



## **SIP SERVER SDK v2.4**

**TECHNICAL DOCUMENTATION**

**VERSION 2.0**

## **CONTENTS**

### **INTRODUCTION AND QUICK START ..... 6**

### **EXPORTED FUNCTIONS ..... 7**

SetLicenseKey()	7
Initialize()	8
SetListenPortRangeRTP()	10
AddListenIP()	11
UnInitialize()	12
AddUser()	13
RemoveUser()	15
RegisterUserExpiry()	16
AcceptRegister()	17
RejectRegister()	18
AuthRegister()	19
AddLine()	20
RemoveLine()	22
RegisterLine()	23
UnRegisterLine()	24
AddDirectProxySIP()	25
RemoveDirectProxySIP()	28
AcceptCallSession()	29
RejectCallSession()	31
CloseCallSession()	32
OpenCallSession()	33
AcceptTransferBlind()	35
AcceptTransferConsult()	37
RejectTransfer()	39
LoadWaveFile()	40
LoadWavePCM()	41
UnLoadWaveFile()	43
PlayWaveStartToCallSession()	44
PlayWaveStopToCallSession()	46
RecordWaveStartToCallSession()	47
RecordWaveStopToCallSession()	48
RecordWavePauseToCallSession()	49
LoadMusicHold()	50
UnLoadMusicHold()	51
AcceptOnHoldRequest()	52
AcceptOffHoldRequest()	53
AcceptChatMessage()	54
RejectChatMessage()	55
SendChatMessageText()	56
AcceptChatStatusSubscribe()	57
RejectChatStatusSubscribe()	59
OpenConferenceRoom()	60
CloseConferenceRoom()	61
AddCallSessionToConferenceRoom()	62
RemoveCallSessionFromConferenceRoom()	64

PlayWaveStartToConferenceRoom()	65
PlayWaveStopToConferenceRoom()	66
RecordWaveStartToConferenceRoom()	67
RecordWaveStopToConferenceRoom()	68
RecordWavePauseToConferenceRoom()	69
VoiceSessionLost()	70
SendInfoVM()	71
DiagnosticLogSIP()	72
StartVaxTeleTick()	73
StopVaxTeleTick()	74
SetUserAgentName()	75
GetUserAgentName()	76
SetSessionNameSDP()	77
GetSessionNameSDP()	78
SendDTMFCallSession()	79
DetectDigitDTMF()	80
GetVaxObjectError()	83
DialToneToCallSession()	85
CallSessionSwapCalls()	86
MergeCallSession()	87
SetUserData()	89
GetUserData()	90
SetLineData()	91
GetLineData()	92
SetDirectProxyData()	93
GetDirectProxyData()	94
SetConferenceRoomData()	95
GetConferenceRoomData()	96
SetCallSessionData()	97
GetCallSessionData()	98
VideoCOMM()	99
GetCallSessionTxCodec()	100
GetCallSessionRxCodec()	101
CallSessionMuteVoice()	102
BusyLampSubscribeAccept()	103
BusyLampSubscribeReject()	104
AddCustomHeader()	105
RemoveCustomHeader()	106
RemoveCustomHeaderAll()	107
CallSessionDetectAMD()	108
CallSessionSendStatusResponse()	110
TrvsUserStart()	112
TrvsUserNext()	113
TrvsLineStart()	114
TrvsLineNext()	115
TrvsDirectProxyStart()	116
TrvsDirectProxyNext()	117
TrvsCallSessionStart()	118
TrvsCallSessionNext()	119
TrvsConferenceRoomStart()	120
TrvsConferenceRoomNext()	121
GetUserCount()	122
GetLineCount()	123

GetDirectProxyCount()	124
GetConferenceRoomCount()	125
GetCallSessionCount()	126
CustomListOpen()	127
CustomListClose()	128
CustomListAddData()	129
CustomListRemoveData()	130
CustomListFindData()	131
CustomListCount()	132
CustomListReset()	133
CustomListTrvsStart()	134
CustomListTrvsNext()	135

## **EXPORTED EVENTS ..... 136**

OnVaxErrorLog()	136
OnLineRegisterTrying()	137
OnLineRegisterFailed()	138
OnLineRegisterSuccess()	139
OnLineUnRegisterTrying()	140
OnLineUnRegisterFailed()	141
OnLineUnRegisterSuccess()	142
OnUnRegisterUser()	143
OnRegisterUser()	144
OnRegisterUserSuccess()	145
OnRegisterUserFailed()	146
OnChatMessageText()	147
OnChatMessageTyping()	149
OnChatStatusSubscribe()	151
OnChatMessageSuccess()	153
OnChatMessageFailed()	154
OnChatMessageTimeout()	155
OnCallSessionOpen()	156
OnCallSessionConnecting()	158
OnCallSessionFailed()	159
OnCallSessionConnected()	160
OnCallSessionLost()	161
OnCallSessionHangup()	162
OnCallSessionTimeout()	163
OnCallSessionCancelled()	164
OnCallSessionOnHold()	165
OnCallSessionOffHold()	166
OnCallSessionTransferBlind()	167
OnCallSessionTransferConsult()	168
OnCallSessionTransferring()	170
OnCallSessionTransferTimeout()	174
OnCallSessionTransferred()	175
OnDetectedDigitDTMF()	176
OnOutgoingDiagnosticLog()	177
OnIncomingDiagnosticLog()	178
OnVaxTeleTick()	179
OnSendTimeoutVM()	180
OnSendSucessVM()	181

OnCallSessionDialToneStart()	182
OnCallSessionDialToneStop()	183
OnCallSessionPlayWaveDone()	184
OnConferenceRoomPlayWaveDone()	185
OnCallSessionOpenCall()	186
OnCallSessionCloseCall()	187
OnCallSessionDetectAMD()	188
OnBusyLampSubscribe()	189
OnBusyLampUnSubscribe()	190
OnBusyLampSubscribeSuccess()	191
OnBusyLampSubscribeFailed()	192
OnCallSessionCreated()	193
OnCallSessionClosed()	194
OnCallSessionRecordedPCM()	195
OnConferenceRoomRecordedPCM()	196

## **SIP CLIENT REGISTRATION FLOW..... 197**

## **SIP PHONE TO SIP PHONE CALL FLOW ..... 198**

CALL BETWEEN TWO SIP CLIENTS	198
------------------------------	-----

## **SIP PHONE TO PSTN CALL FLOW ..... 200**

## **CONNECT VAXTELE WITH PSTN NETWORK ..... 200**

PSTN GATEWAY AS SIP CLIENT	201
DIAL CALL TO PSTN GATEWAY	201
RECEIVE CALL FROM PSTN GATEWAY	202
PSTN GATEWAY AS DIRECT IP TO IP COMMUNICATION	203
DIAL CALL TO PSTN GATEWAY	203
RECEIVE CALL FROM PSTN GATEWAY	204

## **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)..... 205**

DIAL CALL TO SERVICE PROVIDER (ITSP)	206
RECEIVE CALL FROM SERVICE PROVIDER (ITSP)	207

## **LIST OF SIP RESPONSES (SIP RFC 3261) ..... 208**

## **CALL-SESSION..... 209**

Call-Session with two calls (From-Call & To-Call)	209
Call-Session with One call (From-Call)	209
Call-Session with One call (To-Call)	209

## **FUNCTION AND CALL SESSION..... 211**

## **INTRODUCTION AND QUICK START**

The VaxTele COM (Component Object Model) component VaxTeleServerCOM.dll. It is a collection of functions and events that allow you to develop SIP (session initiation protocol) based server, IP-PBX, IVR & Telemarketing systems and other IP-Telephony related services.

The VaxTele COM component (VaxTeleServerCOM.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**

### **SetLicenseKey()**

The trial version of VaxVoIP SDK has trial period limitation of 30 days, so a license key is required after 30 days to avoid evaluation message box. License keys are delivered to customers on order.

The SetLicenseKey( ) method is used to make the trial version working as registered version without expiry and trial period limitation.

### **Syntax**

```
SetLicenseKey(LicenseKey)
```

### **Parameters**

LicenseKey (string)

The value of this parameter is license key provided by the company.

### **Return Value**

No return value.

### **Example**

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

### **See Also**

Initialize(), GetVaxObjectError()

## 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(  
    DomainRealm,  
    SIPListenIP,  
    SIPListenPort,  
    RTPListenIP,  
    RTPListenPort  
)
```

## Parameters

### DomainRealm (string)

This parameter value is internally used to create SIP URI(s). It is used as "realm" field in SIP packets during the authentication of SIP clients (softphone, hardphone etc) and call processing.

This 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 DomainRealm is sip.vaxtele.com then VaxTele creates SIP URI sip:username@sip.vaxtele.com  
But if value of DomainRealm 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 an IP-address should be assigned to DomainRealm.

### SIPListenIP (string)

The value of this parameter 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)

The value of this parameter specifies the port number for SIP server to receive SIP requests. Standard SIP listen port is 5060

### RTPListenIP (string)

The value of this parameter specifies the IP address for VaxTele integrated SIP server to receive voice streams. If multiple IP addresses are assigned to the computer on which SIP server is running then SIPListenIP and RTPListenIP can be different.



**RTPListenPort (integer)**

The RTPListenPort specifies the start port number for the range of RTP ports allocation. The allocation starts from the parameter value and increments for even number as for RTP compliance, RTP port number must be an even number.

This parameter can also be assigned value -1 which let VaxTele to randomly choose a RTP port and allocates to the call for voice streaming.

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 SIP 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**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**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", 6066)
Initialize("vaxtele.com", "209.237.151.17", 5060, "209.237.151.18", -1)
```

**See Also**

UnInitialize(), GetVaxObjectError(), SetListenPortRangeRTP()

**SetListenPortRangeRTP()**

The SetListenPortRangeRTP() sets the given range, so that VaxTele allocates/opens the listen RTP port within that range.

Note: According to the SIP 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.

**Syntax**

```
boolean SetListenPortRangeRTP(ListenStartPort, ListenEndPort)
```

**Parameters**

ListenStartPort (integer)

The value of this parameter specifies starting value of the range.

ListenEndPort (integer)

The value of this parameter specifies end value of the range.

**Return Value**

The function returns a non-zero value on its successful execution otherwise 0, a specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
Result = Initialize("sipsdk.com", "9.7.11.17", 5060, "9.7.11.17", -1)
if(Result == 0) GetVaxObjectError()

SetListenPortRangeRTP(8088, 32406)
```

**See Also**

Initialize(), GetVaxObjectError()

## AddListenIP()

The AddListenIP() function is used to add additional listen IP addresses to VaxTele integrated SIP server application.

When VaxTele integrated application is behind the Firewall/router/NAT and port forwarding is enabled from the router's end. In this scenario, initialize VaxTele with the private IP Address assigned to the computer and add public IP address assigned to the router by using AddListenIP() function.

When multiple IP addresses are assigned to a computer and requires VaxTele to work with those IP addresses.

### Syntax

```
boolean AddListenIP(SIPListenIP, RTPListenIP)
```

### Parameters

SIPListenIP (string)

This parameter specifies the listen IP to be used for SIP packets.

RTPListenIP (string)

This parameter specifies the listen IP to be used for voice streaming.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
Result = Initialize("sipsdk.com", "192.168.0.25", 5060, "192.168.0.25", -1)
if(Result == 0) GetVaxObjectError()

AddListenIP("66.77.88.99", "66.77.88.99")
SetListenPortRangeRTP(8000, 32406)
```

Run VaxTele behind Firewall/router, from the router settings;

- Forward inbound UDP ports 8000 to 32406
- Enable outbound UDP ports 1024 to 65535
- 192.168.0.25 is the IP address assigned to the computer behind router.
- 66.77.88.99 is the IP address assigned to the router.

### See Also

Initialize(), SetListenPortRangeRTP(), GetVaxObjectError()

**UnInitialize()**

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

**Syntax**

```
UnInitialize()
```

**Parameters**

No parameters.

**Return Value**

No return value.

**Example**

```
UnInitialize()
```

**See Also**

Initialize()

## AddUser()

The AddUser() function add users to the SIP server developed by VaxTele COM component. VaxTele COM (VaxTeleServerCOM.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 with SIP server developed by using VaxTele COM component.

## Syntax

```
boolean AddUser(  
    UserName,  
    Password,  
    CodecList,  
)
```

## Parameters

UserName (string)

This parameter value specifies the user's login.

Password (string)

This parameter value specifies the password of user's login.

CodecList (string)

This parameter value specifies the list of voice 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 codecs for the call-request are G729, G711a, iLBC and G711u.

Highest priority codec is 4 (G729) and lowest priority codec is 3 (G711u).

## Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

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

Result = AddUser("9090", "123", "01")
if(Result == 0) GetVaxObjectError()
```

**See Also**

RemoveUser(), GetVaxObjectError()

**RemoveUser()**

The RemoveUser() function removes the provided user previously added by AddUser() function.

**Syntax**

```
RemoveUser(UserName)
```

**Parameters**

UserName (string)

This parameter value specifies the user's login.

**Return Value**

No return value.

**Example**

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

**See Also**

AddUser()

## RegisterUserExpiry()

The RegisterUserExpiry() function configures that which value of Expiration Interval will be added by VaxTele during SIP client registration process.

If -1 value is set then during the registration process, VaxTele accepts the Expiration Interval requested by the SIP client. Otherwise SIP Server ignores the Expiration Interval requested by the SIP client and sends the time, which is set by calling RegisterUserExpiry() function.

If RegisterUserExpiry() function is not called then default value 30 seconds is sent to the SIP client(s).

**Note: In SIP packet Expires header designates the Expiration Interval of the registration.**

### Syntax

```
boolean RegisterUserExpiry(Expiry)
```

### Parameters

Expiry (integer)

The Expire parameter specifies the time interval in seconds.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
RegisterUserExpiry (1800)  
RegisterUserExpiry(-1)
```

### See Also

AddUser(), OnRegisterUser()



## AcceptRegister()

The AcceptRegister() function accepts the registration request which receives from SIP client.

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

VaxTele triggers OnRegisterUser() event upon receiving a registration request. When AcceptRegister() function is called in this event, it simply accepts the registration request without authorizing the login and password.

AcceptRegister() function skips the login authorization process and accepts the registration request.

Please see **SIP CLIENT REGISTRATION FLOW** (Page. 197) for more details.

### Syntax

```
boolean AcceptRegister(Username)
```

### Parameters

Username (string)

This parameter value specifies the unique user name to identify a specific user.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnRegisterUser(UserLogin, Domain, FromIP, FromPort)
{
    AcceptRegister(UserLogin)
}
```

### See Also

AuthRegister(), RejectRegister(), OnRegisterUser()

## RejectRegister()

The RejectRegister() function rejects the registration request send to VaxTele by SIP client.

VaxTele triggers OnRegisterUser() event upon receiving registration request. When RejectRegister() function is called in this event it simply rejects the registration request

Please see **List of SIP Responses** (Page. 208) for more details.

### Syntax

```
boolean RejectRegister(  
    UserName,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

UserName (string)

This parameter value specifies the unique user name to identify a specific user.

StatusCode (integer)

This parameter specifies SIP response status.

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Unauthorized, Not Found etc).

