OnRedirectionResponse()*
- For 4xx & 5xx & 6xx VaxTele triggers *OnFailureResponse()*

Syntax

```
boolean SendResponse(
                    CallId,
                    CodecList,
                    TimeOut
                    )
```

Parameter

CallId (integer)
: Specifies the unique call identification.

StatusCode (integer)
: Status code for specific SIP response from the given above SIP responses table.

ReasonPhrase (string)
: Text phrase for SIP responses.

codecList (string)
List of supported codecs to send dial tone in “183, Session progress” response.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SendResponse (CallId, 404, “Not Allowed”, “3201”)
SendResponse (CallId, 486, “Busy Here”, “32”)
```

See Also

OnIncomingCallFromLine(), OnIncomingCallFromUser(), OnIncomingCallAnonymous()

AcceptCall()

AcceptCall() function is use to accept an incoming call.

VaxTele receives a call request and triggers the event accordingly.

- OnIncomingCallFromLine()
- OnIncomingCallFromUser()
- OnIncomingCallAnonymous()

Then *AcceptCall()* function can be used to accept an incoming call.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
boolean AcceptCall(
                    CallId,
                    CodecList,
                    TimeOut
                    )
```

Parameters

CallId (integer)

Specifies the unique call identification.

CodecList (string)

String value that specifies the list of codecs to be added in the call request.

Supported codecs:

0 = GSM
1 = iLBC
2 = G711 A-Law
3 = G711 U-Law
4 = G729

"132" specifies that iLBC, G711u and G711a are supported for the voice session and in the call request iLBC priority is highest and G711a is at lowest priority.

"02134" specifies that supported codec for to accept the call request. Codecs are GSM, iLBC, G711u and G729, highest priority codec is 0 (GSM) and lowest priority codec is 4 (G729).

TimeOut (integer)

Specifies the time in seconds. If the other party does not send accept call response within TimeOut interval of time then time out occurs and VaxTele triggers *OnAcceptCallTimeOut()* event.

For example, *AccepCall()* sends call accepted response and waits for acknowledgement. Suppose other party is not available due to power or network failure and does not send acknowledgement then time out on VaxTele side occurs and VaxTele triggers *OnAcceptCallTimeOut()* event.

Note: SIP works on UDP protocol. UDP provides an unreliable service and packets may arrive out of order or go missing without notice. That's why SIP server waits for SIP response from other party.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnIncomingCallFromUser()
{
        AcceptCall(CallId, "32401", 20)
}
```

See Also

DialCallToUser(), DialCallToLine(), DialCallToURI(), OnAcceptCallTimeOut()

CloseCall()

VaxTele internally create calls list and allocates resources to each call (memory, socket etc.) If any call is no longer needed then it must be closed by calling *CloseCall()* function. Call to *CloseCall()* function free the resources allocated to the specific call.

It can also be used to disconnect a call. *CloseCall()* function works according to the status of the call.

- Call *CloseCall()* function for incoming call then it rejects the incoming call.
- If a call is already connected then *CloseCall()* function disconnect the call.

FOR EXAMPLE

Two SIP phones (A & B) are connected through VaxTele and having voice session.

1. SIP phone-A disconnects the call.
2. VaxTele receives call disconnect request from SIP phone-A and trigger event *OnDisconnectCall()*
3. In that event call *CloseCall(A)* & *CloseCall(B)*
4. SIP phone-B receives call hungup/disconnected request.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
CloseCall(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnDisconnectCall()
{
        CloseCall(CallIdA)
        CloseCall(CallIdB)
}
```

See Also

OnDisconnectCall(), OnIncomingCallCancel(), OnAcceptCallTimeOut(), OnDialCallTimeOut()

CreateSession()

The *CreateSession()* function can be used to open a voice channel and add call(s) to that voice channel to start voice streaming.

SIP Phone-A	VaxTele Integrated SIP server	SIP phone-B
Send Call request	OnIncomingCallFromUser(A)	
	DialCallToUser(B)	Incoming call, Phone starts ringing.
	OnProvisionalResponse(B)	Send SIP response (trying, ringing etc).
Receives SIP responses	SendResponse(A)	
	OnDialCallConnected(B)	Accept incoming call
	AcceptCall(A)	Call Connected
Call Connected	OnAcceptIncomingCallConnected(A)	
	Session = CreateSession()	
	AddCallToSession(A, Session)	
	AddCallToSession(B, Session)	
Both SIP phones are connected and having voice conversation		

Multiple calls can also be added to a single voice session created by calling *CreateSession()*

SessionId = CreateSession()

AddCallToSession(CallIdA, SessionId)
AddCallToSession(CallIdB, SessionId)
AddCallToSession(CallIdC, SessionId)
AddCallToSession(CallIdD, SessionId)

Now all the 4 persons A, B, C, D can listen and speak to each other. Such feature can be used to add stealth listening in CRM and call centre solutions.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
integer CreateSession()
```

Parameters

No parameters

Return Value

If the function succeeds, the return value is Session-Id otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
if (SessionId == -1) GetVaxObjectError()
```

See Also

AddCallToSession(), RemoveCallToSession(), CloseSession(), GetCallSession()

AddCallToSession()

Call *AddCallToSession()* function to add call to the voice session previously created by the call to *CreateSession()* function.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
boolean AddCallToSession(
                        CallId,
                        SessionId
                        )
```

Parameters

CallId (integer)
Specifies the unique call identification.

SessionId (integer)
Specifies the unique voice session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
if (SessionId == -1) GetVaxObjectError()

AddCallToSession(CallIdA, SessionId)
AddCallToSession(CallIdB, SessionId)
```

See Also

CreateSession(), RemoveCallToSession(), CloseSession(),GetCallSession()

RemoveCallToSession()

Call *RemoveCallToSession()* function to remove call from the voice session previously added by the call to *AddCallToSession()* function.

Syntax

```
boolean RemoveCallToSession(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
if (SessionId == -1) GetVaxObjectError()

AddCallToSession(CallIdA, SessionId)
AddCallToSession(CallIdB, SessionId)

RemoveCallToSession(CallIdA)
RemoveCallToSession(CallIdB)
```

See Also

CreateSession(), AddCallToSession(), CloseSession(), GetCallSession()

CloseSession()

Call *CloseSession()* function to close and destroy the voice session created by the previous call to *CreateSession()* function.

Syntax

```
CloseSession(SessionId)
```

Parameters

SessionId (integer)
Specifies the unique voice session identification.

Return Value

Not return value

Example

```
SessionId = CreateSession()
if (SessionId == -1) GetVaxObjectError()

AddCallToSession(CallIdA, SessionId)
AddCallToSession(CallIdB, SessionId)

CloseSession(SessionId)
```

See Also

CreateSession(), AddCallToSession(), RemoveCallToSession(), GetCallSession()

GetCallSession()

GetCallSession() function can be used to get the current voice session id of a call.

Syntax

```
integer GetCallSession(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is then session-id otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
if (SessionId == -1) GetVaxObjectError()

AddCallToSession(CallIdA, SessionId)
CurrSessionId =GetCallSession(CallIdA)

/* CurrSessionId and SessionId values are same */
```

See Also

CreateSession(), AddCallToSession(), RemoveCallToSession(), CloseSession()

PlayWaveLoadFile()

Call to *PlayWaveLoadFile()* function loads wave file (.wav) data to the memory and returns the unique play wave identification *PlayWaveId*.

That *PlayWaveId* can be used with other play wave functions to play that loaded wave file data to voice channel create by the previous call to *CreateSession()* function.

- PlayWaveStartToSession(PlayWaveId, SessionId, LoopPlay)
- PlayWaveStopToSession(SessionId)
- PlayWaveResetToSession(SessionId)
- PlayWaveUnLoadFile(PlayWaveId)

To avoid media stream type and format conversation and save CPU cycles. *PlayWaveLoadFile()* only works with uncompress wave file (.wav) recorded at format (8000Hz, 16bit, Mono) other media file types (MP3, gsm etc) are not supported.

Syntax

```
integer PlayWaveLoadFile(FileName)
```

Parameters

FileName (string)
Specifies the wave file name.

Return Value

If the function succeeds, the return value is PlayWave-Id otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
PlayWaveId = PlayWaveLoadFile("c:\HoldMusic.wav")
if (PlayWaveId == -1) GetVaxObjectError()

SessionId = CreateSession()
AddCallToSession(CallIdA, SessionId)

PlayWaveStartToSession(PlayWaveId, SessionId, True)
```

See Also

PlayWaveUnLoadFile(), PlayWaveStartToSession(), PlayWaveStopToSession(), PlayWaveResetToSession(), CreateSession(), AddCallToSession()

PlayWaveUnLoadFile()

Call to *PlayWaveUnLoadFile()* function unloads wave file (.wav) data previously loaded by *PlayWaveLoadFile()* function.

Syntax

```
PlayWaveUnLoadFile(PlayWaveId)
```

Parameters

PlayWaveId (integer)
Specifies the unique play wave data identification.

Return Value

No return value.

Example

```
PlayWaveId = PlayWaveLoadFile("c:\HoldMusic.wav")
if (PlayWaveId == -1) GetVaxObjectError()

PlayWaveUnLoadFile(PlayWaveId)
```

See Also

PlayWaveLoadFile(), PlayWaveStartToSession(), PlayWaveStopToSession(), PlayWaveResetToSession(), CreateSession(), AddCallToSession()

PlayWaveStartToSession()

Call to *PlayWaveStartToSession()* function starts playing the wave file data to the specific voice channel/session created by *CreateSession()* exported function.

Single loaded data by the previous call to *PlayWaveLoadFile()* function can be used to play in multiple voice sessions.