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnRegisterUser(UserLogin, Domain, FromIP, FromPort)  
{  
    RejectRegister(UserLogin, 404, "Not Found")  
}
```

### See Also

AuthRegister(), AcceptRegister(), OnRegisterUser()

## AuthRegister()

The AuthRegister() function starts the authentication process for a specified user before accepting registration request.

The SIP clients (softphone, hardphone, wifi phone) send SIP register request to connect and register to SIP server(s). When VaxTele receives register request then it triggers OnRegisterUser() event and starts the authentication process by calling AuthRegister() function.

Call AuthRegister() function and then VaxTele;

1. Starts SIP authentication process.
2. Completes authentication process.
3. Accepts registration (register) request.

Please see **SIP CLIENT REGISTRATION FLOW** (Page. 197) for more details.

## Syntax

```
boolean AuthRegister(Username)
```

## Parameters

Username (string)

This parameter value specifies the unique user name to identify a specific user.

## Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

## Example

```
OnRegisterUser(UserLogin, Domain, FromIP, FromPort)
{
    AuthRegister(UserLogin)
}
```

## See Also

AcceptRegister(), RejectRegister(), OnRegisterUser()

## AddLine()

The AddLine() function adds third party SIP server provided account setting as line in VaxTele thus allowing VaxTele to work with other SIP servers as well.

VaxTele is a SIP (session initiation protocol) based server SDK. SIP server software developed by using VaxTele SDK can also be connected to other SIP servers and devices by using SIP protocol and then it can sends/receives call requests to those SIP servers and devices.

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

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

Note: IP-Telephony Service Providers (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.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 205) for more details.

## Syntax

```
boolean AddLine(  
    LineName,  
    DisplayName,  
    UserName,  
    AuthUser  
    AuthPwd,  
    DomainRealm,  
    ProxySIP,  
    OutboundProxy,  
    CodecList  
)
```

## Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

DisplayName (string)

This Parameter value specifies the display name for user which is provided by IP-Telephony service provider or third party SIP server.

UserName (string)

This Parameter value specifies the user name which is provided by IP-Telephony service provider or third party SIP server.

AuthUser (string)

This Parameter value specifies the user Login which is provided by IP-Telephony service provider or third party SIP server.

AuthPwd (string)

This Parameter value specifies the password which is provided by IP-Telephony service provider or third party SIP server.

DomainRealm (string)

This Parameter value is provided by IP-Telephony service provider or third party SIP server.

ProxySIP (string)

This Parameter value is provided by IP-Telephony service provider or third party SIP server.

OutboundProxy (string)

This Parameter value is provided by IP-Telephony service provider or third party SIP server.

CodecList (string)

This parameter value specifies the list of voice codecs to be added in the call request.

Supported codecs:

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

"4213" specifies that supported codec for the call request are G729, G711a, iLBC and G711u.

Highest priority codec is 4 (G729) and lowest priority codec is 3 (G711u).

## Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

## Example

```
AddLine("LineNameA", "8034", "8034", "8034", "9341", "sipsdk.com",  
        "192.168.0.3", "", "0134")
```

## See Also

RemoveLine(), RegisterLine(), UnRegisterLine(), GetVaxObjectError()

**RemoveLine()**

The RemoveLine() function removes the provided line which was previously added by AddLine() function.

**Syntax**

```
RemoveLine(LineName)
```

**Parameters**

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

**Return Value**

No return value.

**Example**

```
RemoveLine("abc")  
RemoveLine("LineName")
```

**See Also**

AddLine()

## RegisterLine()

The RegisterLine() function registers the line to external third party SIP server or IP-Telephony Service Provider (ITSP). VaxTele triggers registration related events during line registration process e.g. OnLineRegisterTrying(), OnLineRegisterSuccess() etc.

The line should be added by AddLine() function prior to registration.

### Syntax

```
boolean RegisterLine(  
    LineName,  
    Expire  
)
```

### Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

Expire (integer)

The Expire parameter specifies the time interval (in seconds) after which the registration with server will be refreshed consequently server will remain updated about the present client status.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
RegisterLine("abc", 1800)  
RegisterLine("LineName", 3600)
```

### See Also

UnRegisterLine(), OnLineRegisterSuccess(), OnLineRegisterTrying(), AddLine(), OnLineRegisterFailed(), GetVaxObjectError()

## UnRegisterLine()

The UnRegisterLine() function unregisters the provided line which was registered by RegisterLine() function. VaxTele triggers unregister process related events during line unregistration process e.g. OnLineUnRegisterTrying(), OnLineUnRegisterSuccess() etc.

### Syntax

```
boolean UnRegisterLine(LineName)
```

### Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
UnRegisterLine("abc")  
UnRegisterLine("2" )
```

### See Also

RegisterLine(), AddLine(), OnLineUnRegisterSuccess(), GetVaxObjectError()  
OnLineUnRegisterTrying(), OnLineUnRegisterFailed()

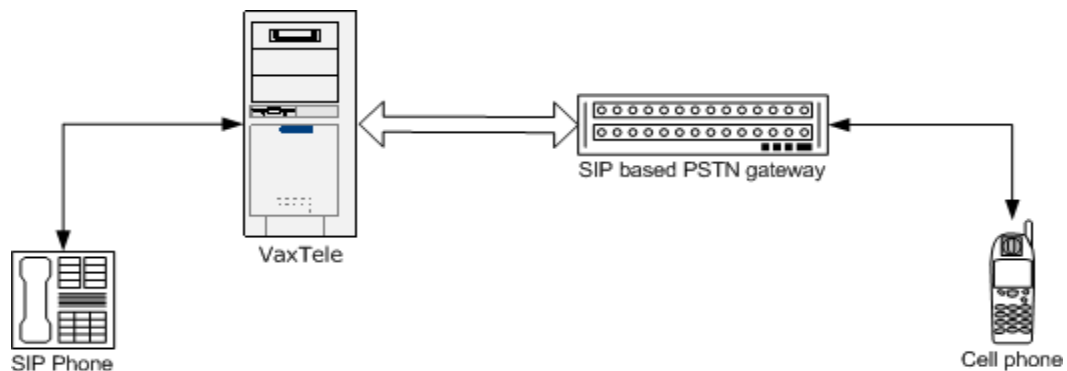


### AddDirectProxySIP()

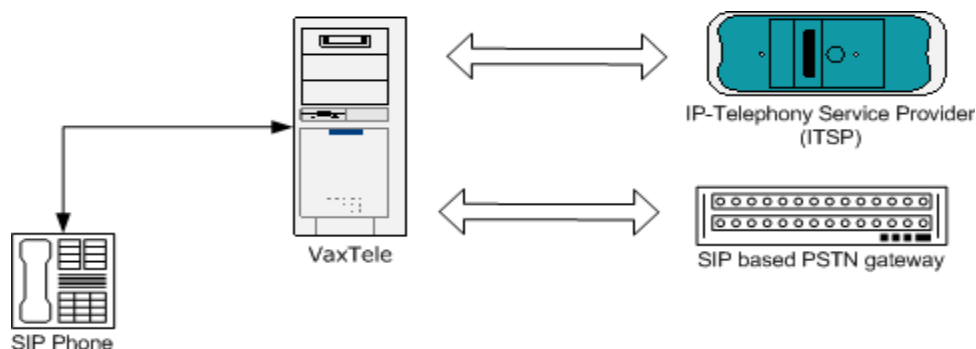
The AddDirectProxySIP() function with other functions AcceptCallSession(), OpenCallSession() and AcceptTransferBlind() provide functionality to send call and other SIP requests directly to the IP and port of remote SIP devices and SIP Servers.

AddDirectProxySIP() function adds direct proxy which used as PeerType and PeerName in AcceptCallSession(), OpenCallSession() and AcceptTransferBlind() functions.

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 AddDirectProxySIP() can be used to configure VaxTele integrated SIP Server 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 the IP & Port of route's SIP servers and SIP devices. AddDirectProxySIP () can also be used in such case.



## Syntax

```
boolean AddDirectProxySIP(  
    ProxyName,  
    AuthLogin,  
    AuthPwd,  
    DomainRealm,  
    ProxyIP,  
    ProxyPort,  
    CodecList  
)
```

## Parameters

ProxyName(string)

This parameter specifies unique direct proxy name.

AuthLogin(string)

This Parameter value specifies the user Login to SIP server or SIP devices where call request will be send.

AuthPwd(string)

This Parameter value specifies the user password to SIP server or SIP devices where call request will be send.

DomainRealm(string)

This parameter specifies the domain realm of the SIP server or SIP devices where call request will be send.

ProxyIP(string)

This parameter specifies the IP of the SIP server or SIP devices where call request will be send.

ProxyPort(integer)

This parameter specifies the proxy of the SIP server or SIP devices where call request will be send.

CodecList(string)

This parameter value specifies the list of voice codecs to be added in the call request.

Supported codecs:

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

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

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling `GetVaxObjectError()` method.

**Example**

```
AddDirectProxySIP("DirectITSP", "8003", "8004", "sip.sdk.com",  
                  "192.168.0.20", 5060, "4213")
```

**See Also**

`RemoveDirectProxySIP()`, `GetVaxObjectError()`

**RemoveDirectProxySIP()**

The RemoveDirectProxySIP () function removes the provided direct proxy which was previously added by AddDirectProxySIP() function.

**Syntax**

```
RemoveDirectProxySIP(ProxyName)
```

**Parameters**

ProxyName(string)

This parameter specifies unique direct proxy name.

**Return Value**

No return value.

**Example**

```
RemoveDirectProxySIP("DirectITSP ")
```

**See Also**

AddDirectProxySIP(), GetVaxObjectError()

## AcceptCallSession()

The AcceptCallSession() function accepts an incoming call and generates an outgoing call that establish a connection between From-Peer and To-Peer.

In VaxTele server the Call-Session refers to collection of one or two calls. The participating calls in a Call-Session are named as From-Call and To-Call. A Call-Session may have one call i.e. From-Call (incoming call) or To-Call (outgoing call) or two calls i.e. From-Call and To-Call (incoming and outgoing call).

Please see **CALL-SESSION** (Page. 209) for more details.

The OnCallSessionOpen() event triggers when VaxTele based SIP Server receives an incoming call request which in turn call AcceptCallSession() method to process the incoming call request accordingly.

VaxTele server SDK can also be used to develop IVR system, in this case server only accepts incoming call and when a call gets connected then plays a wave file and waits for DTMF digit from the caller.

## Syntax

```
boolean AcceptCallSession(  
    SessionId,  
    CallerName,  
    CallerId,  
    DialNo,  
    ToPeerType,  
    ToPeerName,  
    Timeout  
)
```

## Parameters

SessionId (integer)

This parameter specifies a unique identification of established connection between From-Peer and To-Peer.

CallerName (string)

This parameter specifies the caller name.

CallerId (string)

This parameter specifies a unique identification of caller.

DialNo (string)

This parameter value specifies the number to be dialed.

ToPeerType (integer)

This parameter value specifies the type of To-Peer however -1 value can be used to accept only incoming call without generating outgoing call (supports IVR system to accept call and play some wave data).

3001 = User PeerType  
3002 = Line PeerType  
3003 = DirectProxy PeerType

**ToPeerName (string)**

This parameter specifies the name of To-Peer however empty string (" ") can be used to accept only incoming call without generating outgoing call (supports IVR system to accept call and play some wave data).

**Timeout (integer)**

This parameter specifies the time interval (in seconds) for which VaxTele waits for Call-Session to be connected/established, if Call-Session is not established/connected within specified time interval then timeout occurs as a result OnCallSessionTimeout() event triggers.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, CallerName, CallerId, DialNo, 3001,
                      DialNo, 20)
}

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
}
```

### See Also

RejectCallSession(), CloseCallSession(), OpenCallSession(),  
GetVaxObjectError(), OnCallSessionOpen()

## RejectCallSession()

The RejectCallSession() function rejects an incoming call request for a specific Call-Session.

### Syntax

```
boolean RejectCallSession(  
    SessionId,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

StatusCode (integer)

This parameter specifies SIP response status.

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Unauthorized, Not Found etc).

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
    RejectCallSession(SessionId)  
}
```

### See Also

AcceptCallSession(), CloseCallSession(), OnCallSessionOpen(),  
OpenCallSession()

**CloseCallSession()**

The CloseCallSession() function closes the connection and call(s) in that session get disconnected.

This function is called if VaxTele SDK integrated SIP server requires to disconnect a Call-Session in different scenarios for example during a conversation, if VaxTele integrated SIP server checks that user's balance is not enough then SIP server can call CloseCallSession() method and the call in that session get disconnected.

**Syntax**

```
boolean CloseCallSession(SessionId)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
CloseCallSession(SessionId)
```

**See Also**

RejectCallSession(), AcceptCallSession(), OpenCallSession(),  
OnCallSessionClosed()



## OpenCallSession()

A Call-Session consists of either one or two calls where the calls are identified as From-Call and To-Call.

The OpenCallSession() function allocates a new Call-Session with an outgoing call and adds that outgoing call in new Call-Session as To-Call.

Please see **CALL-SESSION** (Page. 209) for more details.

One of the typical scenario for use of OpenCallSession() function is telemarketing in which SIP server dials a call to a number and when call gets connected then it plays wave file of the message, so this function has multiple purposes.

## Syntax

```
integer OpenCallSession(  
    CallerName,  
    CallerId,  
    DialNo,  
    ToPeerType,  
    ToPeerName,  
    Timeout  
)
```

## Parameters

CallerName (string)

This parameter specifies the caller name.

CallerId (string)

This parameter specifies the caller id.

DialNo (string)

This parameter value specifies the number to be dialed.

ToPeerType (integer)

This parameter value specifies the type of To-Peer.

3001 = User PeerType

3002 = Line PeerType

3003 = DirectProxy PeerType

ToPeerName (string)

The value of this parameter specifies the name of To-Peer.

Timeout (integer)

This parameter specifies the time interval (in seconds) for which VaxTele waits for outgoing call to be connected, if outgoing call does not connect in specified time interval then timeout occurs as a result OnCallSessionTimeout() event triggers.

**Return Value**

If the function succeeds, the return value is new SessionId.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

**Example**

```
OpenCallSession("8025", "8025", "8034", 3001, "PeerNameXYZ", 20)
```

**See Also**

RejectCallSession(), CloseCallSession(), AcceptCallSession()

## AcceptTransferBlind()

The AcceptTransferBlind() function accepts the incoming transfer call request from a specific user. This function supports to implement the feature “unannounced/blind call transfer i.e. transferring the call without notifying the desired party/extension of the impending call”.

The OnCallSessionTransferBlind() event triggers when VaxTele server receives blind transfer request from any user, which in turn calls AcceptTransferBlind () method to process the transfer request accordingly.

Event **OnCallSessionTransferBlind()** triggers and AcceptTransferBlind() performs;

- Allocates a New-Call-Session and OnCallSessionCreated() event triggers.
- Generates an outgoing call using TransferTo, PeerType and PeerName values.
- Adds that outgoing call in New-Call-Session as To-Call.
- Removes Transfree-Call from Transferer-Session and adds to New-Call-Session as From-Call.
- New-Call-Session contains two calls Transferee-Call as From-Call and TransferTo-Call as To-Call.
- Closes Transferer-Session and call on Transferer’s side gets disconnected.

## Syntax

```
integer AcceptTransferBlind(  
    TransfererSessionId,  
    CallerName,  
    CallerId,  
    TransferTo,  
    ToPeerType,  
    ToPeerName,  
    Timeout  
)
```

## Parameters

TransfererSessionId (integer)

This parameter specifies a unique identification of a Transferer-CallSession.

CallerName (string)

This parameter specifies the caller name.

CallerId (string)

This parameter specifies the caller id.

TransferTo (string)

This parameter specifies number to be dialed.

ToPeerType (integer)

This parameter value specifies the type of To-Peer.

3001 = User PeerType  
3002 = Line PeerType  
3003 = DirectProxy PeerType

ToPeerName (string)

The value of this parameter specifies the name of To-Peer.

Timeout (integer)

This parameter specifies the time interval (in seconds) for which VaxTele waits for transfered call to be connected, if transfered call does not connect in specified time interval then timeout occurs as a result OnCallSessionTransferTimeout() event triggers.

### Return Value

If the function succeeds, the return value is new SessionId.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

### Example

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererCallType,  
                           Transferer, Transferee, TransferTo)  
{  
    AcceptTransferBlind(TransfererSessionId, Transferee, Transferee,  
                       TransferTo, 3001, TransferTo)  
}  
  
OnCallSessionTransferBlind(TransfererSessionId, TransfererCallType,  
                           Transferer, Transferee, TransferTo)  
{  
    AcceptTransferBlind(TransfererSessionId, Transferee, Transferee,  
                       "0016148448545", 3002, "LineUSA")  
}
```

### See Also

AcceptTransferConsult(), RejectTransfer(), OnCallSessionTransferBlind(),  
GetVaxObjectError()

## AcceptTransferConsult()

The AcceptTransferConsult() function accepts the incoming transfer call request from a specific user. This function supports to implement the feature “announced/consult call transfer i.e. notifying the desired party/extension of the impending call by putting the caller on hold and dialing the desired party/extension”.

The OnCallSessionTransferConsult() event triggers when server receives consult transfer request from a user, which in turn call AcceptTransferConsult () method to process the request accordingly.

Event **OnCallSessionTransferConsult()** triggers and AcceptTransferConsult() performs;

- Allocates a New-Call-Session and OnCallSessionCreated() event triggers.
- Removes Transfree-Call from Transferer-Session and adds to New-Call-Session as From-Call.
- Removes TransferTo-Call from TransferTo-Session and adds to New-Call-Session as To-Call.
- New-Call-Session contains two calls Transferee-Call as From-Call and TransferTo-Call as To-Call.
- Closes both Transferer-Session and TransferTo-Session.
- Both calls on Transferrer side disconnects.

## Syntax

```
integer AcceptTransferConsult(  
                                TransfererSessionId,  
                                TransferToSessionId  
                                )
```

## Parameters

TransfererSessionId (integer)

This parameter specifies a unique identification of a Transferer-CallSession.

TransferToSessionId (integer)

This parameter specifies a unique identification of a TransferTo-CallSession.

## Return Value

If the function succeeds, the return value is new SessionId.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

**Example**

```
OnCallSessionTransferConsult (TransfererSessionId, TransfereeSessionId,  
                             TransferToSessionId, TransfererCallType,  
                             TransfereeCallType, TransferToCallType,  
                             Transferer, Transferee, TransferTo)  
{  
    AcceptTransferConsult(TransfererSessionId, TransferToSessionId)  
}
```

**See Also**

AcceptTransferBlind(), RejectTransfer(), OnCallSessionTransferConsult(),  
GetVaxObjectError()

**RejectTransfer()**

The RejectTransfer() function rejects the incoming transfer call request.

**Syntax**

```
RejectTransfer (SessionId)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

**Return Value**

No return value.