```
PlayWaveId = PlayWaveLoadFile("c:\Music.wav")

SessionIdA = CreateSession()
SessionIdB = CreateSession()

PlayWaveStartToSession(PlayWaveId, SessionIdA, True)
PlayWaveStartToSession(PlayWaveId, SessionIdB, False)
```

Buffered based compression technology is introduced in VaxTele SIP Server SDK to save CPU cycles and put minimum load on CPU, while playing the wave file.

In such technology, VaxTele compress wave data just one time, buffer it and use it every time to play it. Such technology does not put voice compression load to the CPU during the wave playing process and saves a lot of CPU cycles.

Syntax

```
boolean PlayWaveStartToSession(
                                PlayWaveId,
                                SessionId,
                                LoopPlay
                                )
```

Parameters

PlayWaveId (integer)
: Specifies the unique play wave data identification.

SessionId (integer)
: Specifies the unique session identification.

LoopPlay (boolean)
: Boolean value True, plays sound repeatedly.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
PlayWaveId = PlayWaveLoadFile("c:\HoldMusic.wav")
if (PlayWaveId == -1) GetVaxObjectError()

SessionId = CreateSession()
PlayWaveStartToSession(PlayWaveId, SessionId, False)
```

See Also

PlayWaveLoadFile(),PlayWaveUnLoadFile(), PlayWaveStopToSession(), PlayWaveResetToSession(), CreateSession(), AddCallToSession()

PlayWaveStopToSession()

To stop the wave data already playing to voice session by the previous call to *PlayWaveStartToSession()*

Syntax

```
boolean PlayWaveStopToSession(SessionId)
```

Parameters

SessionId (integer)
Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
PlayWaveId = PlayWaveLoadFile("c:\HoldMusic.wav")
if (PlayWaveId == -1) GetVaxObjectError()

SessionId = CreateSession()
PlayWaveStartToSession(PlayWaveId, SessionId, False)

PlayWaveStopToSession(SessionId)
```

See Also

PlayWaveLoadFile(),PlayWaveUnLoadFile(), PlayWaveStartToSession(), PlayWaveResetToSession(), CreateSession(), AddCallToSession()

PlayWaveResetToSession()

Call to *PlayWaveResetToSession()* function to voice session restarts the wave playing.

Syntax

```
boolean PlayWaveResetToSession(SessionId)
```

Parameters

SessionId (integer)
Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
PlayWaveId = PlayWaveLoadFile("c:\HoldMusic.wav")
if (PlayWaveId == -1) GetVaxObjectError()

SessionId = CreateSession()
PlayWaveStartToSession(PlayWaveId, SessionId, False)

PlayWaveResetToSession(SessionId)
```

See Also

PlayWaveLoadFile(),PlayWaveUnLoadFile(), PlayWaveStartToSession(), PlayWaveStopToSession(), CreateSession(), AddCallToSession()

RecordWaveStartToSession()

Call to *RecordWaveStartToSession()* function starts recording the voice stream of a channel/session to a wave file.

VaxTele use (8000Hz, 16bit, mono) format to create uncompress wave file (.wav) to save CPU cycles.

Syntax

```
boolean RecordWaveStartToSession(
                                    FileName,
                                    SessionId
                                    )
```

Parameters

FileName (string)
: Specifies the wave file name.

SessionId (integer)
: Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
AddCallToSession(CallId, SessionId)

RecordWaveStartToSession("Record.wav", SessionId)
```

See Also

RecordWaveStopToSession(), RecordWavePauseToSession(), CreateSession(), AddCallToSession()

RecordWaveStopToSession()

Call to *RecordWaveStopToSession()* function stops recording the voice stream of a channel/session to a wave file.

Syntax

```
boolean RecordWaveStopToSession(SessionId)
```

Parameters

SessionId (integer)
Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
AddCallToSession(CallId, SessionId)

RecordWaveStartToSession("Record.wav", SessionId)
RecordWaveStopToSession(SessionId)
```

See Also

RecordWaveStartToSession(), RecordWavePauseToSession(), CreateSession(), AddCallToSession()

RecordWavePauseToSession()

Call to *RecordWavePauseToSession()* function pause the recording to wave file.

Syntax

```
boolean RecordWavePauseToSession(Enable, SessionId)
```

Parameters

Enable (Boolean)
Pause or un-pause the recording process.

SessionId (integer)
Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionId = CreateSession()
AddCallToSession(CallId, SessionId)

RecordWaveStartToSession("Record.wav", SessionId)

RecordWavePauseToSession(True, SessionId)
RecordWavePauseToSession(False, SessionId)
```

See Also

RecordWaveStartToSession(), RecordWaveStopToSession(), CreateSession(), AddCallToSession()

SendDTMF()

SendDTMF() function is used to send/generate DTMF digits to the specific call.

Syntax

```
boolean SendDTMF(
                    TypeDTMF,
                    CallId,
                    Digit
                    )
```

Parameters

TypeDTMF (integer)
Two type of DTMF supported

- Inband
- RTP based (RFC 2833)

Inband Mode
Value 1 enables Inband mode, in which VaxTele sends DTMF tone in the form of voice. Inband DTMF only works with G711a-Law and G711u-Law codecs, because these codecs are lossless and does not change the voice quality.

RTP based (RFC 2833)
Value 0 enables RTP based (RFC 2833) mode, almost all SIP devices use this standard to send and receive DTMF digits.

CallId (integer)
Specifies the unique call identification.

Digit (integer)
Digit value (0 to 9) and # = 10, * = 11

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SendDTMF(1, CallId-A, 0) // Digit DTMF 0 in RTP based(RFC 2833) mode.
SendDTMF(0, CallId-B, 10)  // Digit DTMF # in Inband mode.
```

See Also

DetectDTMF(), OnDigitDetectDTMF()

DetectDTMF()

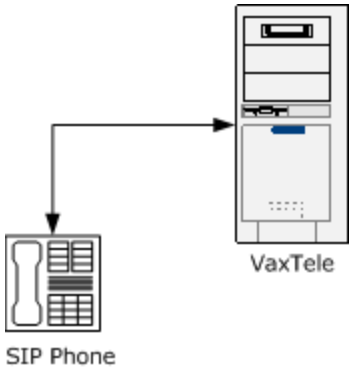
There are total three types of DTMF digit detection standards in IP-Telephony. VaxTele supports all of those three DTMF digit detection standards.

- Inband
- RTP based (RFC2833)
- SIP based (INFO)

Enable DTMF detection to the voice session, if VaxTele receives any DTMF digit then it detects those digits and triggers *OnDigitDetectDTMF()* event accordingly.

Enable either one or both DTMF detection types, it depends upon your requirement.

Note: Inband DTMF analyzes the incoming voice stream and put load on CPU. But RTP & SIP based detection does not put load on CPU.



SIP Phone-A	VaxTele Integrated SIP server
Send call request	OnIncomingCallFromUser(CallId)
	AcceptCall(CallId)
Call connected successfully	OnAcceptCallConnected(CallId)
	SessionId = CreateSession()
	DetectDTMF(0, SessionId, True)

	AddCallToSession(CallId, SessionId)
User press digit on the dial-pad	
	OnDigitDetectDTMF()

RTP BASED (RFC2833) DTMF DETECTION

It is adopted by almost all SIP based devices and softphones. It is widely used to send and receive DTMF digits.

SIP BASED (INFO) DTMF DETECTION

It is also widely used to send and receive DTMF digits.

INBAND DTMD DETECTION

In which DTMF digits are sent in the form of voice tones and VaxTele analyze the incoming voice stream and detects the tones for DTMF digits.
Inband DTMF detection only works if the other party is sending the voice tones in lossless codec G711a-Law or G711u-Law.

So, if there is voice session with G711a-Law or G711u-Law codec and other party send digits in the form of voice tone then VaxTele detects those digits and triggers *OnDigitDetectDTMF()* event.

Inband digit detection does not work with loosy codecs (GSM, G729, iLBC, speex) because these codec highly compress the voice and change its quality and Inband digit detection algorithm fails to detect the digit tones.

Syntax

```
boolean DetectDTMF(
                    TypeDTMF,
                    SessionId,
                    Enable
                    )
```

Parameters

TypeDTMF (integer)
Three types of DTMF detection are supported:

- Inband
- RTP based (RFC 2833)
- SIP based (INFO)

RTP BASED (RFC 2833)
Value 0 enables RTP based (RFC 2833) DTMF detection, almost all SIP devices use this standard to send and receive DTMF digits.

INBAND MODE
Value 1 enables Inband mode, in which VaxTele detects DTMF tone in the form of voice. Inband DTMF detection only works with G711a-Law and G711u-Law codecs, because these codecs are lossless and does not change the voice quality.

SIP BASED (INFO)
Value 2 enables SIP based (INFO) DTMF detection, most of the SIP devices also use this standard to send and receive DTMF digits.

SessionId (integer)
Specifies the unique session identification.

Enable (boolean)
Enable and disable the DTMF detection.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
DetectDTMF(1, SessionId, 1)  // Enable RTP based(RFC 2833) Digit detection
DetectDTMF(0, SessionId, 1)  // Enable Inband Digit detection
DetectDTMF(2, SessionId, 1)  // Enable SIP based (INFO) Digit detection

//Enables all type of digit detections to the voice session.
```

See Also
OnDigitDetectDTMF(), SendDTMF()

SessionLost()

Enable the session lost then during the voice session if VaxTele does not receive voice data from any client/call for 6 minutes then it triggers *OnSessionLost()* event

Syntax

```
boolean SessionLost(Enable)
```

Parameters

Enable (boolean)
Enable and disable the session lost.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
SessionLost(True)
```

See Also

OnSessionLost(), GetVaxObjectError()

TransferCallAccept()

To accept an incoming transfer call request, *TransferCallAccept()* function can be used.

Syntax

```
boolean TransferCallAccept(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnTransferCallBlind(CallId)
{
        TransferCallAccept(CallId)
}
```

See Also

TransferCallReject(), OnTransferCallBlind(), OnTransferCallAttend(), GetVaxObjectError()

TransferCallReject()

Call to *TransferCallReject()* rejects the incoming call transfer request.

Syntax

```
boolean TransferCallReject(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnTransferCallBlind(CallId)
{
	TransferCallReject(CallId)
}
```

See Also

TransferCallAccept(), OnTransferCallBlind(), OnTransferCallAttend(), GetVaxObjectError()

EnableDialTone()

This function can be used to enable ring tone on incoming call request. When you enable the ring tone then VaxTele sends “183, Session progress” SIP response and then starts ring tone streaming to incoming call. You may find more details about “183, Session progress” SIP response on internet.

Syntax

```
boolean EnableDialTone(CallId, SessionId)
```

Parameters

CallId (integer)
: Specifies the unique call identification.

SessionId (integer)
: Specifies the unique session identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
PlayWaveId = PlayWaveLoadFile(“c:\HoldMusic.wav”)

DialToneSession = CreateSession()

/* Start playing in loop */
PlayWaveStartToSession(PlayWaveId, DialToneSession, True)

/* some processing here and waiting for incoming call events */

OnIncomingCallFromUser(CallId)
{
        EnableDialTone(CallId, DialToneSession)
}
```

See Also

DisableDialTone(), GetVaxObjectError()

DisableDialTone()

This function can be used to disable ring tone on incoming call request.

Syntax

```
boolean DisableDialTone(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
OnIncomingCallFromUser(CallId)
{
        DisableDialTone(CallId)
}
```

See Also

EnableDialTone(), GetVaxObjectError()

SetVaxTeleTick()

This function can be used to start a tick and perform certain tasks. Call to *SetVaxTeleTick()* set a time internally and trigger the tick event *OnVaxTeleTick()* after that specific time. It works just like *SetTimer()* win32 API.

SetVaxTeleTick() function with event *OnVaxTeleTick()* can be used for call processing in queues, DTMF press wait time etc.

Syntax

```
boolean SetVaxTeleTick(TickId, Elapse)
```

Parameters

TickId (integer)
: Specifies the unique tick identification. TickId value should be more than 2000.

Elapse (integer)
: Specifies the time in milli seconds.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
/* Triggers OnVaxTeleTick() after every 8 seconds */

SetVaxTeleTick(2001, 8000)
```

See Also

OnVaxTeleTick(), KillVaxTeleTick(), GetVaxObjectError()

KillVaxTeleTick()

This function can be used to stop the specified tick. It works just like *KillTimer()* win32 API.

Syntax

```
boolean KillVaxTeleTick(TickId)
```

Parameters

TickId (integer)
: Specifies the unique tick identification. TickId value should be more than 2000.

Return Value

If the function succeeds, the return value is non-zero otherwise specific error code can be retrieved by calling *GetVaxObjectError()* function.

Example

```
/* Triggers OnVaxTeleTick() after every 8 seconds */

SetVaxTeleTick(2001, 8000)

KillVaxTeleTick(2001)
```

See Also

OnVaxTeleTick(), KillVaxTeleTick(), GetVaxObjectError()

GetCallTxCodecNo()

GetCallTxCodecNo() function returns the codec being used for outbound voice stream for a specific call.

Syntax

```
integer GetCallTxCodecNo(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is codec-number otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

0 = GSM, 1 = iLBC, 2 = G711 A-Law, 3 = G711 U-Law, 4 = G729

Example

```
GetCallTxCodecNo(CallId)
```

See Also

GetCallRxCodecNo(), GetVaxObjectError()

GetCallRxCodecNo()

GetCallRxCodecNo() function returns the codec being used for Inbound voice stream for a specific call.

Syntax

```
integer GetCallRxCodecNo(CallId)
```

Parameters

CallId (integer)
Specifies the unique call identification.

Return Value

If the function succeeds, the return value is codec-number otherwise it returns –1 and specific error code can be retrieved by calling *GetVaxObjectError()* function.

0 = GSM, 1 = iLBC, 2 = G711 A-Law, 3 = G711 U-Law, 4 = G729

Example

```
GetCallRxCodecNo(CallId)
```

See Also

GetCallTxCodecNo(), GetVaxObjectError()

EXPORTED EVENTS

OnClientUnRegister()

VaxTele triggers *OnClientUnRegister()* event when it receives unregister request from any SIP client.

Syntax

```
OnClientUnRegister(UserLogin)
```

Parameters

UserLogin (string)
: Specifies the User's login that sends the request.

Example

```
OnClientUnRegister(UserLogin)
{
        RemoveUser(UserLogin)
}
```

See Also

OnClientRegister(), RemoveUser()

OnClientRegister()

The *OnClientRegister()* event triggers when VaxTele receives register request from any SIP client.

Please see **SIP PHONE REGISTRATION FLOW** (Page. 105) for more details.

Syntax

```
OnClientRegister(UserLogin, Domain, RegId)
```

Parameters

UserLogin (string)
: Specifies the User's login that sends the request.

Domain (string)
: Specifies the domain, which is used to configure and register the SIP clients to VaxTele and other SIP servers.

RegId (integer)
: Specifies the unique registration request identification.

Example

```
OnClientRegister(UserLogin)
{
        AddUser()
}
```

See Also

OnClientUnRegister(), AddUser()

OnClientRegisterSuccess()

The event *OnClientRegisterSuccess()* triggers when a SIP client sends register request and that register request completes successfully.

Please see **SIP PHONE REGISTRATION FLOW** (Page. 105) for more details.

Syntax

```
OnClientRegisterSuccess(UserLogin)
```

Parameters

UserLogin (string)
Specifies the User's login that sends the request.

Example

```
OnClientRegisterSuccess(UserLogin)
{
}
```

See Also

OnClientRegister(), OnClientRegisterFail()

OnClientRegisterFail()

The event *OnClientRegisterFail()* triggers when a SIP client sends register request and that register request fails and does not complete successfully.

Syntax

```
OnClientRegisterFail(UserLogin)
```

Parameters

UserLogin (string)
Specifies the User's login that sends the request.

Example

```
OnClientRegisterFail(UserLogin)
{
	RemoveUser(UserLogin)
}
```

See Also

OnClientRegister(), OnClientRegisterSuccess(), RemoveUser()

OnLineRegisterTrying()

To make VaxTele connect and work with other external SIP servers and IP-Telephony Service providers the function *AddLine()* is used to add SIP account settings as LINE. To register LINE *RegisterLine()* function is used.

During the register process *OnLineRegisterTrying*() event triggers when VaxTele receives SIP response “100, Trying” from other SIP server.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 113) for more details.

Syntax

```
OnLineRegisterTrying(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineRegisterTrying(LineId)
{
}
```

See Also

RegisterLine(), AddLine(), OnLineRegisterFail(), OnLineRegisterSuccess()

OnLineRegisterFail()

VaxTele triggers *OnLineRegisterFail()* event when LINE (SIP account settings) does not register successfully to external SIP server or IP-Telephony service provider.

Syntax

```
OnLineRegisterFail(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineRegisterFail(LineId)
{
}
```

See Also

AddLine(), RegisterLine(), OnLineRegisterTrying(), OnLineRegisterSuccess()

OnLineRegisterSuccess()

VaxTele triggers *OnLineRegisterSuccess()* event when *RegisterLine()* function is call to register a LINE (SIP account settings) to external SIP server or IP-Telephony service provider and LINE registers successfully.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 113) for more details.

Syntax

```
OnLineRegisterSuccess(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineRegisterSuccess(LineId)
{
}
```

See Also

AddLine(), RegisterLine(), OnLineRegisterTrying(), OnLineRegisterFail()

OnLineUnRegisterTrying()

To unregister or disconnect VaxTele to external third party SIP Server the *UnRegisterLine()* is used and during the unregister process *OnLineUnRegisterTrying*() event triggers when VaxTele receives SIP response “100, Trying” from other SIP server.

Syntax

```
OnLineUnRegisterTrying(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineUnRegisterTrying(LineId)
{
}
```

See Also

RegisterLine(), UnRegisterLine(), AddLine(), OnUnLineRegisterFail(), OnUnLineRegisterSuccess()

OnLineUnRegisterFail()

VaxTele triggers *OnLineUnRegisterFail()* event, when a specific LINE fails to get un-registered from external SIP server or IP-Telephony service provider.

Syntax

```
OnLineUnRegisterFail(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineUnRegisterFail(LineId)
{
}
```

See Also

RegisterLine(), UnRegisterLine(), AddLine(), OnUnLineRegisterTrying(), OnUnLineRegisterSuccess()

OnLineUnRegisterSuccess()

VaxTele triggers *OnLineUnRegisterSuccess()* event when Un*RegisterLine()* function is called to un-register/disconnect a LINE (SIP account settings) from external SIP server or IP-Telephony service provider and LINE un-registers successfully.

Syntax

```
OnLineUnRegisterSuccess(LineId)
```

Parameters

LineId (Interger)
Specifies the unique-id for line identification.

Example

```
OnLineUnRegisterSuccess(LineId)
{
}
```

See Also

RegisterLine(), UnRegisterLine(), AddLine(), OnUnLineRegisterTrying(), OnUnLineRegisterFail()

OnIncomingCallFromLine()

Such event triggers when VaxTele receives call request from third party external SIP server through LINE previously added by the call to *AddLine()* function.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 113) for more details.

Syntax

```
OnIncomingCallFromLine(
                        CallId,
                        LineId,
                        CallFrom,
                        CallTo,
                        FromURI,
                        ToURI,
                        FromIP,
                        FromPort
                        )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

LineId (interger)
: Specifies the unique-id for line identification.

CallFrom (string)
: Specifies the caller-Id or caller phone no.

CallTo (string)
: Specifies the callee-Id or callee phone no.

FromURI (string)
: Specifies the SIP FromURI in incoming SIP call request.

ToURI (string)
: Specifies the SIP ToURI in incoming SIP call request.

FromIP (string)
: Specifies the from IP address.

FromPort (integer)
: Specifies the from port number.

Example

```
OnIncomingCallFromLine(CallId, LineId, CallFrom, CallTo, FromURI, ToURI,
FromIP, FromPort)
 {
 }
```

See Also

AddLine(), RegisterLine(), DialCallToLine()

OnIncomingCallFromUser()

Such event triggers when VaxTele receives call request from a registered user previously added by calling to *AddUser()* function.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.

Syntax

```
OnIncomingCallFromUser(
                        CallId,
                        UserName,
                        CallFrom,
                        CallTo,
                        FromURI,
                        ToURI,
                        FromIP,
                        FromPort
                        )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

UserName (string)
: Specifies the user's login from which the call is received.

CallFrom (string)
: Specifies the caller-Id or caller phone no.

CallTo (string)
: Specifies the callee-Id or callee phone no.

FromURI (string)
: Specifies the SIP FromURI in incoming SIP call request.

ToURI (string)
: Specifies the SIP ToURI in incoming SIP call request.

FromIP (string)
: Specifies the from IP address.

FromPort (integer)
: Specifies the from port number.

Example

```
OnIncomingCallFromUser(CallId, UserName, CallFrom, CallTo, FromURI,
ToURI,  FromIP, FromPort)
{
}
```

See Also

AddUser(),OnClientRegister(), DialCallToUser()

OnIncomingCallAnonymous()

Such event triggers when VaxTele receives call request directly on listen IP address.

Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
OnIncomingCallAnonymous(
                        CallId,
                        CallFrom,
                        CallTo,
                        FromURI,
                        ToURI,
                        FromIP,
                        FromPort
                        )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

CallFrom (string)
: Specifies the caller-Id or caller phone no.

CallTo (string)
: Specifies the callee-Id or callee phone no.

FromURI (string)
: Specifies the SIP FromURI in incoming SIP call request.

ToURI (string)
: Specifies the SIP ToURI in incoming SIP call request.

FromIP (string)
: Specifies the from IP address.

FromPort (integer)
: Specifies the from port number.

Example

```
OnIncomingCallAnonymous(CallId, CallFrom, CallTo, FromURI, ToURI,
FromIP, FromPort)
{
}
```

See Also

DialCallToURI()

OnIncomingCallCancel()

Such event triggers when other SIP party sends request to cancel an unprocessed call.

Syntax

```
OnIncomingCallCancel(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnIncomingCallCancel(CallId)
{
        CloseCall(CallId)
}
```

See Also

DialCallToURI(), CloseCall(), AcceptCall()

OnDialCallConnected()

In VaxTele, call request can be sent by using the following functions;

- DialCallToUser()
- DialCallToLine()
- DialCallToURI()

So, OnDialCallConnected() event triggers, when other party accepts the call request.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
OnDialCallConnected(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnDialCallConnected(CallId)
{
        CreateSession()
        AddCallToSession(CallId, SessionId)

        PlayWaveStartToSession(PlayWaveId, SessionId, True)
}
```