**Example**

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererCallType,  
                           Transferer, Transferee, TransferTo)  
{  
    RejectTransfer(TransfererSessionId)  
}
```

**See Also**

AcceptTransferConsult(), AcceptTransferBlind(), OnCallSessionTransferBlind(),  
OnCallSessionTransferConsult()

## LoadWaveFile()

The LoadWaveFile() function loads wave (.wav) data to the memory and returns the unique play wave identification (WaveId).

That WaveId can be used with other play wave functions to play that loaded wave data to a call-session.

- PlayWaveStartToCallSession(SessionId, WaveId, Loop)
- PlayWaveStopToCallSession(SessionId)
- UnLoadWaveFile(WaveId)

To avoid media stream type and format conversation and save CPU cycles LoadWaveFile() only works with uncompressed wave file (.wav) recorded at format (8000Hz, 16bit, Mono), other media file types (MP3, gsm etc) are not supported. But if it is required then third-party libraries can be used to convert .mp3 to .wav and pass the .wav file to LoadWaveFile() method.

## Syntax

```
integer LoadWaveFile(FileName)
```

## Parameters

FileName (string)

The value of this parameter specifies the file name.

## Return Value

On successful execution this function returns WaveId for loaded wavefile otherwise it returns -1 value and specific error code can be retrieved by calling GetVaxObjectError() method.

## Example

```
MusicWaveId = LoadWaveFile("C:\Music.wav")
if(MusicWaveId = -1) GetVaxObjectError()

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    PlayWaveStartToCallSession(SessionId, MusicWaveId, 1)
}
```

## See Also

LoadWavePCM(), UnLoadWaveFile(), PlayWaveStartToCallSession() ,  
OpenCallSession(), PlayWaveStopToCallSession(), GetVaxObjectError()



## LoadWavePCM()

The LoadWavePCM() function allows to pass voice PCM data to VaxTele and returns the unique play wave identification (WaveId).

That WaveId can be used with other play wave functions to play that loaded wave data to a call-session.

- PlayWaveStartToCallSession(SessionId, WaveId, Loop)
- PlayWaveStopToCallSession(SessionId)
- UnLoadWaveFile(WaveId)

To avoid media stream type and format conversation and save CPU cycles LoadWavePCM() only works with uncompressed PCM format (8000Hz, 16bit, Mono)

LoadWavePCM() can be used to integrate third-party TTS (Text-To-Speech) engines or libraries. TTS engines accept text data and generates voice data, get voice data of format 8000Hz, 16bit, Mono from third-party TTS engine and pass it to VaxTele by using LoadWavePCM() method. For more details about TTS engine, please contact to the vendor of TTS engine.

LoadWavePCM() can also be used to play mp3 and other audio files by using third-party libraries. Use third-party library and convert mp3 or other audio file's compressed voice data to PCM data of format 8000Hz, 16bit, Mono and pass it to VaxTele by using LoadWavePCM() method. Please contact the third-party vendor for further details.

## Syntax

```
integer LoadWavePCM(DataPCM, SizePCM)
```

## Parameters

DataPCM (array)

The value of this parameter specifies the voice PCM data.

SizePCM (integer)

The value of this parameter specifies the size of voice PCM data.

## Return Value

On successful execution this function returns WaveId for loaded PCM data otherwise it returns -1 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    Third-Party-Engine-TTS("time is ten fourty", VoicePCM[], SizePCM)

    WaveId = LoadWavePCM(VoicePCM[], SizePCM)

    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    PlayWaveStartToCallSession(SessionId, WaveId, 0)
}

OnCallSessionPlayWaveDone(SessionId, WaveId)
{
    UnLoadWaveFile(WaveId)
}
```

**See Also**

LoadWaveFile(), UnLoadWaveFile(), PlayWaveStartToCallSession() ,  
OpenCallSession(), PlayWaveStopToCallSession(), GetVaxObjectError()

## UnLoadWaveFile()

The UnLoadWaveFile() function unloads Wave-Id voice data.

### Syntax

```
UnLoadWaveFile(WaveId)
```

### Parameters

WaveId (integer)

This parameter value specifies the unique identification of voice data to be played.

### Return Value

No return value.

### Example

```
MusicWaveId = LoadWaveFile("C:\Music.wav")
if(MusicWaveId = -1) GetVaxObjectError()

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    PlayWaveStartToCallSession(SessionId, MusicWaveId, 1)
}

OnCallSessionClosed(SessionId, CallType, ReasonCode)
{
    PlayWaveStopToCallSession(SessionId)
}
```

### See Also

LoadWavePCM(), LoadWaveFile(), PlayWaveStartToCallSession(),  
OpenCallSession(), PlayWaveStopToCallSession()

## **PlayWaveStartToCallSession()**

The PlayWaveStartToCallSession() function starts playing the wave file (.wav) data to the specific Call-Session.

Buffered based compression technology is introduced in VaxTele SDK to save CPU cycles and highly minimize the processing load on CPU, while playing the wave file data, VaxTele encodes wave-data to voice-codec just one time and cached it for later playing it repeatedly without putting encoding loading again and again on the CPU.

### **Syntax**

```
boolean PlayWaveStartToCallSession(  
                                     SessionId,  
                                     WaveId,  
                                     Loop  
                                   )
```

### **Parameters**

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

Loop (boolean)

This parameter value can be 0 or 1. Assign value 1 to play sound repeatedly otherwise zero.

### **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
HoldWaveId = LoadWaveFile("C:\HoldMusic.wav")
if (HoldWaveId = -1) GetVaxObjectError()

GreetingWaveId = LoadWaveFile("C:\Greeting.wav")
if (GreetingWaveId = -1) GetVaxObjectError()

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
    FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    PlayWaveStartToCallSession(SessionId, GreetingWaveId, 1)

    DetectDigitDTMF(SessionId, 201, 0, 1, 2000) // Enable RFC-2833
    DetectDigitDTMF(SessionId, 201, 2, 1, 2000) // Enable SIP INFO
}

OnDetectedDigitDTMF(SessionId, CallType, DigitDTMF)
{
    if(DigitDTMF = "#22#")
    {
        PlayWaveStopToCallSession(SessionId)
        PlayWaveStartToCallSession(SessionId, HoldWaveId, 1)
    }
}
```

**See Also**

LoadWaveFile(), UnLoadWaveFile(), OpenCallSession(), GetVaxObjectError(),  
PlayWaveStopToCallSession(), DetectDigitDTMF()

## PlayWaveStopToCallSession()

The PlayWaveStopToCallSession() stops playing wave data to a particular Call-Session.

### Syntax

```
boolean PlayWaveStopToCallSession(SessionId)
```

### Parameters

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
GreetingWaveId = LoadWaveFile("C:\Greeting.wav")
if (GreetingWaveId = -1) GetVaxObjectError()

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    PlayWaveStartToCallSession(SessionId, GreetingWaveId, 1)

    DetectDigitDTMF(SessionId, 201, 0, 1, 2000) // Enable RFC-2833
    DetectDigitDTMF(SessionId, 201, 2, 1, 2000) // Enable SIP INFO
}

OnDetectedDigitDTMF(SessionId, CallType, DigitDTMF)
{
    if(DigitDTMF = "#22#")
    {
        PlayWaveStopToCallSession(SessionId)
    }
}
```

### See Also

LoadWaveFile(), UnLoadWaveFile(), OpenCallSession(), GetVaxObjectError(), PlayWaveStopToCallSession(), DetectDigitDTMF()

## RecordWaveStartToCallSession()

The RecordWaveStartToCallSession() function starts recording the voice stream of a particular Call-Session to a wave file (.wav).

VaxTele use (8000Hz, 16bit, mono) format and creates uncompressed wave file (.wav) to save CPU cycles.

### Syntax

```
boolean RecordWaveStartToCallSession(  
                                     SessionId  
                                     FileName,  
                                     )
```

### Parameters

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

FileName (string)

The value of this parameter specifies the file name.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionConnected(SessionId)  
{  
    RecordWaveStartToCallSession(SessionId, "Record.wav")  
}  
  
OnCallSessionClosed(SessionId, CallType, ReasonCode)  
{  
    RecordWaveStopToCallSession(SessionId)  
}
```

### See Also

RecordWaveStopToCallSession(), RecordWavePauseToCallSession(),  
OpenCallSession(), GetVaxObjectError()

## **RecordWaveStopToCallSession()**

The RecordWaveStopToCallSession() function stops the recording of voice stream on a Call-Session.

### **Syntax**

```
boolean RecordWaveStopToCallSession(SessionId)
```

### **Parameters**

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

### **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### **Example**

```
OnCallSessionConnected(SessionId)
{
    RecordWaveStartToCallSession(SessionId, "Record.wav")
}

OnCallSessionClosed(SessionId, CallType, ReasonCode)
{
    RecordWaveStopToCallSession(SessionId)
}
```

### **See Also**

RecordWaveStartToCallSession(), RecordWavePauseToCallSession(),  
OpenCallSession(), GetVaxObjectError()



## RecordWavePauseToCallSession()

The RecordWavePauseToCallSession() function pauses the recording to wave file.

### Syntax

```
boolean RecordWavePauseToCallSession(  
                                     SessionId  
                                     Enable  
                                     )
```

### Parameters

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

Enable (Boolean)

This parameter value can be 0 or 1. Assign value 1 to pause the recording process or 0 to un-pause the recording process.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionConnected(SessionId)  
{  
    RecordWaveStartToCallSession(SessionId , "Record.wav")  
  
    DetectDigitDTMF(SessionId, 201, 0, 1, 2000)  
    DetectDigitDTMF(SessionId, 201, 1, 1, 2000)  
}  
  
OnDetectedDigitDTMF(SessionId, Digit)  
{  
    If(Digit = "#220")  
        RecordWavePauseToCallSession(SessionId, 1)  
    Else  
        RecordWavePauseToCallSession(SessionId, 0)  
}
```

### See Also

RecordWaveStartToCallSession(), RecordWaveStopToCallSession(),  
OpenCallSession(), GetVaxObjectError

**LoadMusicHold()**

The LoadMusicHold() function loads the wave file for hold music.

**Syntax**

```
boolean LoadMusicHold(FileName)
```

**Parameters**

FileName (string)

This parameter value specifies the file name.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
LoadMusicHold("C:\HoldMusic.wav")
```

**See Also**

UnLoadMusicHold(), GetVaxObjectError()

**UnLoadMusicHold()**

The UnLoadMusicHold() function unloads the wave file for hold music.

**Syntax**

```
UnLoadMusicHold()
```

**Parameters**

No parameters.

**Return Value**

No return value.

**Example**

```
UnLoadMusicHold()
```

**See Also**

LoadMusicHold(), GetVaxObjectError()

## AcceptOnHoldRequest()

The AcceptOnHoldRequest() function accepts the on-hold request for particular Call-Session.

VaxTele, when receives hold request then it triggers OnCallSessionOnHold() event in which call to AcceptOnHoldRequest() accepts on-hold request.

### Syntax

```
boolean AcceptOnHoldRequest(SessionId)
```

### Parameters

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionOnHold(SessionId, CallType)
{
    AcceptOnHoldRequest(SessionId)
}
```

### See Also

AcceptOffHoldRequest(), OnCallSessionOnHold(), GetVaxObjectError()

## AcceptOffHoldRequest()

The AcceptOffHoldRequest() function accepts off-hold request for particular Call-Session.

VaxTele receives request for off-hold then it triggers OnCallSessionOffHold() event which in return call AcceptOffHoldRequest() to accept off-hold request.

### Syntax

```
boolean AcceptOffHoldRequest(SessionId)
```

### Parameters

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionOffHold(SessionId, CallType)
{
    AcceptOffHoldRequest(SessionId)
}
```

### See Also

AcceptOnHoldRequest(), OnCallSessionOffHold(), GetVaxObjectError()

## AcceptChatMessage()

The AcceptChatMessage() function accepts the chat message to be transferred to desired party. When VaxTele SDK receives chat message from its client then it triggers OnChatMessageText() event, call AcceptChatMessage() to accept chat message to forward it to desired destination or call RejectChatMessage() to reject the chat message.

### Syntax

```
boolean AcceptChatMessage(  
    ChatMsgId,  
    ToPeerType,  
    ToPeerName  
)
```

### Parameters

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

ToPeerType (integer)

This parameter value specifies the type of To-Peer.

3001 = User PeerType

3002 = Line PeerType

3003 = DirectProxy PeerType

ToPeerName (string)

The value of this parameter specifies the name of To-Peer.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnChatMessageText(ChatMsgId, MsgFrom, MsgTo, MsgText, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
    AcceptChatMessage(ChatMsgId, 3001, "UserABC")  
}
```

### See Also

RejectChatMesage(), GetVaxObjectError()

## RejectChatMessage()

The RejectChatMessage() function rejects the chat message to be transferred to other party.

VaxTele triggers OnChatMessageText() event when it receives chat message request, call to RejectChatMessage() method rejects the incoming chat message request by sending specific SIP error response.

### Syntax

```
RejectChatMessage(  
    ChatMsgId,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

StatusCode (integer)

This parameter specifies SIP response status.

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Unauthorized, Not Found etc).

### Return Value

No return value.

### Example

```
OnChatMessageText(ChatMsgId, MsgFrom, MsgTo, MsgText, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
    RejectChatMessage(ChatMsgId, 404, "Not found")  
}
```

### See Also

AcceptChatMessage(), GetVaxObjectError()

## SendChatMessageText()

The SendChatMessageText() function sends text chat message to a peer (user, line or direct proxy).

### Syntax

```
integer SendChatMessageText(  
                                MsgFrom,  
                                MsgTo,  
                                MsgText,  
                                ToPeerType,  
                                ToPeerName  
                                )
```

### Parameters

MsgFrom (string)

This parameter specifies from name/id.

MsgTo (string)

This parameter specifies to name/id.

MsgText (string)

This parameter specifies the text message.

ToPeerType (integer)

This parameter value specifies the type of To-Peer.

3001 = User PeerType

3002 = Line PeerType

3003 = DirectProxy PeerType

ToPeerName (string)

The value of this parameter specifies the name of To-Peer.

### Return Value

On successful execution this function returns Message-Id value otherwise it returns -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
MsgId = SendChatMessageText ("8025", "8034", "Hello", 3001, "8034")
```

### See Also

AcceptChatMesage(), GetVaxObjectError()



## AcceptChatStatusSubscribe()

The AcceptChatStatusSubscribe() accepts the chat status subscribe request.

VaxTele server when receives chat status subscribe request from its client it triggers OnChatStatusSubscribe () event which in return call AcceptChatStatusSubscribe() to accept the subscription request.

### Syntax

```
boolean AcceptChatStatusSubscribe(  
                                SubscribId,  
                                MsgFrom,  
                                MsgTo,  
                                ToPeerType,  
                                ToPeerName  
                                )
```

### Parameters

SubscribId (integer)

This parameter value specifies a unique identification of status subscription.

MsgFrom (string)

This parameter specifies the From-user.

MsgTo (string)

The value of this parameter specifies the To-user.

ToPeerType (integer)

This parameter value specifies the type of To-Peer.