See Also

CreateSession(), CloseCall(), OnAcceptIncomingCallConnected()

OnAcceptIncomingCallConnected()

To accept an incoming call request *AcceptCall()* is used and VaxTele triggers event *OnAcceptIncomingCallConnected()*when the accepted call gets connected.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
OnAcceptIncomingCallConnected(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnAcceptIncomingCallConnected(CallId)
{
        CreateSession()
        AddCallToSession(CallId, SessionId)

        PlayWaveStartToSession(PlayWaveId, SessionId, True)
}
```

See Also

CreateSession(), CloseCall(), OnDialCallConnected()

OnDisconnectCall()

Such event triggers, when VaxTele receives call disconnect request.

Please see **SIP PHONE TO SIP PHONE CALL FLOW** (Page. 106) for more details.
Please see **SIP PHONE TO PSTN CALL FLOW** (Page. 108) for more details.

Syntax

```
OnDisconnectCall(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnDisconnectCall(CallId)
{
        CloseCall(CallId)
}
```

See Also

CreateSession(), CloseCall(), OnDialCallConnected()

OnDialCallTimeOut()

In VaxTele, call request can be sent by using the following functions.

- DialCallToUser()
- DialCallToLine()
- DialCallToURI()

If the other party to which call request is sent does not send any SIP response within TimeOut interval of time then time out occurs and VaxTele triggers *OnDialCallTimeOut()* event.

Syntax

```
OnDialCallTimeOut(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnDialCallTimeOut(CallId)
{
        CloseCall(CallId)
}
```

See Also

DialCallToUser(), DialCallToLine(), DialCallToLine(), CloseCall()

OnAcceptCallTimeOut()

In VaxTele, call request can be accepted by calling *AcceptCall()* function.

Specifies the time in seconds. If the other party does not send accept call response within TimeOut interval of time then time out occurs and VaxTele triggers *OnAcceptCallTimeOut()* event.

Syntax

```
OnAcceptCallTimeOut(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnAcceptCallTimeOut(CallId)
{
        CloseCall(CallId)
}
```

See Also

AcceptCall(), CloseCall()

OnProvisionalResponse()

This event triggers, when the call is dialed by using any of the following functions.

- DialCallToUser()
- DialCallToLine()
- DialCallToURI()

And VaxTele receives SIP provisional responses.

SIP Provisional responses 1xx			
100	Trying	180	Ringing
181	Call Is Being Forwarded	182	Queued
183	Session Progress		

Syntax

```
OnProvisionalResponse(
                        CallId,
                        StatusCode,
                        ReasonPhrase
                        )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

StatusCode (integer)
: SIP response status code (100, 180, 183 etc).

ReasonPhrase (string)
: SIP response reason phrase (Trying, Ringing etc).

Example

```
OnProvisionalResponse(CallId, StatusCode, ReasonPhrase)
{
        SendResponse()
}
```

See Also

SendResponse(), CloseCall(), OnFailureResponse(), OnRedirectionResponse()

OnFailureResponse()

This event triggers when the call is dialed by using any of the following functions.

- DialCallToUser()
- DialCallToLine()
- DialCallToURI()

And VaxTele receives SIP failure responses.

Request Failure 4xx			
400	Bad Request	401	Unauthorized
402	Payment Required	403	Forbidden
404	Not Found	405	Method Not Allowed
406	Not Acceptable	407	Proxy Authentication Required
408	Request Timeout	410	Gone
413	Request Entity Too Large	414	Request-URI Too Long
415	Unsupported Media Type	416	Unsupported URI Scheme
420	Bad Extension	421	Extension Required
423	Interval Too Brief	480	Temporarily Unavailable
481	Call/Transaction Does Not Exist	482	Loop Detected
483	Too Many Hops	484	Address Incomplete
485	Ambiguous	486	Busy Here
487	Request Terminated	488	Not Acceptable Here
491	Request Pending	493	Undecipherable
Server Failure 5xx			
500	Server Internal Error	501	Not Implemented
502	Bad Gateway	503	Service Unavailable
504	Server Time-out	505	Version Not Supported
513	Message Too Large		
Global Failures 6xx			
600	Busy Everywhere	603	Decline
604	Does Not Exist Anywhere	606	Not Acceptable

Syntax

```
OnFailureResponse(
                    CallId,
                    StatusCode,
                    ReasonPhrase
                    )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

StatusCode (integer)
: SIP response status code (100, 180, 183 etc).

ReasonPhrase (string)
: SIP response reason phrase (Trying, Ringing etc).

Example

```
OnFailureResponse(CallId, StatusCode, ReasonPhrase)
{
        SendResponse()
}
```

See Also

SendResponse(), CloseCall(), OnProvisionalResponse(), OnRedirectionResponse()

OnRedirectionResponse()

This event triggers when the call is dialed by using any of the following functions.

- DialCallToUser()
- DialCallToLine()
- DialCallToURI()

And VaxTele receives SIP redirection responses.

Redirection 3xx			
300	Multiple Choices	301	Moved Permanently
302	Moved Temporarily	305	Use Proxy
380	Alternative Service		

Syntax

```
OnRedirectionResponse(
                        CallId,
                        StatusCode,
                        ReasonPhrase,
                        Contact
                        )
```

Parameters

CallId (integer)
: Specifies the unique-id for call identification.

StatusCode (integer)
: SIP response status code (100, 180, 183 etc).

ReasonPhrase (string)
: SIP response reason phrase (Trying, Ringing etc).

Contact (string)
: Specifies the redirected SIP-URI.

Example

```
OnRedirectionResponse(CallId, StatusCode, ReasonPhrase, Contact)
{
        SendResponse()
}
```

See Also

SendResponse(), CloseCall(), OnFailureResponse(), OnProvisionalResponse()

OnPlayWaveToSessionDone()

VaxTele notifies that it has played the entire wave file to a session.

Syntax

```
OnPlayWaveToSessionDone(PlayWaveId, SessionId)
```

Parameters

PlayWaveId (integer)
: Specifies the unique play wave data identification.

SessionId (integer)
: Specifies the unique-id for session identification.

Example

```
OnPlayWaveToSessionDone(PlayWaveId, SessionId)
{
	PlayWaveUnLoadFile(PlayWaveId)
}
```

See Also

PlayWaveLoadFile(), PlayWaveStartToSession()

OnDigitDetectDTMF()

VaxTele notifies about the DTMF digit or key in the call.

Syntax

```
OnDigitDetectDTMF(SessionId, CallId, Digit)
```

Parameters

SessionId (integer)
Specifies the unique-id for session identification.

CallId (integer)
Specifies the unique-id for call identification.

Digit (integer)
Digit value (0 to 9) and # = 10, * = 11

Example

```
OnDigitDetectDTMF(SessionId, CallId, Digit)
{
	if(Digit = 10) // if user press #
	PlayWaveStartToSession()
}
```

See Also

DetectDTMF()

OnSessionLost()

If the session lost is enabled by the previous call to *SessionLost()* function then during the voice session if VaxTele does not receive voice data from any client/call within 6 minutes then it triggers *OnSessionLost()* event

Syntax

```
OnSessionLost(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id for call identification.

Example

```
OnSessionLost(CallId)
{
        CloseCall()
}
```

See Also

SessionLost(), CloseCall()

OnTransferCallBlind()

Such event triggers when VaxTele receives blind transfer request from client/call.

Syntax

```
OnTransferCallBlind(
                    CallId,
                    Transferee,
                    TransferTo,
                    TransferToURI
                    )
```

Parameters

Transferee (string)
: Specifies the transferee user name.

TransferTo (string)
: Specifies the transfer to user name.

TransferTo (string)
: Specifies the complete transfer to SIP URI.

Example

```
OnTransferCallBlind(CallId, Transferee, TransferTo, TransferToURI)
{
        TransferCallAccept()
        DialCallToUser(TransferTo)
}
```

See Also

TransferCallAccept(), DialCallToUser()

OnTransferCallAttend()

Such event triggers when VaxTele receives attend transfer request from client/call.

Syntax

```
OnTransferCallAttend(CallId , TransferToCallId)
```

Parameters

TransferToCallId (integer)
Specifies the unique-id to identify a call.

Example

```
OnTransferCallAttend(CallId , TransferToCallId)
{
TransferCallAccept()
}
```

See Also

TransferCallAccept()

OnHoldCall()

Such event triggers, when VaxTele receives call on-hold request from any client/call.

Syntax

```
OnHoldCall(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id to identify a call.

Example

```
OnHoldCall(CallId)
{
        AddCallToSession()    // add to hold music session.
}
```

See Also

OffHoldCall(), CreateSession(), PlayWaveStartToSession()

OffHoldCall()

Such event triggers, when VaxTele receives call off-hold request from any client/call.

Syntax

```
OffHoldCall(CallId)
```

Parameters

CallId (integer)
Specifies the unique-id to identify a call.

Example

```
OffHoldCall(CallId)
{
        RemoveCallToSession()       // add to hold music session.
}
```

See Also

OnHoldCall(), CreateSession(), PlayWaveStartToSession()

OnVaxTeleTick()

SetVaxTeleTick() set a time internally and triggers the tick event *OnVaxTeleTick()* after that specific time. It works just like *SetTimer()* win32 API.

SetVaxTeleTick() function with event *OnVaxTeleTick()* can be used for call processing in queues, DTMF press wait time etc.

Syntax

```
OnVaxTeleTick(TickId)
```

Parameters

TickId (integer)
Specifies the unique-id to identify a TickId.

Example