3001 = User PeerType

3002 = Line PeerType

3003 = DirectProxy PeerType

ToPeerName (string)

The value of this parameter specifies the name of To-Peer.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OnChatstatusSubscribe(SubscribId, MsgFrom, MsgTo, FromPeerType,  
                      FromPeerName, FromIP, FromPort)  
  
{  
    AcceptChatStatusSubscribe( 8, "8021", "8095", 302, "8095")  
}
```

**See Also**

AcceptChatStatusSubscribe(), OnChatStatusSubscribe(), GetVaxObjectError()

## RejectChatStatusSubscribe()

The RejectChatStatusSubscribe() rejects the chat status subscribe request.

When VaxTele receives chat status subscribe request from its client it triggers OnChatStatusSubscribe() event which in return call RejectChatStatusSubscribe() to reject the subscription request.

### Syntax

```
RejectChatStatusSubscribe(  
    SubscribId,  
    MsgFrom,  
    MsgTo  
)
```

### Parameters

SubscribId (integer)  
This parameter value specifies a unique identification of status subscription.

MsgFrom (string)  
This parameter specifies the *From user*.

MsgTo (string)  
The value of this parameter specifies the *To user*.

### Return Value

No return value.

### Example

```
OnChatstatusSubscribe(SubscribeId, MsgFrom, MsgTo, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
    RejectChatStatusSubscribe(3, "8032", "1002")  
}
```

### See Also

AcceptChatStatusSubscribe, OnChatStatusSubscribe, GetVaxObjectError()

**OpenConferenceRoom()**

The OpenConferenceRoom() function creates a conference room to add multiple users. The added users are allowed to speak/listen in a conference.

**Syntax**

```
boolean OpenConferenceRoom(RoomName)
```

**Parameters**

RoomName (string)

This parameter specifies the name of a conference room.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OpenConferenceRoom("RoomA")
```

**See Also**

CloseConferenceRoom(), GetVaxObjectError()

**CloseConferenceRoom()**

The CloseConferenceRoom() function closes the conference room.

**Syntax**

```
CloseConferenceRoom(RoomName)
```

**Parameters**

RoomName (string)

This parameter specifies the name of a conference room.

**Return Value**

No return value.

**Example**

```
CloseConferenceRoom()
```

**See Also**

OpenConferenceRoom(), GetVaxObjectError()

## **AddCallSessionToConferenceRoom()**

The AddCallSessionToConferenceRoom() method adds a Call-Session to conference room thus allowing multiple users to listen/speak in conference.

To add a Call-Session in conference room its mandatory to specify the type of participant. There are three types of participants

### STEALTH SPEAKER

The stealth speaker can hear all participants of conference room however only stealth listener can hear stealth speaker voice.

### STEALTH LISTENER

The stealth listener can hear all participants of conference room.

### NORMAL LISTENER

The normal listener can hear/speak to stealth listener and normal listener.

## **Syntax**

```
boolean AddCallSessionToConferenceRoom(  
                                     RoomName,  
                                     SessionId,  
                                     Type  
                                     )
```

## **Parameters**

RoomName (string)

This parameter specifies the name of conference room.

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

Type (integer)

This parameter value specifies the type of conference room participant.

102 = Stealth Speaker

101 = Stealth Listener

100 = Normal Listener

## **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
AddCallSessionToConferenceRoom("Room1", 102)
```

**See Also**

`RemoveCallSessionFromConferenceRoom()`, `GetVaxObjectError()`

**RemoveCallSessionFromConferenceRoom()**

The RemoveCallSessionFromConferenceRoom() method removes a specified Call-Session from conference room.

**Syntax**

```
boolean RemoveCallSessionFromConferenceRoom(SessionId)
```

**Parameters**

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
RemoveCallSessionFromConferenceRoom()
```

**See Also**

AddCallSessionToConferenceRoom(), GetVaxObjectError()



## **PlayWaveStartToConferenceRoom()**

The PlayWaveStartToConferenceRoom() function starts playing the wave file data to a specific Conference room created by OpenConferenceRoom() exported function.

### **Syntax**

```
boolean PlayWaveStartToConferenceRoom(  
                                     RoomName,  
                                     WaveId,  
                                     Loop  
                                     )
```

### **Parameters**

RoomName (string)

This parameter specifies the name of conference room.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

Loop (boolean)

This parameter value can be 0 or 1. Assign value 1 to play sound repeatedly otherwise zero.

### **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### **Example**

```
PlayWaveStartToConferenceRoom("xyz", 6, 1)
```

### **See Also**

PlayWaveStopToConferenceRoom(), GetVaxObjectError()

**PlayWaveStopToConferenceRoom()**

The PlayWaveStopToConferenceRoom() stops playing wave data to a particular conference room.

**Syntax**

```
boolean PlayWaveStopToConferenceRoom(RoomName)
```

**Parameters**

RoomName (string)

This parameter specifies the name of conference room.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
PayWaveStopToConferenceRoom("Room3")
```

**See Also**

PlayWaveStartToConferenceRoom(), GetVaxObjectError()

## **RecordWaveStartToConferenceRoom()**

The RecordWaveStartToConferenceRoom() function starts recording the voice stream of a particular conference room to a wave file.

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

### **Syntax**

```
boolean RecordWaveStartToConferenceRoom(  
                                           RoomName,  
                                           FileName  
                                           )
```

### **Parameters**

RoomName (string)  
This parameter specifies the name of conference room.

FileName (string)  
The value of this parameter specifies the file name.

### **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### **Example**

```
RecordWaveStartToConferenceRoom("Room5", "voice.wav")
```

### **See Also**

RecordWaveStopToConferenceRoom(), RecordWavePauseToConferenceRoom,  
GetVaxObjectError()

**RecordWaveStopToConferenceRoom()**

The RecordWaveStopToConferenceRoom() function stops recording the voice stream of specific conference room.

**Syntax**

```
boolean RecordWaveStopToConferenceRoom(RoomName)
```

**Parameters**

RoomName (string)

This parameter specifies the name of conference room.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
RecordWaveStopToConferenceRoom("Room4")
```

**See Also**

RecordWaveStartToConferenceRoom(), RecordWavePauseToConferenceRoom, GetVaxObjectError()

## **RecordWavePauseToConferenceRoom()**

The RecordWavePauseToConferenceRoom() function pause the recording of provided conference room.

### **Syntax**

```
boolean RecordWavePauseToConferenceRoom(  
                                           RoomName,  
                                           Enable  
                                           )
```

### **Parameters**

RoomName (string)

This parameter specifies the name of conference room.

Enable (boolean)

The value of this parameter can be 0 or 1. Set the value of this parameter to 1 to pause the recording process or zero to un-pause the recording process .

### **Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### **Example**

```
RecordWavePauseToConferenceRoom( "Room3", 1)
```

### **See Also**

RecordWaveStartToConferenceRoom(), RecordWaveStopToConferenceRoom, GetVaxObjectError()

## VoiceSessionLost()

The VoiceSessionLost() method enables the detection of voice data. VaxTele waits for define interval of time for voice data, if it does not receive data within that interval then it triggers OnSessionLost() event.

### Syntax

```
boolean VoiceSessionLost(  
    Enable,  
    Timeout  
)
```

### Parameters

Enable (boolean)

The value of this parameter can be 0 or 1. Assign value 1 to this parameter to enable voice session lost detection or 0 to disable it.

Timeout (integer)

This parameter value specifies the time interval (in second ) for detection of voice data.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
VoiceSessionLost(1, 20)
```

### See Also

OnCallSessionLost(), GetVaxObjectError()

## SendInfoVM()

The SendInfoVM() function send Voice mail related information to SIP based softphones/hardphones.

VaxTele triggers voice mail related events i.e. OnSendTimeoutVM(), OnSendSuccessVM() upon execution of this function.

### Syntax

```
integer SendInfoVM(  
    UserLogin,  
    NewMsgCount,  
    OldMsgCount,  
    MsgWaiting  
)
```

### Parameters

UserLogin (string)

This parameter value specifies the user's login.

NewMsgCount (integer)

This parameter value specifies the count for new messages.

OldMsgCount (integer)

This parameter value specifies the count for old messages.

MsgWaiting (boolean)

The value of this parameter can be 0 or 1. Set the value of this parameter to 1 to inform sip client that new messages are in queue or 0 if there are no new messages.

### Return Value

If the function succeeds, the return value is MessageId.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

### Example

```
SendInfoVM( "8053", 2, 8, 1)
```

### See Also

OnSendTimeoutVM(), OnSendSuccessVM()

## DiagnosticLogSIP()

The DiagnosticLogSIP() method provides the logging and monitoring of SIP messages.

VaxTele triggers OnIncomingDiagnosticLog()/OnOutgoingDiagnosticLog() event when it receives/sends SIP messages.

### Syntax

```
boolean DiagnosticLogSIP(  
    Inbound,  
    Outbound  
)
```

### Parameters

Inbound (boolean)

The value of this parameter can be 0 or 1. To enable diagnostic of inbound voice stream assign value 1 to this parameter or 0 to disable it.

Outbound (boolean)

The value of this parameter can be 0 or 1. To enable diagnostic of outbound voice stream assign value 1 to this parameter or 0 to disable it.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
DignosticLogSIP(1, 0)
```

### See Also

OnOutgoingDiagnosticLog(), OnIncomingDiagnosticLog(), GetVaxObjectError()



## StartVaxTeleTick()

The function StartVaxTeleTick() sets a time interval, certain task is performed on expiry of this set interval.

When StartVaxTeleTick() method is called, it sets a time internally and trigger the tick event OnVaxTeleTick() after that specific time.

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

### Syntax

```
boolean StartVaxTeleTick(  
    TickId,  
    Elapse  
)
```

### Parameters

TickId (integer)

This parameter specifies the unique tick identification. TickId value should be more than 2000.

Elapse (integer)

The value of this parameter specifies the time (milliseconds).

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
/* Triggers OnVaxTeleTick() after every 8 seconds */  
  
StartVaxTeleTick(2001, 8000)
```

### See Also

StopVaxTeleTick(), OnVaxTeleTick(), GetVaxObjectError()

**StopVaxTeleTick()**

The StopVaxTeleTick() method is used to stop the time counter for specified tick. It works just like KillTimer() win32 API.

**Syntax**

```
boolean StopVaxTeleTick(TickId)
```

**Parameters**

TickId (integer)

This parameter specifies the unique tick identification. TickId value should be more than 2000.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
/* Triggers OnVaxTeleTick() after every 8 seconds */  
  
StartVaxTeleTick(2001, 8000)  
  
StopVaxTeleTick(2001)
```

**See Also**

StartVaxTeleTick(), OnVaxTeleTick(), GetVaxObjectError()

**SetUserAgentName()**

The SetUserAgentName() function sets the user-agent header field of SIP packet.

**Syntax**

```
boolean SetUserAgentName(Name)
```

**Parameters**

Name (string)

This parameter value specifies the User agent name.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
Result = SetUserAgentName("abc")  
if(Result == 0) GetVaxObjectError()
```

**See Also**

GetUserAgentName(), GetVaxObjectError()

**GetUserAgentName()**

The GetUserAgentName() function returns the user-agent value previously set by calling SetUserAgentName() function.

**Syntax**

```
string GetUserAgentName()
```

**Parameters**

No parameters.

**Return Value**

The function returns the user agent name otherwise empty string.

**Example**

```
GetUserAgentName()
```

**See Also**

SetUserAgentName()

**SetSessionNameSDP()**

The SetSessionNameSDP() function sets the session-name field of SIP packet.

**Syntax**

```
boolean SetSessionNameSDP(Name)
```

**Parameters**

Name (string)

This parameter specifies the value that is to be set as session name of SIP packet.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
SetSessionNameSDP("xyz")
```

**See Also**

GetSessionNameSDP(), GetVaxObjectError()

**GetSessionNameSDP()**

The GetSessionNameSDP() function returns the session-name value previously set by SetSessionNameSDP() function.

**Syntax**

```
string GetSessionNameSDP()
```

**Parameters**

No parameters.

**Return Value**

The function returns the session name otherwise empty string.

**Example**

```
GetSessionNameSDP()
```

**See Also**

SetSessionNameSDP()

## SendDTMFCallSession()

The SendDTMFCallSession() function send DTMF digits to the specific Call-Session.

### Syntax

```
boolean SendDTMFCallSession(  
                                SessionId,  
                                TypeDTMF,  
                                Digit  
                                )
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

TypeDTMF (integer)

The value of this parameter specifies mode of DTMF send.

0 = RTP based (RFC 2833)

1 = Inband

2 = SIP based (INFO)

Digit (integer)

Pass the digit to be sent. (0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11).

Where 10 = \* and 11 = #

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
SendDTMFCallSession(SessionId, 0, 3)
```

### See Also

DetectDigitDTMF(), OnDetectedDigitDTMF()

## DetectDigitDTMF()

The DetectDigitDTMF() function enables/disables the detection of DTMF digit at From-Call or To-Call of a Call-Session.

It initiates the digit DTMF detection process and triggers OnDetectedDigitDTMF() event accordingly.

For example

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)
    DetectDigitDTMF(SessionId, 201, 0, 1, 2000)

    // Incoming call is as From-Call in the Call-Session.
}
```

- Enables RTP based detection at From-Call for two seconds time-out.
- VaxTele receives first digit. (e.g. 4)
- VaxTele waits for other digits with time-out interval.
- With two seconds interval, remote person press another digit (e.g. 8)
- VaxTele receives another digit. (e.g. 8)
- Waits for more digits within two seconds time-out interval.
- VaxTele receives no digit within two seconds time-out interval.
- VaxTele triggers **OnDetectedDigitDTMF(SessionId, 201, "48")**

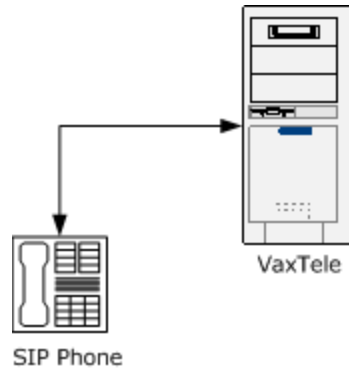
There are total three types of DTMF digit detection standards in IP-Telephony. VaxTele supports all of those three DTMF digit detection standards.

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

Enable either one or all three DTMF detection types, it depends upon the 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.*





### 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

It sends DTMF digits 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.

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 DetectDigitDTMF(  
    SessionId,  
    CallType,  
    TypeDTMF,  
    Enable,  
    MilliSecTimeout  
)
```

### **Parameters**

SessionId (integer)  
This parameter specifies a unique identification of a Call-Session.

CallType (integer)  
This parameter value specifies the call identification.

201 = From-Call  
202 = To-Call

TypeDTMF (integer)

The value of this parameter specifies mode of DTMF detection.

0 = RTP based (RFC 2833)  
1 = Inband  
2 = SIP based (INFO)

Enable (boolean)

This parameter value can be 0 or 1. Assign value 1 to this parameter to enable the detection of DTMF digit or 0 to disable it.

MilliSecTimeout (integer)

The value of this parameter specifies the time interval (in milliseconds) to detect DTMF digit.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling `GetVaxObjectError()` method.

### Example

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, CallerName, CallerId, DialNo, 3001,
                      DialNo, 20)

    DetectDigitDTMF(SessionId, 201, 0, 1, 2000)
}

OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)

    DetectDigitDTMF(SessionId, 201, 0, 1, 2000)
    DetectDigitDTMF(SessionId, 201, 2, 1, 1000)
}
```

### See Also

`OnDetectedDigitDTMF()`, `SendDTMFCallSession()`, `GetVaxObjectError()`

## GetVaxObjectError()

The GetVaxObjectError() method gets the error code for the last operation which is failed to execute.

### Syntax

```
integer GetVaxObjectError()
```

### Parameters

No parameters.

### Return Value

The GetVaxObjectError() returns the error code.

### LIST OF ERROR CODES

200	VaxTele failed to initialize RTP Socket.
201	VaxTele failed to allocate RTP port or provided value of RTP listen IP is incorrect.
202	VaxTele failed to create RTP task manager.
203	Failed to start media thread.
204	Unable to create RTP communication manager.
205	Unable to use RTP communication manager.
206	Unable/failed to open file.
207	Unable to read file.
208	Incorrect wave file format, please use 8000Hz, 16bit, mono uncompressed wave file.
209	Invalid play wave ID.
210	Unable to start recording manager.
211	Selected call is not in the voice session.
212	Failed to initialize VaxTele object.
213	Unable to create SDP body.
214	Unable to create INVITE request.
215	Error to initialize SIP communication layer.
216	VaxTele failed to open SIP listen port or provided value of SIP listen IP is incorrect.
217	Failed to create SIP task manager.
218	Unable to create REGISTER request.
219	Failed to create UN-REGISTER request.
220	Failed to create BYE request.
221	Unable to create CANCEL request.
222	Invalid Call-Id or call does not exist.
223	Invalid session-Id or session does not exist.
224	Provided login does not exist.

225	Login length is incorrect.
226	Provided line does not exist.
227	Unable to send SIP response.
228	Invalid SIP response code, please check SIP RFC 3261.
229	Error to create SIP message.
230	Provided line already exist.
231	Incorrect Reg-Id or Reg-Id does not exist.
232	Error to create SIP response.
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.
243	Invalid to URI.
244	Unable to perform desired operation, call is not connected.
245	Unable to perform desired operation on connected call.
246	Direct communication is not supported
247	Invalid parameter(s) value.
248	Invalid chat message Id.
249	Invalid subscribe Id.
250	Failed to allocate a unique Id.
251	Invalid Id.
252	Provided Id or value already exist.
253	Provided Id or value does not exist.
254	Failed to create Call-Session.
255	Failed to enable DTMF detection.

### Example

```
SetLicenseKey("LicenseKey")

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

### See Also

Initialize(), SetLicenseKey()

## DialToneToCallSession()

The DialToneToCallSession() method enables/disables the dial tone to a particular Call-Session.