```
OnVaxTeleTick(TickId)
{
}
```

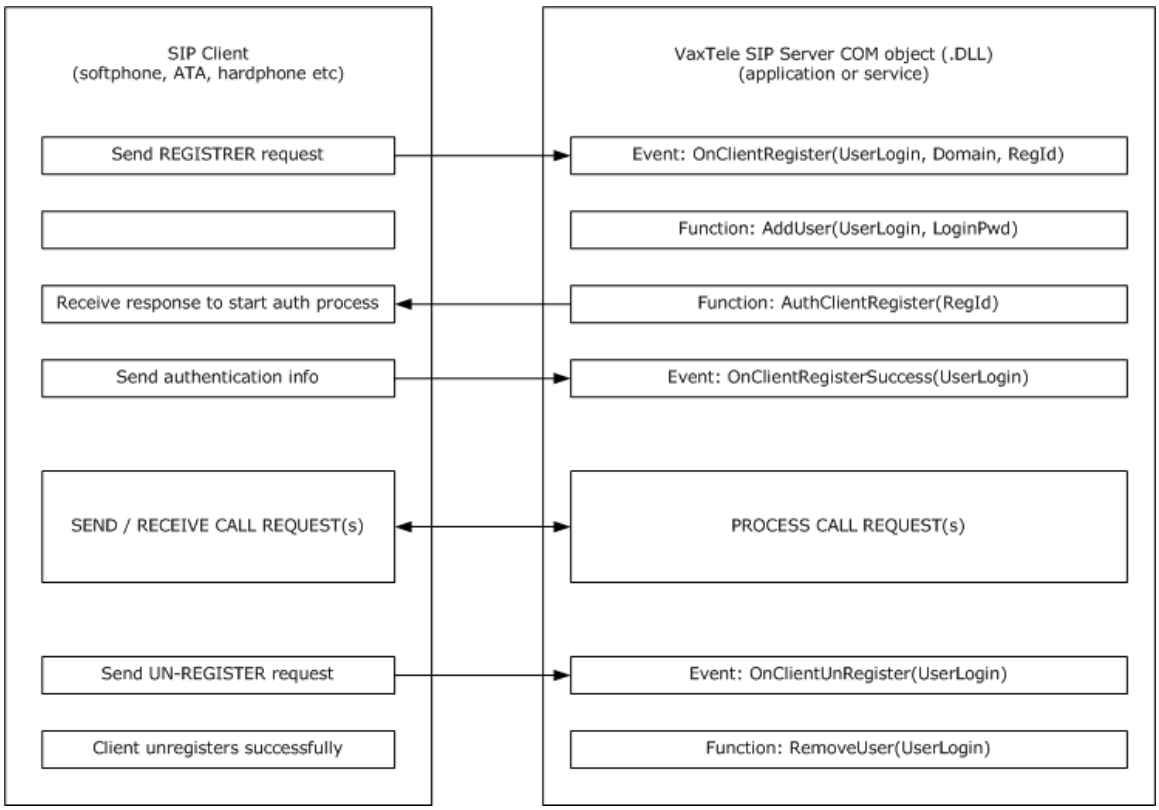
See Also

SetVaxTeleTick()

LIST OF ERROR CODES

200	Fail to initialize RTP Socket.
201	Fail to allocate RTP port or RTP listen IP is not correct.
202	Fail to create RTP task manager.
203	Fail to start media thread.
204	Fail to create RTP communication manager.
205	Unable to use RTP communication manager.
206	Unable to open file.
207	Unable to read file.
208	Wave file format is not correct, please use 8000Hz, 16bit, mono uncompressed wave file.
209	Invalid play wave ID.
210	Fail to start recording manager.
211	Call is not in the voice session.
212	Fail to initialize VaxTele object.
213	Error to create SDP body.
214	Error to create INVITE request.
215	Error to initialize SIP communication layer.
216	Error to open SIP listen port or SIP listen IP is not correct.
217	Error to create SIP task manager.
218	Error to create REGISTER request.
219	Error to create UN-REGISTER request.
220	Error to create BYE request.
221	Error to create CANCEL request.
222	Call-Id is not valid or call does not exist.
223	Session-Id is not valid or session does not exist.
224	Login does not exist.
225	Login length is not correct.
226	Line does not exist.