The DialToneToCallSession() in return triggers dial tone related events to perform any required task before/after starting/stopping dial tone.

### Syntax

```
boolean DialToneToCallSession(  
                                Enable,  
                                SessionId,  
                                WaveId  
                                )
```

### Parameters

Enable (boolean)

This parameter value can be 0 or 1. Assign value 1 to this parameter to enable dial tone to Call-Session or 0 to disable it.

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played however it can also be assigned value -1 which results in enabling of dial tone but no wave file will be played.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
DialToneToCallSession(1, 4, 2)
```

### See Also

OnCallSessionDialToneStart(), OnCallSessionDialToneStop()

**CallSessionSwapCalls()**

The CallSessionSwapCalls() function swaps the calls in a Call-Session i.e. it swaps the From-Call with To-Call of the provided Call-Session and vice versa.

**Syntax**

```
boolean CallSessionSwapCalls(SessionId)
```

**Parameters**

SessionId (integer)

This parameter value specifies a unique identification of a Call-Session.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
SessionA = OpenCallSession("8024", "8024", "8018", 3001, "xyz", 1)  
SwapCallSession(SessionA)
```

**See Also**

OpenCallSession(), MergeCallSession()

## MergeCallSession()

The MergeCallSession() function merges the two Call-Session by taking From-Call from From-CallSession and To-Call from To-CallSession and moves them into a new Call-Session.

The Call-Session is a collection of either one or two calls where the participating calls are identified as From-Call and To-Call.

Please see **CALL-SESSION** (Page. 209) for more details.

One of the typical scenario for use of MergeCallSession() is telemarketing in which SIP server wants to start a call-session between an agent and remote party by first calling to remote party and then joining them later e.g. SIP server dials a call to a number and when call gets connected then it plays wave file that may ask for certain number to press to talk to agent then SIP Server dials another call to respective agent and then adds both calls into a new Call-Session.

Following sequence of methods calls will be followed in the above scenario

- SessionIdA = OpenCallSession(CallerName, CallerId, DailNo, ToPeerType, ToPeerName, Timeout)
- SessionIdB = OpenCallSession(CallerName, CallerId, DailNo, ToPeerType, ToPeerName, Timeout)
- SwapCallSession(SessionIdA)
- MergeCallSession(SessionIdA, SessionIdB)

## Syntax

```
integer MergeCallSession(  
    FromSessionId,  
    ToSessionId  
)
```

## Parameters

FromSessionId (integer)

This parameter specifies a unique identification of a From-CallSession.

ToSessionId

This parameter specifies a unique identification of a To-CallSession.

## Return Value

On successful execution this function returns SessionId of newly created Call-Session otherwise it returns -1 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OnCallSessionTimeout(SessionIdA, 202)
{
    SessionIdB = MergeCallSession(SessionIdA, -1)

    AcceptCallSession(SessionIdB, "80", "80", " ", -1, " ", 30)
    PlayWaveStartToCallSession(SessionIdB, WaveId, True)
}

OnCallSessionFailed(SessionIdA, 202, 486, "Busy Here")
{
    SessionIdB = OpenCallSession("82", "82", "8034", 3001, "8034", 20)
    SessionIdC = MergeCallSession(SessionIdA, SessionIdB)
}
```

**See Also**

OpenCallSession(), SwapCallSession()



## SetUserData()

The SetUserData() function attaches an instance/object of custom class as custom data to a specific user.

For further details, please see sample code after downloading from the website.

### Syntax

```
boolean SetUserData(  
    UserName,  
    CustomData  
)
```

### Parameters

UserName(String)

This parameter specifies the User name.

CustomData(ClassObject)

This parameter is an instance of custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
AddUser("UserName", "Password", "03421")  
  
CCustomUserData objData = new CCustomUserData()  
SetUserData("UserName", objData)  
  
OnRegisterUserSuccess(UserLogin)  
{  
    CCustomUserData objData = GetUserData(UserLogin)  
    objData.SetRegistered(TRUE)  
}
```

### See Also

GetUserData(), GetVaxObjectError()

## **GetUserData()**

The GetUserData() returns the instance/object of custom class attached to a specific user. Please have a look at sample code after downloading from the website.

### **Syntax**

```
ClassObject GetUserData(Username)
```

### **Parameters**

Username(String)

This parameter specifies the User name.

### **Return Value**

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

### **Example**

```
AddUser("UserName", "Password", "03421")

CCustomUserData objData = new CCustomUserData()
SetUserData("UserName", objData)

OnUnRegisterUser(UserLogin)
{
    CCustomUserData objData = GetUserData(UserLogin)
    objData.SetRegistered(FALSE)
}
```

### **See Also**

SetUserData(), GetVaxObjectError()

## SetLineData()

The SetLineData() function allows to attach an instance/object of custom class as custom data to a specific line.

For further details, please see sample code after downloading from the website.

### Syntax

```
boolean SetLineData(  
    LineName,  
    CustomData  
)
```

### Parameters

LineName(String)  
This parameter specifies the Line name.

CustomData(ClassObject)  
This parameter is an instance of custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
AddLine("LineName", "DisplayName", "UserName", "AuthUser", "AuthPwd"  
        "DomainRealm", "ProxySIP", "", "02314")  
  
CCustomLineData objData = new CCustomLineData()  
SetLineData("LineName", objData)  
  
OnLineRegisterSuccess(LineName)  
{  
    CCustomLineData objData = GetLineData(LineName)  
    objData.bRegistered = TRUE  
}
```

### See Also

GetLineData(), AddLine(), GetVaxObjectError()

## GetLineData()

The GetLineData() returns the instance/object of custom class attached to a specific line.

Please have a look at sample code after downloading from the website.

### Syntax

```
ClassObject GetLineData(LineName)
```

### Parameters

LineName(String)

This parameter specifies the Line name.

### Return Value

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

### Example

```
AddLine("LineName", "DisplayName", "UserName", "AuthUser", "AuthPwd"
        "DomainRealm", "ProxySIP", "", "02314")

CCustomLineData objData = new CCustomLineData()
SetLineData("LineName", objData)

OnLineRegisterFailed(LineName)
{
    CCustomLineData objData = GetLineData(LineName)
    objData.bRegistered = FALSE
}
```

### See Also

SetLineData(), GetVaxObjectError()

## SetDirectProxyData()

The SetDirectProxyData() function attaches an instance/object of custom class as custom data to a specific proxy.

### Syntax

```
boolean SetDirectProxyData(  
    ProxyName,  
    CustomData  
)
```

### Parameters

ProxyName(String)  
This parameter specifies the proxy name.

CustomData(ClassObject)  
This parameter is an instance of custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
AddDirectProxySIP(ProxyName, AuthLogin, AuthPwd, DomainRealm  
    ProxyIP, ProxyPort, CodecList)  
  
CCustomProxyData objData = new CCustomProxyData()  
SetDirectProxyData(ProxyName, objData)
```

### See Also

GetDirectProxyData(), GetVaxObjectError()

**GetDirectProxyData()**

The GetDirectProxyData() returns the instance/object of custom class attached to a specific proxy.

**Syntax**

```
ClassObject GetDirectProxyData(ProxyName)
```

**Parameters**

ProxyName(String)

This parameter specifies the Proxy name.

**Return Value**

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

**Example**

```
RemoveDirectProxySIP(ProxyName)
```

```
CCustomProxyData objData = GetDirectProxyData(ProxyName)  
objData = nil
```

**See Also**

SetDirectProxyData(), GetVaxObjectError()

## SetConferenceRoomData()

The SetConferenceRoomData() function hook-ups an instance/object of custom class as custom data to a specific conference room.

### Syntax

```
boolean SetConferenceRoomData(  
                                RoomName,  
                                CustomData  
                                )
```

### Parameters

RoomName(String)

This parameter specifies the room name.

CustomData(ClassObject)

This parameter is an instance of a custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OpenConferenceRoom ("Room")  
  
CCustomConfRoomData objData = new CCustomConfRoomData()  
SetConferenceRoomData("Room", objData)  
  
OnCallSessionConnected(SessionId)  
{  
    AddCallSessionToConferenceRoom("Room", SessionId, 100)  
  
    CCustomConfRoomData objData = GetConferenceRoomData("Room")  
    objData.nRoomSessionCount = objData.nRoomSessionCount + 1  
}
```

### See Also

GetConferenceRoomData(), GetVaxObjectError()

**GetConferenceRoomData()**

The GetConferenceRoomData() returns the instance/object of custom class hooked-up to a specific room.

**Syntax**

```
ClassObject GetConferenceRoomData(RoomName)
```

**Parameters**

RoomName(String)

This parameter specifies the Room name.

**Return Value**

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

**Example**

```
OnCallSessionClosed(SessionId, CallType, ReasonCode)
{
    CCustomConfRoomData objData = GetConferenceRoomData("Room")
    objData.nRoomSessionCount = objData.nRoomSessionCount - 1

    RemoveCallSessionFromConferenceRoom(SessionId)
}
```

**See Also**

SetConferenceRoomData(), GetVaxObjectError()



## SetCallSessionData()

The SetCallSessionData() function allows to attach an instance/object of custom class as custom data to a specific call-session.

For further details, please see sample code after downloading from the website.

### Syntax

```
boolean SetCallSessionData(  
    SessionId,  
    CustomData  
)
```

### Parameters

SessionId(integer)

This parameter specifies a unique identification of a Call-Session.

CustomData(ClassObject)

This parameter is an instance of a custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionCreated(SessionId, ReasonCode)  
{  
    CCustomCallSessionData objData = new CCustomCallSessionData()  
    SetCallSessionData(SessionId, objData)  
}
```

### See Also

GetCallSessionData(), GetVaxObjectError()

## GetCallSessionData()

The GetCallSessionData() returns the instance/object of custom class attached to a specific call-session.

Please have a look at sample code after downloading from the website.

### Syntax

```
ClassObject GetCallSessionData(SessionId)
```

### Parameters

SessionId(integer)

This parameter specifies a unique identification of a Call-Session.

### Return Value

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

### Example

```
OnCallSessionClosed(SessionId, CallType, ReasonCode)
{
    CCustomCallSessionData objData = GetCallSessionData(SessionId)

    delete objCustomCallData
    objCustomCallData = nil
}
```

### See Also

SetCallSessionData(), GetVaxObjectError()

## VideoCOMM()

The VideoCOMM() function enables video communication. When video communication enables, VaxTele SDK starts acting as proxy server for video streaming.

For video stream proxy process:

1. Only two codecs are supported H263 and H263+
2. Video stream must be encoded using either H263 or H263+
3. Atleast one same video codec either H263 or H263+ must be supported by both SIP clients/peers.

If there are two softphones (A & B), A only supports H263 and B only supports H263+ then video does not work because VaxTele only works as video proxy server, it does not perform video codec conversion.

If A supports H263 and B support H263 & H263+ then video conference works. Similarly if A support H263 & H263+ and B support H263+ & H263 then it also makes the video work.

## Syntax

```
boolean VideoCOMM(Enable)
```

## Parameters

Enable (boolean)

This parameter enables or disables the video communication.

## Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

## Example

```
Result = Initialize("sipsdk.com", "9.7.11.17", 5060, "9.7.11.17", -1)
if(Result == 0) GetVaxObjectError()

VideoCOMM(1)
```

## See Also

Initialize(), GetVaxObjectError()

## GetCallSessionTxCodec()

The GetCallSessionTxCodec() returns audio codec that is being used by VaxTele to compress outbound voice stream of a specific call in a call-session.

### Syntax

```
integer GetCallSessionTxCodec(SessionId, CallType)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

### Return Value

If the function succeeds, the return value is audio codec number.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

Audio Codec Numbers:

0 = GSM

1 = iLBC

2 = G711 A-Law

3 = G711 U-Law

4 = G729

### Example

```
OnCallSessionConnected(SessionId)
{
    TxAudioCodec = GetCallSessionTxCodec(SessionId, 201)
}
```

### See Also

GetCallSessionRxCodec(), GetVaxObjectError()

## GetCallSessionRxCodec()

The GetCallSessionRxCodec() specifies audio codec that is being applied by VaxTele to inbound voice stream of a specific call in a call-session.

### Syntax

```
integer GetCallSessionRxCodec(SessionId, CallType)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

### Return Value

If the function succeeds, the return value is audio codec number.

If the function fails, the return value is -1. To get extended error information, call GetVaxObjectError().

Audio Codec Numbers:

0 = GSM

1 = iLBC

2 = G711 A-Law

3 = G711 U-Law

4 = G729

### Example

```
OnCallSessionConnected(SessionId)
{
    RxAudioCodec = GetCallSessionRxCodec(SessionId, 201)
}
```

### See Also

GetCallSessionTxCodec(), GetVaxObjectError()

## CallSessionMuteVoice()

The CallSessionMuteVoice() mutes listen or speak of a specific call in a call-session.

### Syntax

```
boolean CallSessionMuteVoice(SessionId, CallType, Listen, Speak)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

Listen (Boolean)

This parameter specifies to mute the listening.

Speak (Boolean)

This parameter specifies to mute the speaking.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                   FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, "", 20)

    AddCallSessionToConferenceRoom("RoomName", SessionId, 100)
    CallSessionMuteVoice(SessionId, 201, 1, 0)
}
```

### See Also

OnCallSessionOpen(), GetVaxObjectError()

## BusyLampSubscribeAccept()

The BusyLampSubscribeAccept() function accepts incoming BLF subscribe request. VaxTele receives BLF subscribe request and triggers OnBusyLampSubscribe() event, call BusyLampSubscribeAccept() to accept that incoming BLF subscribe request.

### Syntax

```
boolean BusyLampSubscribeAccept (SubScrbId, Authenticate, Expires)
```

### Parameters

SubScrbId (integer)

This parameter specifies a unique identification of a BLF subscribe request.

Authenticate (boolean)

If Value is non-zero, VaxTele completes login authorization process then accepts the BLF request.

If Value is 0, VaxTele skips login authorization process and accepts the BLF request.

Expires (integer)

This parameter specifies expiry time in seconds of a BLF subscription.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnBusyLampSubscribe(SubScrbId, UserName, ToUserName, FromIP,  
                    FromPort)  
{  
    BusyLampSubscribeAccept(SubScrbId, 1, 1800)  
}
```

### See Also

BusyLampSubscribeReject(), OnBusyLampSubscribe(),  
OnBusyLampSubscribeSuccess()

## BusyLampSubscribeReject()

The BusyLampSubscribeReject() function rejects incoming BLF subscribe request. VaxTele receives BLF subscribe request and triggers OnBusyLampSubscribe() event, call BusyLampSubscribeReject() to reject that incoming BLF subscribe request.

### Syntax

```
boolean BusyLampSubscribeReject (SubScrbId, StatusCode, ReasonPhrase)
```

### Parameters

SubScrbId (integer)

This parameter specifies a unique identification of a BLF subscribe request.

StatusCode (integer)

This parameter specifies SIP response status.

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Unauthorized, Not Found etc).

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
OnBusyLampSubscribe(SubScrbId, UserName, ToUserName, FromIP,  
                    FromPort)  
{  
    BusyLampSubscribeReject(SubScrbId, 404, "Not found")  
}
```

### See Also

BusyLampSubscribeAccept(), OnBusyLampSubscribe(),  
OnBusyLampSubscribeSuccess()



## AddCustomHeader()

The AddCustomHeader() function can be used to add custom header fields in the SIP packets of different SIP requests.

Some of the SIP requests; REGISTER, INVITE, ACK, CANCEL, BYE, OPTIONS

### Syntax

```
boolean AddCustomHeader(ReqId, Name, Value)
```

### Parameters

ReqId (integer)

This parameter specifies a unique identification of a SIP request.  
Supported ReqId values are;

0 = INVITE

1 = REFER

Name (string)

This parameter specifies the name of custom header field.

Value (string)

This parameter specifies the value of custom header field.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
Result = Initialize("sipsdk.com", "192.168.0.25", 5060, "192.168.0.25", -1)  
if(Result == 0) GetVaxObjectError()
```

```
AddCustomHeader(0, "Call_Info", "WaitingTime = 0")
```

### See Also

RemoveCustomHeader(), RemoveCustomHeaderAll(),

## RemoveCustomHeader()

The RemoveCustomHeader() function removes the custom header fields added by using AddCustomHeader() function.

### Syntax

```
boolean RemoveCustomHeader(ReqId, Name)
```

### Parameters

ReqId (integer)

This parameter specifies a unique identification of a SIP request.  
Supported ReqId values are;

0 = INVITE

1 = REFER

Name (string)

This parameter specifies the custom header field.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
RemoveCustomHeader(0, "Call_Info")
```

### See Also

AddCustomHeader(), RemoveCustomHeaderAll()

**RemoveCustomHeaderAll()**

The RemoveCustomHeaderAll() function removes all custom header fields added by using AddCustomHeader() function.

**Syntax**

```
boolean RemoveCustomHeaderAll(ReqId)
```

**Parameters**

ReqId (integer)

This parameter specifies a unique identification of a SIP request.  
Supported ReqId values are;

0 = INVITE

1 = REFER

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
RemoveCustomHeaderAll(0)
```

**See Also**

AddCustomHeader(), RemoveCustomHeader()

## CallSessionDetectAMD()

The CallSessionDetectAMD() method enables/disables the detection of answering machine for a particular call in a call-session.

VaxTele completes the detection and triggers OnCallSessionDetectAMD() event.

### Syntax

```
boolean CallSessionDetectAMD(  
                                SessionId,  
                                CallType,  
                                Enable,  
                                AnalysisTime,  
                                SilenceTime,  
                                SilenceCount  
                                )
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

Enable (Boolean)

This parameter value can be 0 or 1. Assign value 1 to enable the answering machine detection for a particular call and 0 to disable it.

AnalysisTime (integer)

Analysis Time is the specific time period (in millisecond) in which VaxTele detects the answering machine.

SilenceTime (integer)

This parameter value specifies the time interval (in millisecond) for silence i.e. no human voice listens in particular time interval.

SilenceCount (integer)

This parameter value specifies the number of silence interval in a specified time period.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", -1, 30);
    CallSessionDetectAMD(SessionId, 202, 1, 6000, 300, 2)
}
```

**See Also**

OnCallSessionDetectAMD(), GetVaxObjectError()

## CallSessionSendStatusResponse()

The CallSessionSendStatusResponse() function can be used to send response (Trying, Ringing etc.) manually to From-Call for a particular call-session.

### Syntax

```
boolean CallSessionSendStatusResponse(  
    SessionId,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)  
This parameter specifies the unique identification for a Call-Session.

StatusCode (integer)  
This parameter specifies SIP response status.  
**List of SIP Responses** (Page. 208)

ReasonPhrase (string)  
This parameter specifies SIP response reason phrase (Unauthorized, Not Found etc.)

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
Integer nSessionId  
  
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
    nSessionId = SessionId  
    CallSessionSendStatusResponse(SessionId, 180, "Ringing")  
    StartVaxTeleTick(3000)  
}  
  
OnVaxTeleTick(TickId)  
{  
    AcceptCallSession(nSessionId, "", "", "", -1, "", 30)  
}
```

**See Also**

StartVaxTeleTick (), AcceptCallSession()

**TrvsUserStart()**

The TrvsUserStart() starts iteration in the list of users added by using AddUser() function.

**Syntax**

```
boolean TrvsUserStart()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
TrvsUserStart()
Loop(GetUserCount())
{
    UserName = TrvsUserNext()
}
```

**See Also**

TrvsUserNext(), GetVaxObjectError()



**TrvsUserNext()**

The TrvsUserNext() iterates and returns user name.

**Syntax**

```
string TrvsUserNext ()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns user-name value otherwise it returns empty string value.

**Example**

```
TrvsUserStart()
Loop()
{
    UserName = TrvsUserNext()
    if(UserName.length = 0) exit loop
}
```

**See Also**

TrvsUserStart(), AddUser()

**TrvsLineStart()**

The TrvsLineStart() starts iteration in the list of lines added by using AddLine() function.

**Syntax**

```
boolean TrvsLineStart()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
TrvsLineStart()
Loop(GetLineCount())
{
    LineName = TrvsLineNext()
}
```

**See Also**

TrvsLineNext(), GetVaxObjectError(), AddLine()

**TrvsLineNext()**

The TrvsLineNext() iterates and returns line name.

**Syntax**

```
string TrvsLineNext ()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns line-name value otherwise it returns empty string value.

**Example**

```
TrvsLineStart()  
  
Loop()  
{  
    LineName = TrvsLineNext()  
    if(LineName.length = 0) exit loop  
}
```

**See Also**

TrvsLineStart(), AddLine()

**TrvsDirectProxyStart()**

The TrvsDirectProxyStart() starts iteration in the list of direct proxies added by using AddDirectProxySIP() function.

**Syntax**

```
boolean TrvsDirectProxyStart()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
TrvsDirectProxyStart()
Loop(GetDirectProxyCount())
{
    ProxyName = TrvsDirectProxyNext()
}
```

**See Also**

TrvsDirectProxyNext(), GetVaxObjectError(), AddDirectProxySIP()

**TrvsDirectProxyNext()**

The TrvsDirectProxyNext() iterates and returns proxy name.

**Syntax**

```
string TrvsDirectProxyNext ()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns proxy-name value otherwise it returns empty string value.

**Example**

```
TrvsDirectProxyStart()
Loop()
{
    ProxyName = TrvsDirectProxyNext()
    if(ProxyName.length = 0) exit loop
}
```

**See Also**

TrvsDirectProxyStart(), AddDirectProxySIP()

**TrvsCallSessionStart()**

The TrvsCallSessionStart() starts iteration in the list of call-sessions.

**Syntax**

```
boolean TrvsCallSessionStart()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
TrvsCallSessionStart()

Loop(GetCallSessionCount())
{
    SessionId = TrvsCallSessionNext()
}
```

**See Also**

TrvsCallSessionNext(), GetVaxObjectError(), OnCallSessionOpen()

**TrvsCallSessionNext()**

The TrvsCallSessionNext() iterates and returns SessionId.

**Syntax**

```
integer TrvsCallSessionNext()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns SessionId value otherwise it returns -1 value.

**Example**

```
TrvsCallSessionStart()

Loop()
{
    SessionId = TrvsCallSessionNext()
    if(SessionId = -1) exit loop
}
```

**See Also**

TrvsDirectProxyStart(), OpenCallSession(), OnCallSessionOpen()

**TrvsConferenceRoomStart()**

The TrvsConferenceRoomStart() starts iteration in the list of conference rooms added by using OpenConferenceRoom() function.

**Syntax**

```
boolean TrvsConferenceRoomStart()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
TrvsConferenceRoomStart()
Loop(GetConferenceRoomCount())
{
    RoomName = TrvsConferenceRoomNext()
}
```

**See Also**

TrvsConferenceRoomNext(), GetVaxObjectError(), OpenConferenceRoom()



**TrvsConferenceRoomNext()**

The TrvsConferenceRoomNext() iterates and returns conference room name.

**Syntax**

```
string TrvsConferenceRoomNext()
```

**Parameters**

No parameters.

**Return Value**

On successful execution this function returns conference room name otherwise it returns empty string value.

**Example**

```
TrvsConferenceRoomStart()
Loop()
{
    RoomName = TrvsConferenceRoomNext()
    if(RoomName.length = 0) exit loop
}
```

**See Also**

TrvsConferenceRoomStart(), OpenConferenceRoom()

**GetUserCount()**

The GetUserCount() returns total number of users added by using AddUser() function.

**Syntax**

```
integer GetUserCount()
```

**Parameters**

No parameters.

**Return Value**

The function returns total users count on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
AddUser("UserA", "PwdA", "0213")
AddUser("UserB", "PwdB", "3124")

TrvsUserStart()

Loop(GetUserCount())
{
    UserName = TrvsUserNext()
}
```

**See Also**

AddUser(), GetVaxObjectError(), TrvsUserStart(), TrvsUserNext()

**GetLineCount()**

The GetLineCount() returns total number of lines added by using AddLine() function.

**Syntax**

```
integer GetLineCount()
```

**Parameters**

No parameters.

**Return Value**

The function returns total lines count on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
AddLine("LineA", "8034", "8034", "8034", "9341", "sipsdk.com",  
        "192.168.0.3", "", "0134")  
  
AddLine("LineB", "7034", "7034", "7034", "0000", "sipsdk.com",  
        "192.168.0.13", "", "20134")  
  
TrvsLineStart()  
  
Loop(GetLineCount())  
{  
    LineName = TrvsLineNext()  
}
```

**See Also**

AddLine(), GetVaxObjectError(), TrvsLineStart(), TrvsLineNext()

## GetDirectProxyCount()

The GetDirectProxyCount() returns total number of direct proxies added by using AddDirectProxySIP() function.

### Syntax

```
integer GetDirectProxyCount()
```

### Parameters

No parameters.

### Return Value

The function returns total direct proxies count on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
AddDirectProxySIP("DirectA", "8003", "8004", "sip.sdk.com",  
                  "192.168.0.20", 5060, "4213")  
  
AddDirextProxySIP("DirectB", "7003", "7004", "sdk.com",  
                  "66.77.88.99", 5060, "4213")  
  
TrvsDirectProxyStart()  
  
Loop(GetDirectProxyCount())  
{  
    ProxyName = TrvsDirectProxyNext()  
}
```

### See Also

AddDirectProxySIP(), GetVaxObjectError(), TrvsDirectProxyStart(),  
TrvsDirectProxyNext()

**GetConferenceRoomCount()**

The GetConferenceRoomCount() returns total number of conference rooms added by calling OpenConferenceRoom() function.

**Syntax**

```
integer GetConferenceRoomCount()
```

**Parameters**

No parameters.

**Return Value**

The function returns total conference rooms count on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OpenConferenceRoom("RoomA")
OpenConferenceRoom("RoomB")

TrvsConferenceRoomStart()

Loop(GetConferenceRoomCount())
{
    RoomName = TrvsConferenceRoomNext()
}
```

**See Also**

OpenConferenceRoom(), GetVaxObjectError(), TrvsConferenceRoomStart(),  
TrvsConferenceRoomNext()

**GetCallSessionCount()**

The GetCallSessionCount() returns total number of call-sessions.

**Syntax**

```
integer GetCallSessionCount()
```

**Parameters**

No parameters.

**Return Value**

The function returns total call-sessions count on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
OpenCallSession("8025", "8025", "8034", 3001, "PeerNameXYZ", 20)
OpenCallSession("1025", "1025", "1034", 3001, "PeerNameABC", 20)

TrvsCallSessionStart()

Loop(GetCallSessionCount())
{
    SessionId = TrvsCallSessionNext()
}
```

**See Also**

OpenCallSession(), GetVaxObjectError(), TrvsCallSessionStart(),  
TrvsCallSessionNext(), OnCallSessionOpen()

**CustomListOpen()**

The CustomListOpen() function implements and open Hash-Table data structure. Hash-Table data structure is the fastest way to insert, remove and search data.

**Syntax**

```
integer CustomListOpen(TableSize)
```

**Parameters**

TableSize (integer)

This parameter specifies the table size of a Hash-Table.

**Return Value**

On successful execution this function returns a unique identification of a Hash-Table otherwise it returns -1 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
Result = Initialize("sipsdk.com", "9.7.11.17", 5060, "9.7.11.17", -1)
if(Result == 0) GetVaxObjectError()

ListId = CustomListOpen(4096)
```

**See Also**

Initialize(), CustomListClose(), GetVaxObjectError()

**CustomListClose()**

The CustomListClose() function removes Hash-Table data structure from the memory and free the memory resources occupied by the Hash-Table.

**Syntax**

```
boolean CustomListClose(ListId)
```

**Parameters**

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
CustomListClose(ListId)  
UnInitialize()
```

**See Also**

UnInitialize(), CustomListOpen(), GetVaxObjectError()



## CustomListAddData()

The CustomListAddData() inserts an instance/object of custom class as custom data in a specific hash-table.

### Syntax

```
boolean CustomListAddData(ListId, Key, CustomData)
```

### Parameters

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

Key (string)

This parameter specifies the Hash-Key to identify custom data.

CustomData (ClassObject)

This parameter is an instance of a custom data class object.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
CDialPlanData objData = new CDialPlanData()  
CustomListAddData(ListId, "DialPlanData", objData)
```

### See Also

UnInitialize(), CustomListOpen(), SetCallSessionData, GetCallSessionData()

**CustomListRemoveData()**

The CustomListRemoveData() removes an instance/object of custom class from a specific hash-table.

**Syntax**

```
boolean CustomListRemoveData(ListId, Key)
```

**Parameters**

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

Key (string)

This parameter specifies the Hash-Key to identify custom data.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
CustomListRemoveData(ListId, "DialPlanData")
```

**See Also**

CustomListAddData(), SetCallSessionData(), GetUserData()

## CustomListFindData()

The CustomListFindData() searches an instance/object of custom class with a specific hash-key from a specific hash-table.

### Syntax

```
ClassObject CustomListFindData(ListId, Key)
```

### Parameters

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

Key (string)

This parameter specifies the Hash-Key to identify custom data.

### Return Value

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

### Example

```
CDialPlanData objData = CustomListFindData(ListId, "DialPlanData")
```

### See Also

CustomListAddData(), GetCallSessionData(), SetUserData()

## CustomListCount()

The CustomListCount() returns total number of instances/objects of custom data class in a specific hash-table.

### Syntax

```
integer CustomListCount(ListId)
```

### Parameters

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

### Return Value

The function returns total count of data objects in a specific Hash-Table on its successful execution otherwise it return -1 and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
ListId = CustomListOpen(4096)

CDialPlanData objDataA = new CDialPlanData()
CustomListAddData(ListId, "DialPlanDataA", objDataA)

CDialPlanData objDataB = new CDialPlanData()
CustomListAddData(ListId, "DialPlanDataB", objDataB)

DataCount = CustomListCount(ListId)
```

### See Also

CustomListAddData(), GetCallSessionData(), SetUserData()

## CustomListReset()

The CustomListReset() removes all the data objects and reset the specific hash-table.

### Syntax

```
boolean CustomListReset(ListId)
```

### Parameters

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

### Return Value

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

### Example

```
ListId = CustomListOpen(4096)

CDialPlanData objDataA = new CDialPlanData()
CustomListAddData(ListId, "DialPlanDataA", objDataA)

CDialPlanData objDataB = new CDialPlanData()
CustomListAddData(ListId, "DialPlanDataB", objDataB)

DataCount = CustomListCount(ListId)
CustomListReset(ListId)
```

### See Also

CustomListAddData(), GetCallSessionData(), SetUserData()

**CustomListTrvsStart()**

The CustomListTrvsStart() starts iteration in a specific Hash-Table.

**Syntax**

```
boolean CustomListTrvsStart(ListId)
```

**Parameters**

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

**Return Value**

On successful execution this function returns non-zero value otherwise it returns 0 value and specific error code can be retrieved by calling GetVaxObjectError() method.

**Example**

```
CustomListTrvsStart(ListId)

Loop(CustomListCount(ListId))
{
    CDialPlanData objData = CustomListTrvsNext(ListId)
}
```

**See Also**

CustomListTrvsNext(), GetCallSessionData(), SetUserData()

**CustomListTrvsNext()**

The CustomListTrvsNext() iterates and returns data object in a specific hash-table.

**Syntax**

```
ClassObject CustomListTrvsNext(ListId)
```

**Parameters**

ListId (integer)

This parameter specifies the unique identification of a Hash-Table.

**Return Value**

If the function succeeds, the return value is instance/object of custom class.

If the function fails, the return value is nil, NULL or 0. To get extended error information, call GetVaxObjectError().

**Example**

```
CustomListTrvsStart(ListId)

Loop(CustomListCount(ListId))
{
    CDialPlanData objData = CustomListTrvsNext(ListId)
}
```

**See Also**

CustomListTrvsStart(), GetCallSessionData(), SetUserData()

## **EXPORTED EVENTS**

### **OnVaxErrorLog()**

VaxTele triggers OnVaxErrorLog() when execution of any function fails.

Please see **LIST OF ERROR CODES** (Page. 83) for more details.

### **Syntax**

```
OnVaxErrorLog(FuncName, ErrorCode, ErrorMsg)
```

### **Parameters**

FuncName (string)

This parameter value specifies name of the function.

ErrorCode (integer)

This parameter value specifies error code.

ErrorMsg (string)

This parameter value specifies error text message.

### **Example**

```
Result = Initialize("sipsdk.com", "9.7.11.17", 5060, "9.7.11.17", -1)

// if Initialize() fails then OnVaxErrorLog() triggers

OnVaxErrorLog(FuncName, ErrorCode, ErrorMsg)
{
}
}
```

### **See Also**

GetVaxObjectError()



## OnLineRegisterTrying()

VaxTele triggers OnLineRegisterTrying() event when it receives SIP response "100, Trying" from other SIP server.

VaxTele connect and work with other external SIP servers and IP-Telephony Service providers by using AddLine() and RegisterLine() functions.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 205) for more details.

### Syntax

```
OnLineRegisterTrying(LineName)
```

### Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

### Example

```
OnLineRegisterTrying(LineName)
{
}
```

### See Also

RegisterLine(), AddLine(), OnLineRegisterFailed(), OnLineRegisterSuccess()

## OnLineRegisterFailed()

VaxTele triggers OnLineRegisterFailed() event when line (SIP account settings) registration to external SIP server or IP-Telephony service provider failed.

### Syntax

```
OnLineRegisterFailed(  
    LineName,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

StatusCode (integer)

This parameter specifies SIP response status code (408, 403 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Request Timeout, Forbidden etc).