227	Cann't send SIP response.
228	SIP response code is not valid, please check SIP RFC 3261.
229	Error to create SIP message.
230	Line already exist.
231	Reg-Id is not correct or does not exist.
232	Error to create SIP reponse.
233	User is not registered.
234	Invalid codec OR there is no codec found for voice streaming.
235	Invalid DTMF type.
236	Other party does not support RFC2833 DTMF digits.
237	Error to create REFER request to transfer the call.
238	Error to create event handler.
239	Invalid license key.
240	Invalid digit for DTMF.
241	Invalid proxy URI.
242	Line is not registered.

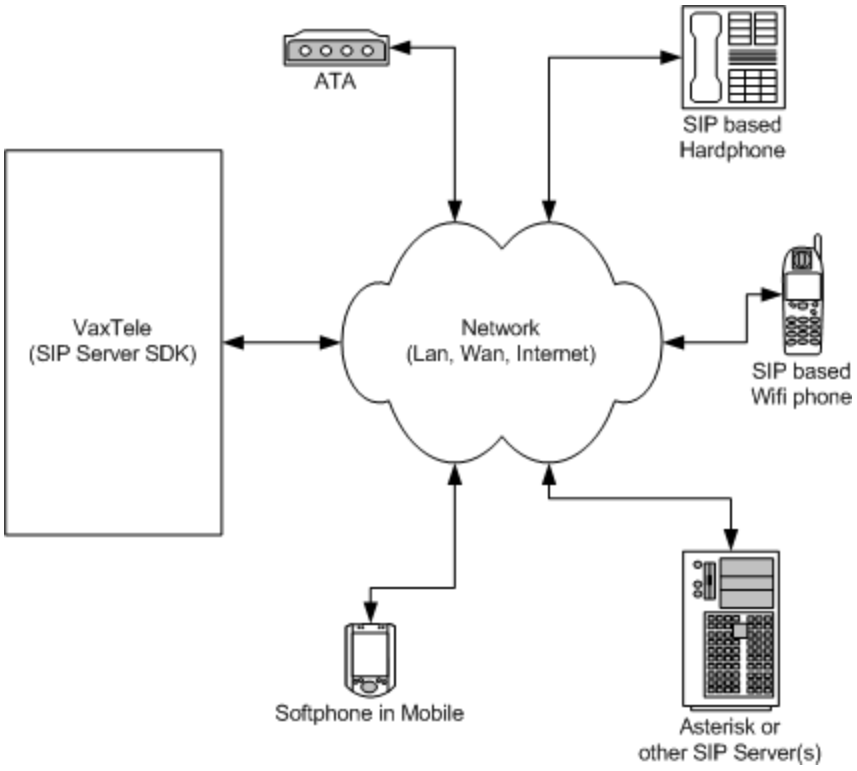
SIP PHONE REGISTRATION FLOW



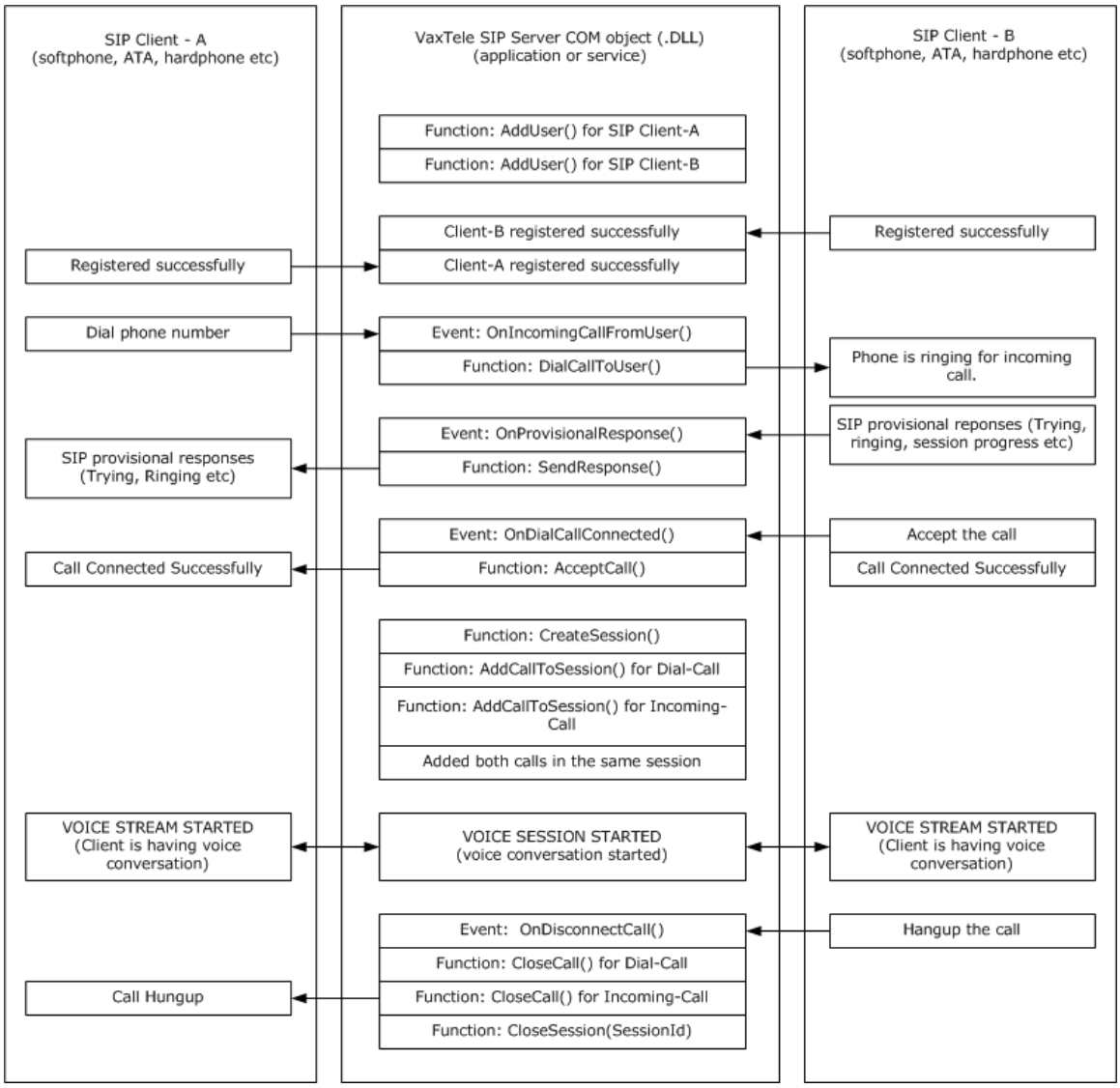
SIP PHONE TO SIP PHONE CALL FLOW

VaxTele can also be used to develop SIP client to SIP client call services. There are many SIP based softphone, hardphone and device are available in the market, which can be connected to VaxTele SIP server as SIP client to dial and receive SIP client to SIP client VoIP calls.

Even other SIP server (asterisk, OpenSer etc.) can also be connected as SIP client to forward the calls to VaxTele SIP Server.



CALL BETWEEN TWO SIP CLIENTS



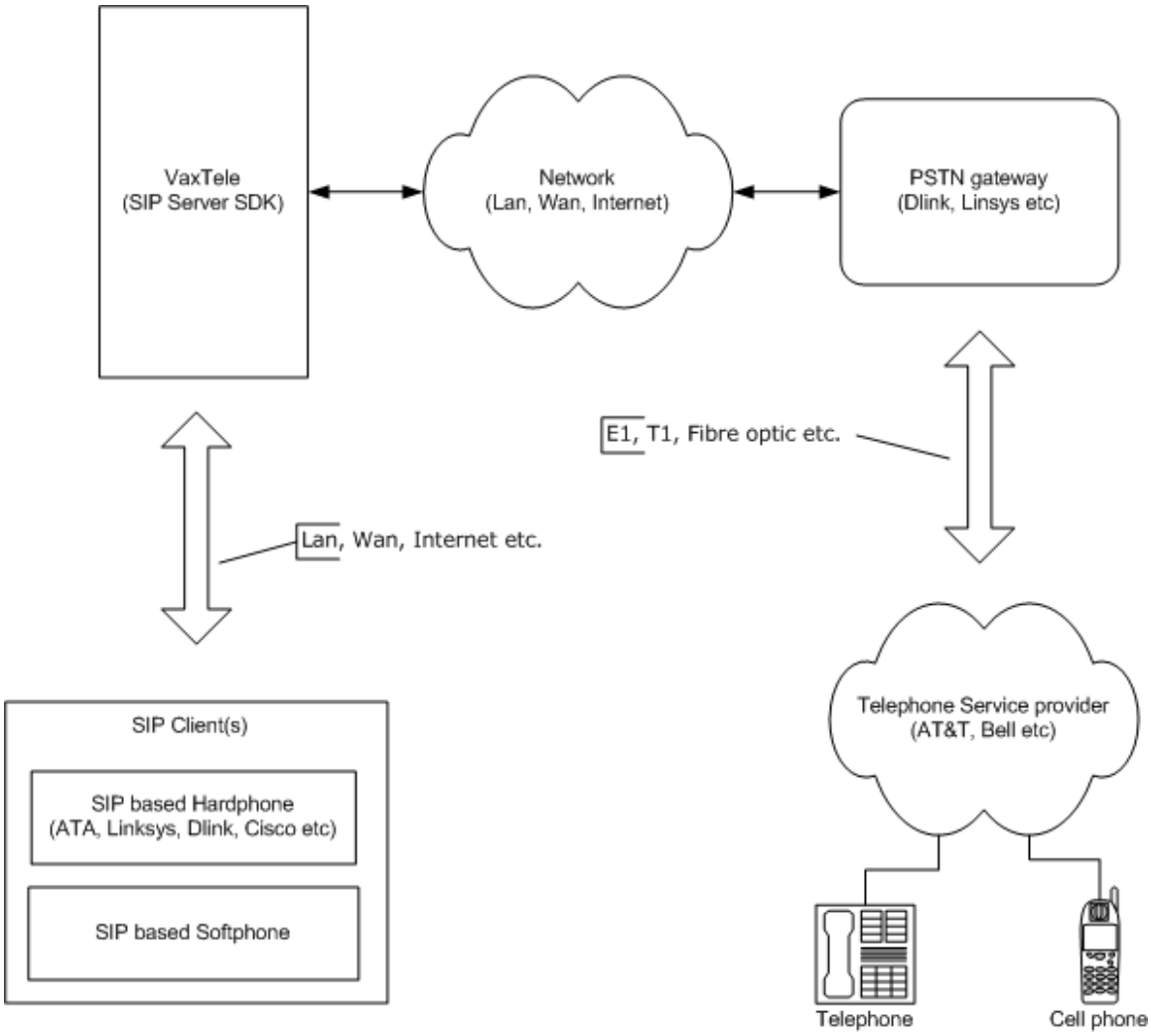