### Example

```
OnLineRegisterFailed(LineName, StatusCode, ReasonPhrase)  
{  
}  
}
```

### See Also

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

**OnLineRegisterSuccess()**

VaxTele triggers OnLineRegisterSuccess() event when line (SIP account settings) registration request (by RegisterLine() function) to external SIP server or IP-Telephony service provider is successfully completed.

Please see **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)** (Page. 205) for more details.

**Syntax**

```
OnLineRegisterSuccess(LineName)
```

**Parameters**

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

**Example**

```
OnLineRegisterSuccess(LineName)
{
}
```

**See Also**

AddLine(), RegisterLine(), OnLineRegisterTrying(), OnLineRegisterFailed()

**OnLineUnRegisterTrying()**

VaxTele triggers OnLineUnRegisterTrying() event when it receives SIP response "100, Trying" from other SIP server during unregister process.

To unregister or disconnect VaxTele from external third party SIP Server the UnRegisterLine() is used.

**Syntax**

```
OnLineUnRegisterTrying(LineName)
```

**Parameters**

LineName (integer)

This parameter value specifies the unique line name to identify a specific line.

**Example**

```
OnLineUnRegisterTrying(LineName)
{
}
```

**See Also**

RegisterLine(), UnRegisterLine(), AddLine(), OnLineUnRegisterFailed(), OnLineUnRegisterSuccess()

## OnLineUnRegisterFailed()

VaxTele triggers OnLineUnRegisterFailed() event, when a provided line failed to un-registered from external SIP server or IP-Telephony service provider.

### Syntax

```
OnLineUnRegisterFailed(  
    LineName,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

StatusCode (integer)

This parameter specifies SIP response status code (408, 403 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Request Timeout, Forbidden etc).

### Example

```
OnLineUnRegisterFailed(LineName)  
{  
}
```

### See Also

RegisterLine(), UnRegisterLine(), AddLine(), OnLineUnRegisterTrying(),  
OnLineUnRegisterSuccess()

**OnLineUnRegisterSuccess()**

VaxTele triggers OnLineUnRegisterSuccess() event when line unregistered successfully from external SIP server or IP-Telephony service provider.

VaxTele calls UnRegisterLine() function to un-register/disconnect a line (SIP account settings) from external SIP server or IP-Telephony service provider and if unregister request is successfully executed then OnLineUnRegisterSuccess() event triggers.

**Syntax**

```
OnLineUnRegisterSuccess(LineName)
```

**Parameters**

LineName (string)

This parameter value specifies the unique line name to identify a specific line.

**Example**

```
OnLineUnRegisterSuccess(LineName)
{
}
```

**See Also**

RegisterLine(), UnRegisterLine(), AddLine(), OnLineUnRegisterTrying(),  
OnLineUnRegisterFailed()

**OnUnRegisterUser()**

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

**Syntax**

```
OnUnRegisterUser(UserLogin)
```

**Parameters**

UserLogin (string)

This parameter specifies the user's login of SIP client.

**Example**

```
OnUnRegisterUser(UserLogin)
{
    RemoveUser(UserLogin)
}
```

**See Also**

OnRegisterUser(), RemoveUser(), AddUser()

## OnRegisterUser()

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

Please see **SIP CLIENT REGISTRATION FLOW** (Page. 197) for more details.

### Syntax

```
OnRegisterUser(  
    UserLogin,  
    Domain,  
    FromIP  
    FromPort  
)
```

### Parameters

UserLogin (string)

This parameter specifies the user's login of SIP client.

Domain (string)

This parameter specifies the domain and its value is used to configure and register the SIP clients to VaxTele and other SIP servers.

FromIP (string)

This parameter value specifies the *from* IP address.

FromPort (integer)

This parameter specifies the *from* port number.

### Example

```
OnRegisterUser(UserLogin, Domain, FromIP, FromPort)  
{  
    AddUser()  
}
```

### See Also

OnUnRegisterUser(), AddUser(), RemoveUser()



**OnRegisterUserSuccess()**

The OnRegisterUserSuccess() event triggers when SIP client successfully registers to VaxTele server.

Please see **SIP CLIENT REGISTRATION FLOW** (Page. 197) for more details.

**Syntax**

```
OnRegisterUserSuccess(UserLogin)
```

**Parameters**

UserLogin (string)

This parameter specifies the user's login of SIP client.

**Example**

```
OnRegisterUserSuccess(UserLogin)
{
}
```

**See Also**

OnRegisterUser(), OnRegisterUserFailed()

**OnRegisterUserFailed()**

The OnRegisterUserFailed() event triggers when SIP client fails to register with VaxTele server.

**Syntax**

```
OnRegisterUserFailed(UserLogin)
```

**Parameters**

UserLogin (string)

This parameter specifies the user's login of SIP client.

**Example**

```
OnRegisterUserFailed(UserLogin)
{
    RemoveUser(UserLogin)
}
```

**See Also**

OnRegisterUser(), OnRegisterUserSuccess(), RemoveUser()

## OnChatMessageText()

The OnChatMessageText() event triggers when VaxTele server receives chat message from SIP client.

### Syntax

```
OnChatMessageText(  
    ChatMsgId,  
    MsgFrom,  
    MsgTo,  
    MsgText,  
    FromPeerType,  
    FromPeerName,  
    FromIP,  
    FromPort  
)
```

### Parameters

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

MsgFrom (string)

This parameter specifies the *From user*.

MsgTo (string)

This parameter specifies the *To user*.

MsgText (string)

This parameter value specifies the message text.

FromPeerType (integer)

This parameter value specifies the type of From-Peer.

3001 = User

3002 = Line

3003 = DirectProxy

FromPeerName (string)

This parameter value specifies the name of From-Peer.

FromIP (string)

This parameter value specifies the *from IP* address.

FromPort (integer)

This parameter specifies the *from port* number.

**Example**

```
OnChatMessageText( ChatMsgId, MsgFrom, MsgTo, MsgText, FromPeerType,  
                  FromPeerName, FromIP, FromPort )  
{  
}
```

**See Also**

## OnChatMessageTyping()

The OnChatMessageTyping() event triggers when SIP client starts typing a message.

### Syntax

```
OnChatMessageTyping(  
    ChatMsgId,  
    MsgFrom,  
    MsgTo,  
    IsTypingStart,  
    FromPeerType,  
    FromPeerName,  
    FromIP,  
    FromPort  
)
```

### Parameters

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

MsgFrom (string)

This parameter specifies the *From user*.

MsgTo (string)

This parameter specifies the *To user*.

IsTypingStart (boolean)

This parameter value can be 0 or 1. Assign value 1 if client has started typing a message otherwise 0.

FromPeerType (integer)

This parameter value specifies the type of From-Peer.

3001 = User

3002 = Line

3003 = DirectProxy

FromPeerName (string)

This parameter value specifies the name of From-Peer.

FromIP (string)

This parameter value specifies the *From IP* address.

FromPort (integer)

This parameter specifies the *From port* number.

**Example**

```
OnChatMessageTyping(ChatMsgId, MsgFrom, MsgTo, IsTypingStart,  
                    FromPeerType, FromPeerName, FromIP, FromPort)  
{  
}
```

**See Also**

## OnChatStatusSubscribe()

The OnChatStatusSubscribe() event triggers when VaxTele receives chat status subscribe request from its SIP client.

### Syntax

```
OnChatStatusSubscribe(  
    SubscribId,  
    MsgFrom,  
    MsgTo,  
    FromPeerType,  
    FromPeerName,  
    FromIP,  
    FromPort  
)
```

### Parameters

SubscribId (integer)

This parameter value specifies a unique identification of status subscription.

MsgFrom (string)

This parameter specifies the *From user*.

MsgTo (string)

This parameter specifies the *To user*.

FromPeerType (integer)

This parameter value specifies the type of From-Peer.

3001 = User

3002 = Line

3003 = DirectProxy

FromPeerName (string)

This parameter value specifies the name of From-Peer.

FromIP (string)

This parameter value specifies the *From IP* address.

FromPort (integer)

This parameter specifies the *From port* number.

### Example

```
OnChatStatusSubscribe(SubscribId, MsgFrom, MsgTo, FromPeerType,  
    FromPeerName, FromIP, FromPort)  
{  
}
```

**See Also**

AcceptChatStatusSubscribe(), RejectChatStatusSubscribe()



**OnChatMessageSuccess()**

The OnChatMessageSuccess() event triggers when VaxTele successfully sends the chat message to SIP client or other SIP server.

**Syntax**

```
OnChatMessageSuccess(ChatMsgId)
```

**Parameters**

ChatMsgId (integer)  
This parameter value specifies a unique identification of a particular chat message.

**Example**

```
OnChatMessageSuccess(ChatMsgId)  
{  
}
```

**See Also**

OnChatMessageFailed(), AcceptChatMessage(), RejectChatMessage()

## OnChatMessageFailed()

The OnChatMessageFailed() event triggers when VaxTele failed to send the chat message to SIP client or other SIP server.

### Syntax

```
OnChatMessageFailed(  
    ChatMsgId,  
    StatusId,  
    ReasonPhrase  
)
```

### Parameters

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

StatusCode (integer)

This parameter specifies SIP response status code (408, 403 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Request Timeout, Forbidden etc).

### Example

```
OnChatMessageFailed(ChatMsgId, StatusId, ReasonPhrase)  
{  
}
```

### See Also

OnChatMessageSuccess(), AcceptChatMessage(), RejectChatMessage()

**OnChatMessageTimeout()**

The OnChatMessageTimeout() event triggers when VaxTele failed to receive chat message received response from SIP client or other SIP server within specified time interval.

**Syntax**

```
OnChatMessageTimeout(ChatMsgId)
```

**Parameters**

ChatMsgId (integer)

This parameter value specifies a unique identification of a particular chat message.

**Example**

```
OnChatMessageTimeout(ChatMsgId)
{
}
```

**See Also**

OnChatMessageFailed(), AcceptChatMessage(), RejectChatMessage(),  
OnChatMessageSuccess()

## OnCallSessionOpen()

The OnCallSessionOpen() event triggers when VaxTele gets the notification for an incoming call.

### Syntax

```
OnCallSessionOpen(  
    SessionId,  
    CallerName,  
    CallerId,  
    DialNo,  
    FromPeerType,  
    FromPeerName,  
    FromIP,  
    FromPort  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallerName (string)

This parameter specifies the caller name.

CallerId (string)

This parameter specifies a unique identification of caller.

DialNo (string)

This parameter value specifies the number to be dialed.

FromPeerType (integer)

This parameter value specifies the type of From-Peer.

3001 = User

3002 = Line

3003 = DirectProxy

FromPeerName (string)

This parameter value specifies the name of From-Peer.

FromIP (string)

This parameter value specifies the *From IP* address.

FromPort (integer)

This parameter specifies the *From port* number.

**Example**

```
OnCallSessionOpen(SessionId, CallerName, CallerId, DialNo, FromPeerType,  
                  FromPeerName, FromIP, FromPort)  
{  
  
}
```

**See Also**

AcceptCallSession(), OnCallSessionConnected, OnCallSessionFailed()

## OnCallSessionConnecting()

The OnCallSessionConnecting() event triggers as soon as call connection process begins. VaxTele Server receives SIP status responses from SIP client/third party server during call connection process.

### Syntax

```
OnCallSessionConnecting(  
    SessionId,  
    CallType,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the type of *call*.

201 = From-Call

202 = To-Call

StatusCode (integer)

This parameter specifies SIP response status code (100, 181 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Trying, Ringing etc).

### Example

```
OnCallSessionConnecting(SessionId, CallType, StatusCode, ReasonPhrase)  
{  
}
```

### See Also

AcceptCallSession(), OnCallSessionConnected, OnCallSessionOpen()

## OnCallSessionFailed()

The OnCallSessionFailed() event triggers when VaxTele receives failure responses during call connection process and failed to established a Call-Session with SIP client/third party server.

### Syntax

```
OnCallSessionFailed(  
    SessionId,  
    CallType,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the type of *call*.

201 = From-Call

202 = To-Call

StatusCode (integer)

This parameter specifies SIP response status code (486, 404 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Busy here, Not found etc).

### Example

```
OnCallSessionFailed(SessionId, CallType, StatusCode, ReasonPhrase)  
{  
}
```

### See Also

AcceptCallSession(), OnCallSessionConnected, OnCallSessionOpen()

**OnCallSessionConnected()**

The OnCallSessionConnected() event triggers when VaxTele successfully established a Call-Session with SIP client/third party server.

**Syntax**

```
OnCallSessionConnected(SessionId)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

**Example**

```
OnCallSessionConnected(SessionId)
{
}
```

**See Also**

AcceptCallSession(), OnCallSessionOpen()



## OnCallSessionLost()

The OnCallSessionLost() event triggers when VaxTele does not receive voice data for define interval of time. The event is triggered by VoiceSessionLost() method.

### Syntax

```
OnCallSessionLost(  
    SessionId,  
    CallType  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the type of *call*.

201 = From-Call

202 = To-Call

### Example

```
OnCallSessionLost(SessionId, CallType)  
{  
}  
}
```

### See Also

VoiceSessionLost()

## OnCallSessionHangup()

The OnCallSessionHangup() event triggers when remote party hang up the phone.

### Syntax

```
OnCallSessionHangup(  
    SessionId,  
    CallType  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

### Example

```
OnCallSessionHangup(SessionId, CallType)  
{  
}
```

### See Also

## OnCallSessionTimeout()

The OnCallSessionTimeout() event triggers when VaxTele fails to establish a Call-Session and does not receive the response from SIP client with in time period specified in AcceptCallSession() / Create CallSession() method. This event is triggered by AcceptCallSession() or OpenCallSession() function.

### Syntax

```
OnCallSessionTimeout(  
    SessionId,  
    CallType  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the type of *call*.

201 = From-Call

202 = To-Call

### Example

```
OnCallSessionTimeout(SessionId, CallType)  
{  
}  
}
```

### See Also

AcceptCallSession(), OpenCallSession()

**OnCallSessionCancelled()**

The OnCallSessionCancelled() event triggers when caller cancels the call prior to its acceptance by callee.

**Syntax**

```
OnCallSessionCancelled(SessionId)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session..

**Example**

```
OnCallSessionCancelled(SessionId)
{
}
```

**See Also**

## OnCallSessionOnHold()

The OnCallSessionOnHold() event triggers, when VaxTele receives call on-hold request from SIP client for specific Call-Session.

### Syntax

```
OnCallSessionOnHold(  
    SessionId,  
    CallType  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session..

CallType(integer)

This parameter value specifies the type of *call*.

201 = From-Call

202 = To-Call

### Example

```
OnCallSessionOnHold(SessionId, CallType)  
{  
}  
}
```

### See Also

AcceptOnHoldRequest(), OnCallSessionOffHold()

**OnCallSessionOffHold()**

The OnCallSessionOffHold() event triggers, when VaxTele receives call off-hold request from SIP client for specific Call-Session.