SIP PHONE TO PSTN CALL FLOW

VaxTele can be used to develop many services and products by connecting it to the other SIP based PSTN gateways and ITSP (IP-Telephony service providers).

1. How to connect VaxTele to PSTN network to dial and receive calls to mobile and other telephone numbers.
2. How to connect VaxTele to IP-Telephony service provider (ITSP) to dial and receive calls from mobile and other telephone numbers.

CONNECT VAXTELE WITH PSTN NETWORK

There are many SIP based PSTN gateways are available in the market (Dlink, Linksys, Cisco quantum etc), you may search on Internet with "SIP based PSTN gateways". Those gateways can be used to connect VaxTele SIP server to the PSTN (E1, T1, etc) Network.
PSTN network is use to dial/receive phone calls to other telephone and mobile numbers.



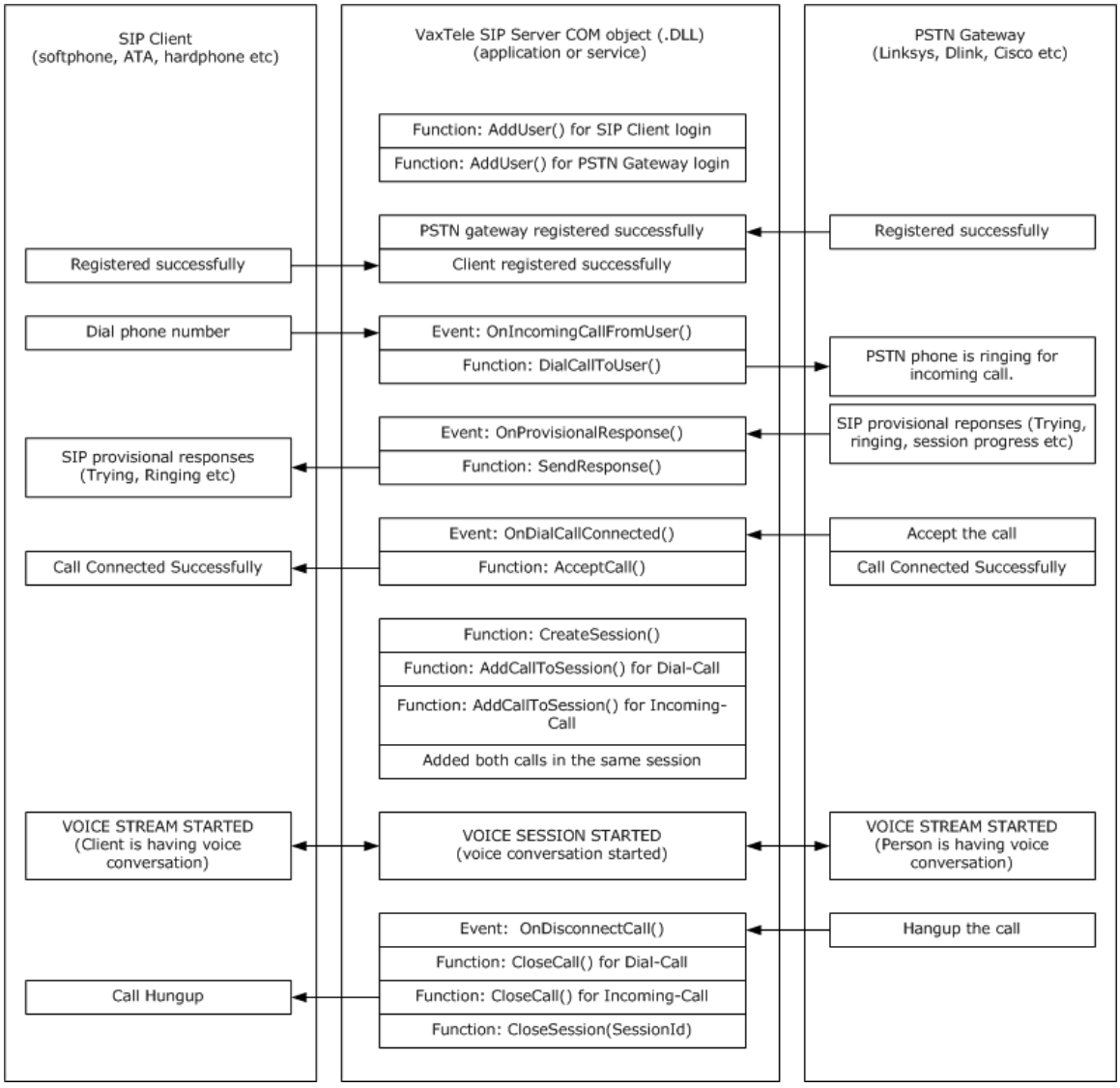
PSTN gateway can be configured;

1. PSTN gateway as SIP client
2. PSTN gateway as direct IP to IP communication

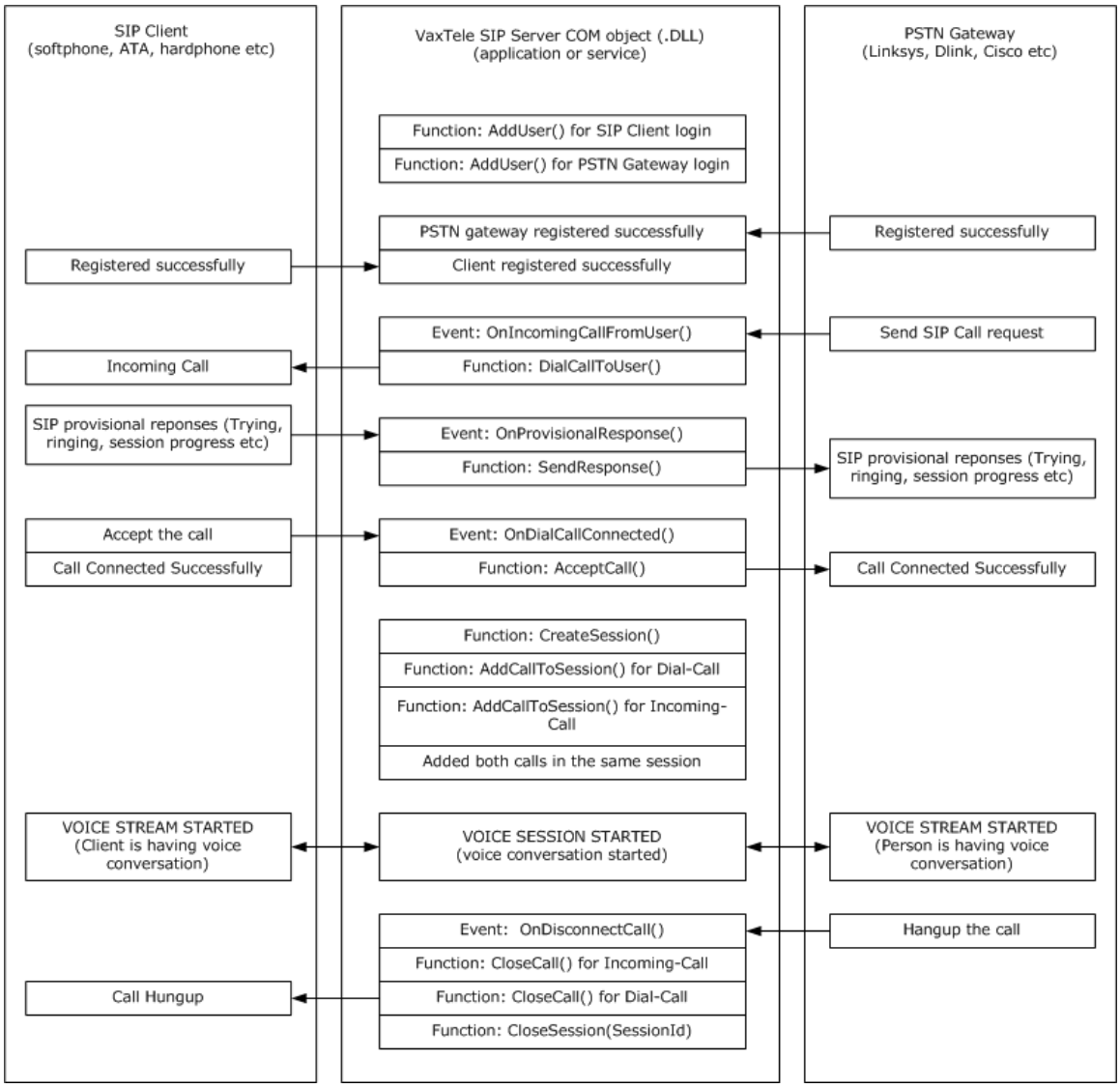
PSTN GATEWAY AS SIP CLIENT

If you configure your PSTN gateway in a way that it acts as SIP client, registers to the VaxTele SIP server and then send and receive call requests.

DIAL CALL TO PSTN GATEWAY



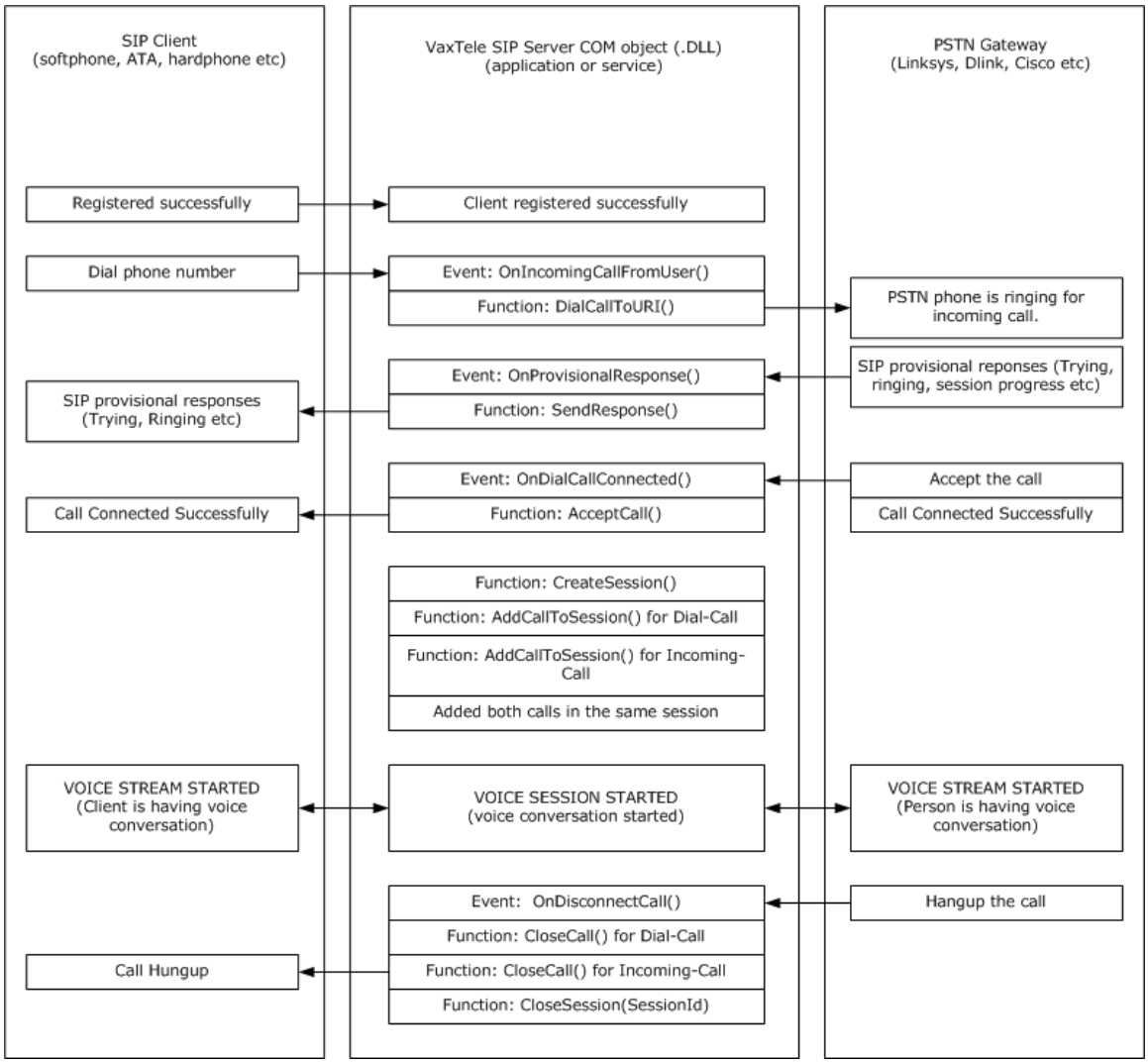
RECEIVE CALL FROM PSTN GATEWAY



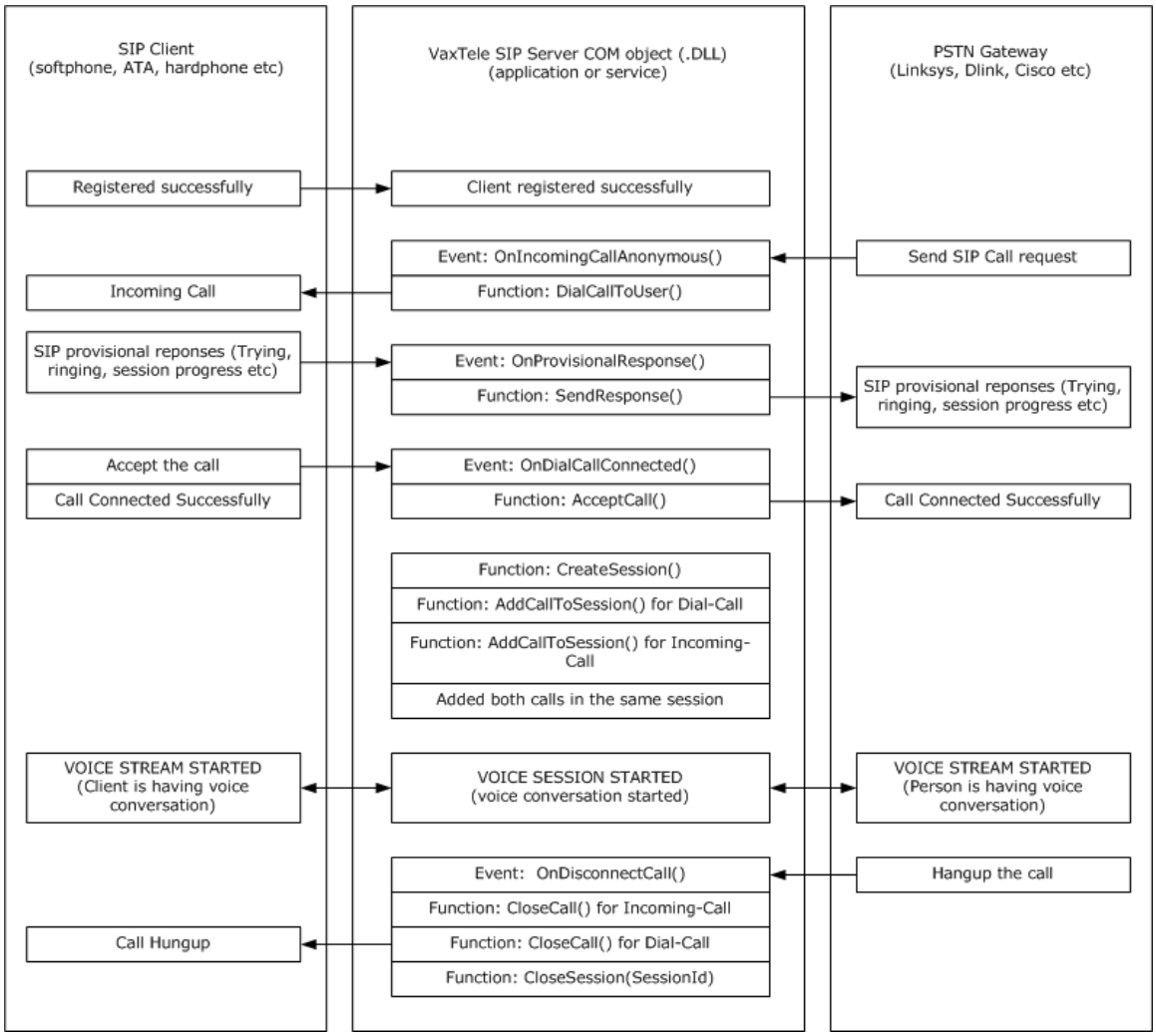
PSTN GATEWAY AS DIRECT IP TO IP COMMUNICATION

If you configure your PSTN gateway in a way that it directly receive call requests on listen IP and send call requests directly to the VaxTele SIP Server IP.

DIAL CALL TO PSTN GATEWAY



RECEIVE CALL FROM PSTN GATEWAY

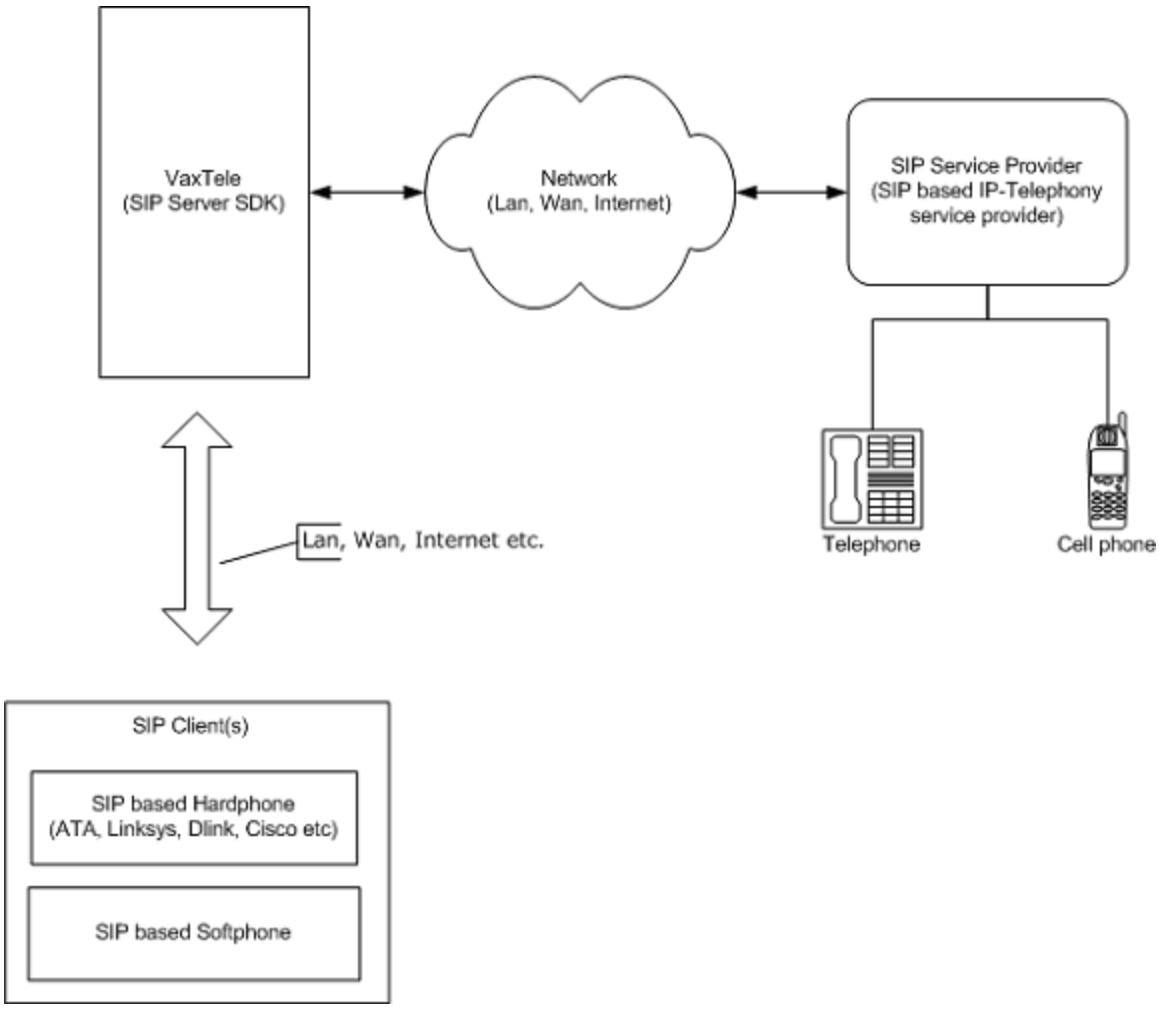


CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)

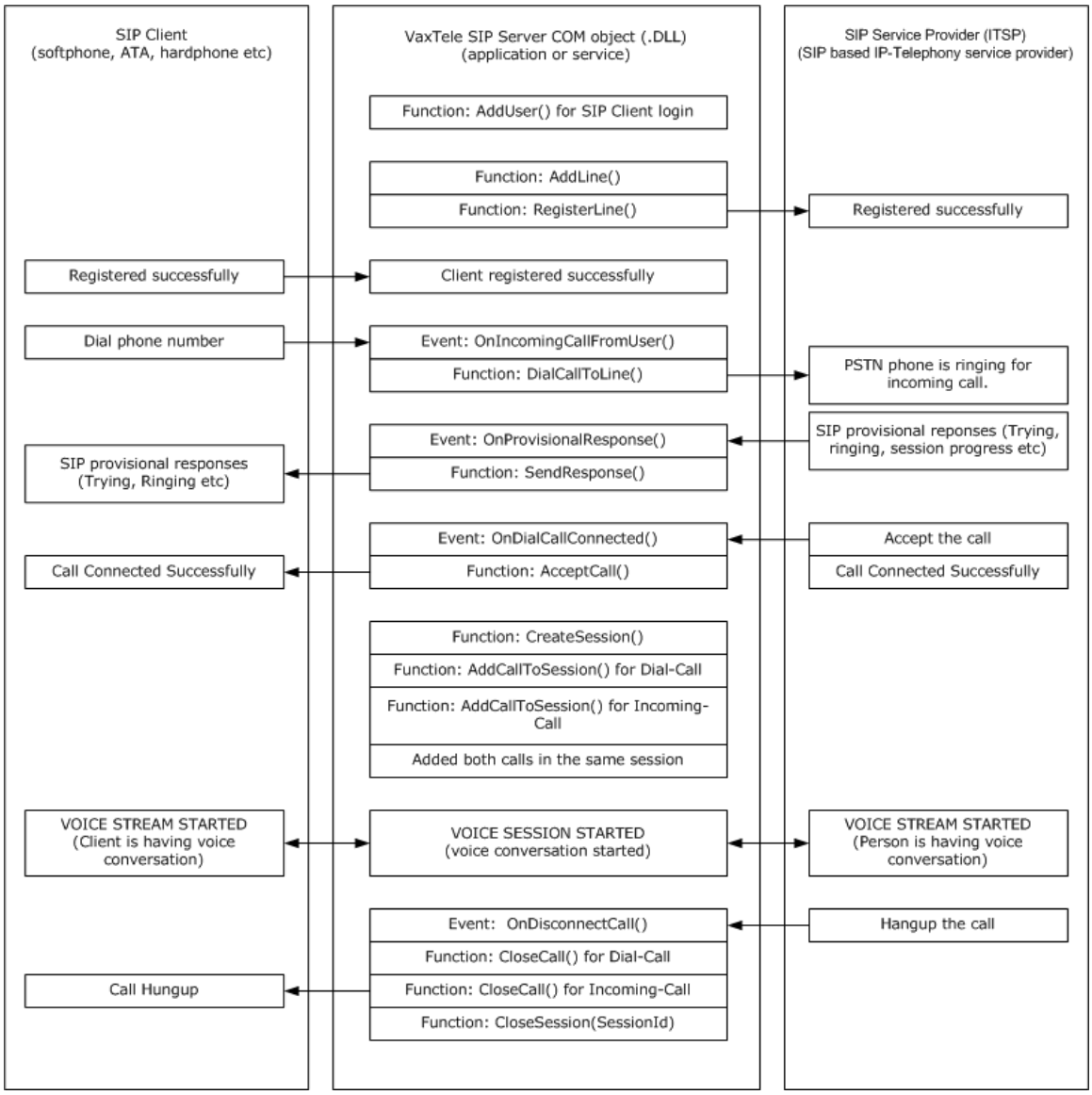
There are many ITSP (IP-Telephony Service provider) can be found on Internet, you can buy minutes or account settings directly from them. Use those settings in VaxTele SIP Server to dial and receive phone calls to mobile & phone numbers.

You may search on Internet with “SIP based service providers”. Some of them are:

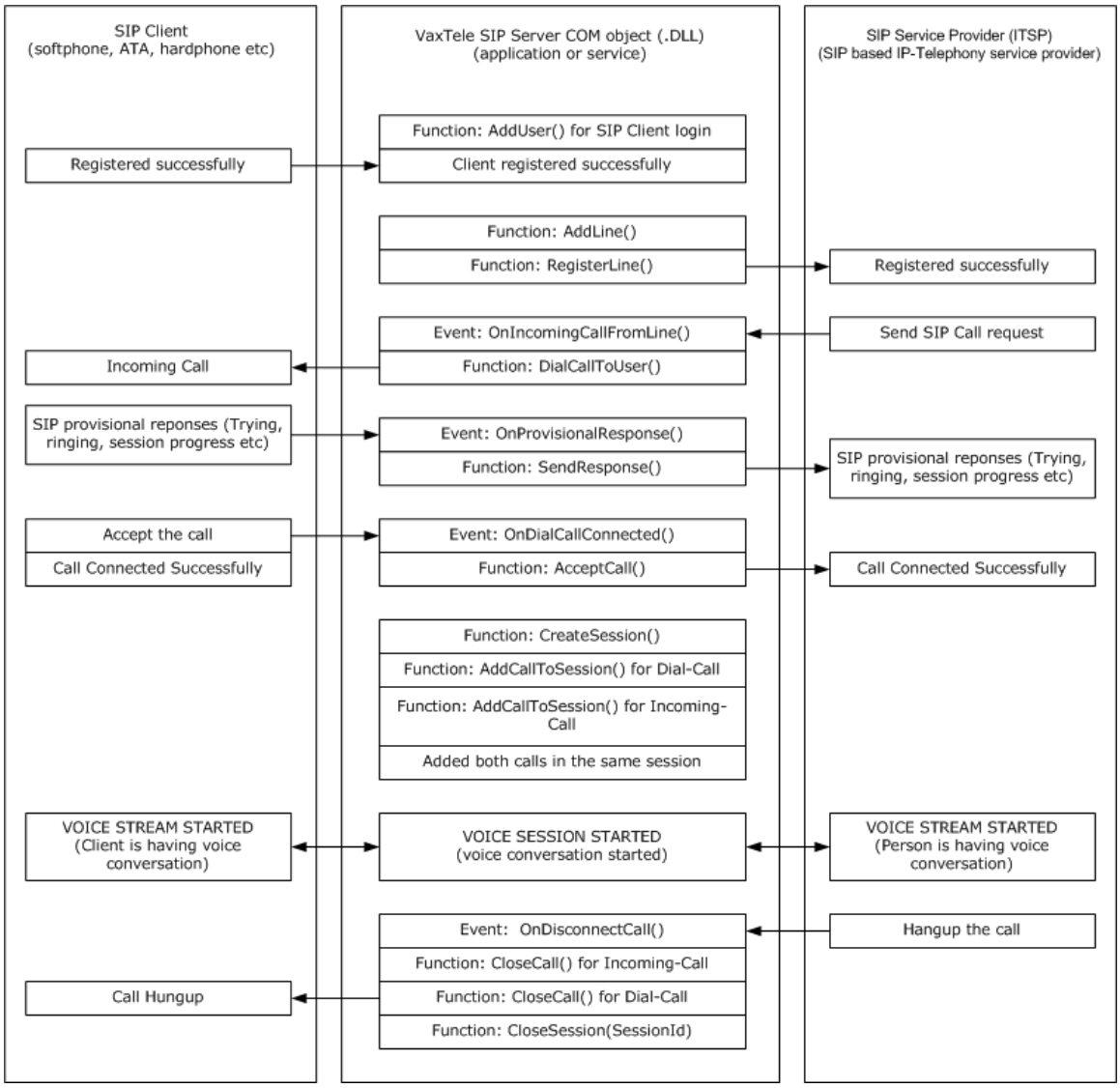
- www.broadvoice.com
- www.voipvoip.com
- www.inphonex.com
- www.didww.com
- www.voxbone.com
- www.verizon.com



DIAL CALL TO SERVICE PROVIDER (ITSP)



RECEIVE CALL FROM SERVICE PROVIDER (ITSP)



REFERENCES

[1] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[3] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.