**Syntax**

```
OnCallSessionOffHold(  
    SessionId,  
    CallType  
)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session..

CallType(integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

**Example**

```
OnCallSessionOffHold(SessionId, CallType)  
{  
  
}
```

**See Also**

AcceptOffHoldRequest(), OnCallSessionOnHold()

## OnCallSessionTransferBlind()

The OnCallSessionTransferBlind ()event triggers when VaxTele receives blind call transfer request from SIP client.

### Syntax

```
OnCallSessionTransferBlind(  
    TransfererSessionId,  
    TransfererCallType,  
    Transferer,  
    Transferee,  
    TransferTo  
)
```

### Parameters

TransfererSessionId (integer)  
This parameter specifies the session identification of transferer.

TransfererCallType (integer)  
This parameter value specifies the type of *call*.

201 = From-Call  
202 = To-Call

Transferer (string)  
The parameter value specifies the transferer user name.

Transferee (string)  
The parameter value specifies the transferee user name.

TransferTo(string)  
This parameter specifies *transferer To* user name.

### Example

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererCallType,  
    {  
        Transferer, Transferee, TransferTo  
    })
```

### See Also

AcceptTransferBlind(), AcceptTransferConsult(),OnCallSessionTransferConsult()

## OnCallSessionTransferConsult()

The OnCallSessionTransferConsult() event triggers when VaxTele receives consult call transfer request from SIP client.

### Syntax

```
OnCallSessionTransferConsult(  
    TransfererSessionId,  
    TransfereeSessionId,  
    TransferToSessionId,  
    TransfererCallType,  
    TransfereeCallType,  
    TransferToCallType,  
    Transferer,  
    Transferee,  
    TransferTo  
)
```

### Parameters

TransfererSessionId (integer)

This parameter specifies the session identification of transferer.

TransfereeSessionId (integer)

This parameter specifies the session identification of transferee.

TransferToSessionId (integer)

This parameter specifies the session identification of *transfer To*.

TransfererCallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

TransfereeCallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

TransferToCallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

Transferer (string)

The parameter value specifies the transferer user name.



Transferee (string)

The parameter value specifies the transferee user name.

TransferTo (string)

This parameter specifies Transfer *To* user name.

### Example

```
OnCallSessionTransferConsult (TransfererSessionId, TransfereeSessionId,  
                             TransferToSessionId, TransfererCallType,  
                             TransfereeCallType, TransferToCallType,  
                             Transferer, Transferee, TransferTo)  
{  
}  
}
```

### See Also

AcceptTransferBlind(), AcceptTransferConsult(), OnCallSessionTransferBlind()

## OnCallSessionTransferring()

The OnCallSessionTransferring() event triggers when a call transfer process initiates and VaxTele starts receiving SIP status responses regarding transferring of call.

### Syntax

```
OnCallSessionTransferring(  
    SessionId,  
    TransferType,  
    CallType,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

TransferType (integer)

This parameter value specifies the type of transfer i.e. blind or consult.

401 = Transfer-Blind

402 = Transfer-Consult

CallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

StatusCode (integer)

This parameter specifies SIP response status code (100, 181 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Trying, Ringing etc).

### Example

```
OnCallSessionTransferring(SessionId, TransferType, CallType, StatusCode,  
    ReasonPhrase)  
{  
}
```

**See Also**

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(),  
OnCallSessionTransferred(), TransferReject()

## OnCallSessionTransferFailed()

The OnCallSessionTransferFailed() event triggers when VaxTele receives failure responses during call transfer process and failed to transfer the call.

### Syntax

```
OnCallSessionTransferFailed(  
    SessionId,  
    TransferType,  
    CallType,  
    StatusCode,  
    ReasonPhrase  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

TransferType (integer)

This parameter value specifies the type of transfer i.e. blind or consult.

401 = Transfer-Blind

402 = Transfer-Consult

CallType (integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

StatusCode (integer)

This parameter specifies SIP response status code (486, 404 etc).

**List of SIP Responses** (Page. 208)

ReasonPhrase (string)

This parameter specifies SIP response reason phrase (Busy here, Not found etc).

### Example

```
OnCallSessionTransferFailed(SessionId, TransferType, CallType, StatusCode,  
    ReasonPhrase)  
{  
}
```

**See Also**

OnCallSessionTransferBlind(), OnCallSessionTransferConsult(),  
RejectTransfer()

**OnCallSessionTransferTimeout()**

The OnCallSessionTransferTimeout() event triggers when call transfer process is failed to complete and VaxTele does not receive the response from SIP client ( Transfer-To) with in specified time limit.

**Syntax**

```
OnCallSessionTransferTimeout(  
    SessionId,  
    TransferType,  
    CallType  
)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

TransferType (integer)

This parameter value specifies the type of transfer i.e. blind or consult.

401 = Transfer-Blind

402 = Transfer-Consult

CallType(integer)

This parameter value specifies the type of call.

201 = From-Call

202 = To-Call

**Example**

```
OnCallSessionTransferTimeout(SessionId, TransferType, CallType)  
{  
}
```

**See Also**

OnCallSessionTransferBlind(), OnCallSessionTransferConsult(),  
RejectTransfer()

## OnCallSessionTransferred()

The OnCallSessionTransferred() event triggers when VaxTele successfully transferred a call.

### Syntax

```
OnCallSessionTransferred(  
    SessionId,  
    TransferType  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

TransferType (integer)

This parameter value specifies the type of transfer i.e. blind or consult.

401 = Transfer-Blind

402 = Transfer-Consult

### Example

```
OnCallSessionTransferred(SessionId, TransferType)  
{  
}
```

### See Also

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(),  
OnCallSessionTransferring()

## OnDetectedDigitDTMF()

The OnDetectedDigitDTMF() event triggers when VaxTele notifies about the DTMF digit receives from From-Call or To-Call in a Call-Session.

### Syntax

```
OnDetectedDigitDTMF(  
    SessionId  
    CallType  
    DigitDTMF  
)
```

### Parameters

SessionId (integer)  
This parameter specifies a unique identification of a Call-Session..

CallType (integer)  
This parameter value specifies the call identification.

201 = From-Call  
202 = To-Call

DigitDTMF (string)  
This parameter value specifies any digit that has been pressed.

### Example

```
OnDetectedDigitDTMF(SessionId, CallType, DigitDTMF)  
{  
    if(DigitDTMF = "#") // if user press #  
        PlayWaveStartToCallSession()  
}
```

### See Also

DetectDigitDTMF()



## OnOutgoingDiagnosticLog()

The OnOutgoingDiagnosticLog() event triggers when VaxTele sends a SIP packet. This event can be use for logging and monitoring of outbound SIP messages.

### Syntax

```
OnOutgoingDiagnosticLog(  
    MsgSIP,  
    ToIP,  
    ToPort  
)
```

### Parameters

- MsgSIP (string)  
This parameter value specifies the SIP packet message.
- ToIP (string)  
This parameter value specifies the To IP address.
- ToPort (integer)  
This parameter value specifies the To port number.

### Example

```
OnOutgoingDiagnosticLog(MsgSIP, ToIP, ToPort)  
{  
  
}
```

### See Also

DiagnosticLogSIP(), OnIncomingDiagnosticLog()

## OnIncomingDiagnosticLog()

The OnIncomingDiagnosticLog() event triggers when VaxTele receives a SIP packet. This event can be use for logging and monitoring of inbound SIP messages.

### Syntax

```
OnIncomingDiagnosticLog(  
    MsgSIP,  
    FromIP,  
    FromPort  
)
```

### Parameters

MsgSIP (string)

This parameter value specifies the SIP packet message.

FromIP (string)

This parameter value specifies the *From IP* address.

FromPort (integer)

This parameter specifies the *From port* number.

### Example

```
OnIncomingDiagnosticLog(MsgSIP, FromIP, FromPort)  
{  
  
}
```

### See Also

DiagnosticLogSIP(), OnOutgoingDiagnosticLog()

**OnVaxTeleTick()**

The OnVaxTeleTick() event triggers after a specified time interval set by StartVaxTeleTick() function.

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

**Syntax**

```
OnVaxTeleTick(TickId)
```

**Parameters**

TickId (integer)

This parameter specifies the unique tick identification. TickId value should be more than 2000.

**Example**

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

**See Also**

StartVaxTeleTick(), StopVaxTeleTick()

**OnSendTimeoutVM()**

The OnSendTimeoutVM() event triggers when VaxTele fails to send voice mail related information to SIP based softphones/hardphones in specified time interval.

**Syntax**

```
OnSendTimeoutVM(MsgIdVM)
```

**Parameters**

MsgIdVM (integer)

This parameter specifies a unique identification of voice mail message.

**Example**

```
OnSendTimeoutVM(MsgIdVM)
{
}
}
```

**See Also**

OnSendSucessVM(), SendInfoVM()

**OnSendSucessVM()**

The OnSendSucessVM() event triggers when VaxTele successfully sends voice mail related information to SIP based softphones/hardphones.

**Syntax**

```
OnSendSucessVM(MsgIdVM)
```

**Parameters**

MsgIdVM (integer)

This parameter specifies a unique identification of voice mail message.

**Example**

```
OnSendSucessVM(MsgIdVM)
{
}
}
```

**See Also**

OnSendTimeout(), SendInfoVM()

**OnCallSessionDialToneStart()**

The OnCallSessionDialToneStart() event triggers when VaxTele starts playing the dial tone.

**Syntax**

```
OnCallSessionDialToneStart(  
    SessionId,  
    WaveId  
)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

**Example**

```
OnCallSessionDialToneStart(SessionId, WaveId)  
{  
  
}
```

**See Also**

DialToneToCallSession(), OnCallSessionDialToneStop()

**OnCallSessionDialToneStop()**

The OnCallSessionDialToneStop() event triggers when VaxTele stops playing the dial tone.

**Syntax**

```
OnCallSessionDialToneStop(  
    SessionId,  
    WaveId  
)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

**Example**

```
OnCallSessionDialToneStop(SessionId, WaveId)  
{  
  
}
```

**See Also**

DialToneToCallSession(), OnCallSessionDialToneStart()

**OnCallSessionPlayWaveDone()**

The OnCallSessionPlayWaveDone() event triggers when a wave file played successfully for a particular call-session.

**Syntax**

```
OnCallSessionPlayWaveDone(  
                                SessionId,  
                                WaveId  
                                )
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a call-session.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

**Example**

```
OnCallSessionPlayWaveDone(SessionId, WaveId)  
{  
  
}
```

**See Also**

LoadWaveFile(), LoadWavePCM(), PlayWaveStartToCallSession(),  
PlayWaveStopToCallSession()



**OnConferenceRoomPlayWaveDone()**

The OnConferenceRoomPlayWaveDone() event triggers when a wave file played successfully for a specific conference room created by OpenConferenceRoom().

**Syntax**

```
OnConferenceRoomPlayWaveDone
(
    RoomName,
    WaveId
)
```

**Parameters**

RoomName (string)

This parameter specifies the name of conference room.

WaveId (integer)

This parameter value specifies the unique identification of wave data to be played.

**Example**

```
OnConferenceRoomPlayWaveDone(RoomName, WaveId)
{
}
}
```

**See Also**

PlayWaveStartToConferenceRoom(), PlayWaveStopToConferenceRoom()

## OnCallSessionOpenCall()

The OnCallSessionOpenCall() event triggers when VaxTele sends or receives a call request.

### Syntax

```
OnCallSessionOpenCall(  
    SessionId,  
    CallType,  
    PeerType,  
    PeerName  
)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

PeerType (integer)

This parameter value specifies the peer type.

3001 = User PeerType

3002 = Line PeerType

3003 = DirectProxy PeerType

PeerName (string)

The value of this parameter specifies the peer name.

### Example

```
OnCallSessionOpenCall(SessionId, CallType, PeerType, PeerName)  
{  
  
}
```

### See Also

OnCallSessionCloseCall()

## OnCallSessionCloseCall()

The OnCallSessionCloseCall() event triggers when VaxTele close a call of a call-session.

### Syntax

```
OnCallSessionOpenCall(  
    SessionId,  
    CallType,  
    PeerType,  
    PeerName  
)
```

### Parameters

SessionId (integer)  
This parameter specifies the unique identification for a call-session.

CallType (integer)  
This parameter value specifies the call identification.

201 = From-Call  
202 = To-Call

PeerType (integer)  
This parameter value specifies the peer type.

3001 = User PeerType  
3002 = Line PeerType  
3003 = DirectProxy PeerType

PeerName (string)  
The value of this parameter specifies the peer name.

### Example

```
OnCallSessionCloseCall(SessionId, CallType, PeerType, PeerName)  
{  
  
}
```

### See Also

OnCallSessionOpenCall()

**OnCallSessionDetectAMD()**

The OnCallSessionDetectAMD() event triggers when request for detection of answering machine on a specific call of a particular call-session successfully completes.

**Syntax**

```
OnCallSessionDectecAMD(  
    SessionId,  
    CallType,  
    IsHuman  
)
```

**Parameters**

SessionId (integer)

This parameter specifies a unique identification of a Call-Session.

CallType (integer)

This parameter value specifies the call identification.

201 = From-Call

202 = To-Call

IsHuman (boolean)

This parameter value can be 0 or 1. The value 1 corresponds to human voice and value 0 corresponds to answering machine.

**Example**

```
OnDetectAMD(SessionId, CallType, IsHuman)  
{  
}
```

**See Also**

CallSessionDetectAMD()

## OnBusyLampSubscribe()

The OnBusyLampSubscribe() event triggers when VaxTele receives BLF subscribe request.

### Syntax

```
OnBusyLampSubscribe(  
    SubscribId,  
    UserName,  
    ToUserName,  
    FromIP,  
    FromPort  
)
```

### Parameters

SubscribId (integer)  
This parameter value specifies a unique subscribe identification.

UserName (string)  
This parameter value specifies BLF subscribe user.

ToUserName (string)  
This parameter value specifies BLF monitor user.

FromIP (string)  
This parameter value specifies the from IP address.

FromPort (integer)  
This parameter value specifies the from Port number.

### Example

```
OnBusyLampSubscribe(SubScribId, UserName, ToUserName, FromIP,  
    FromPort)  
{  
    BusyLampSubscribeAccept(SubScribId, 1, 1800)  
}
```

### See Also

OnBusyLampUnSubscribe(), BusyLampSubscribeAccept(),  
BusyLampSubscribeReject(), OnBusyLampSubscribeSuccess()

## OnBusyLampUnSubscribe()

The OnBusyLampUnSubscribe() event triggers when VaxTele receives BLF unSubscribe request.

### Syntax

```
OnBusyLampUnSubscribe(  
    SubscribId,  
    UserName,  
    ToUserName,  
    FromIP,  
    FromPort  
)
```

### Parameters

SubscribId (integer)

This parameter value specifies a unique subscribe identification.

UserName (string)

This parameter value specifies BLF subscribe user.

ToUserName (string)

This parameter value specifies BLF monitor user.

FromIP (string)

This parameter value specifies the from IP address.

FromPort (integer)

This parameter value specifies the from Port number.

### Example

```
OnBusyLampUnSubscribe(SubscribId, UserName, ToUserName, FromIP,  
    FromPort)  
{  
}
```

### See Also

OnBusyLampSubscribe()

**OnBusyLampSubscribeSuccess()**

The OnBusyLampSubscribeSuccess() event triggers when the BLF subscribe request completes successfully.

**Syntax**

```
OnBusyLampSubscribeSuccess(  
                                SubscribId,  
                                UserName  
                                )
```

**Parameters**

SubscribId (integer)

This parameter value specifies a unique subscribe identification.

UserName (string)

This parameter value specifies BLF subscribe user.

**Example**

```
OnBusyLampSubscribe(SubScribId, UserName, ToUserName, FromIP,  
                    FromPort)  
{  
    BusyLampSubscribeAccept(SubScribId, 1, 1800)  
}  
  
OnBusyLampSubscribeSuccess(SubScribId, UserName)  
{  
  
}
```

**See Also**

OnBusyLampSubscribe(), OnBusyLampSubscribeFailed()

**OnBusyLampSubscribeFailed()**

The OnBusyLampSubscribeFailed() event triggers when the BLF subscribe request fails to complete.

**Syntax**

```
OnBusyLampSubscribeFailed(  
    SubscribId,  
    UserName  
)
```

**Parameters**

SubscribId (integer)

This parameter value specifies a unique subscribe identification.

UserName (string)

This parameter value specifies BLF subscribe user.

**Example**

```
OnBusyLampSubscribe(SubScribId, UserName, ToUserName, FromIP,  
    FromPort)  
{  
    BusyLampSubscribeAccept(SubScribId, 1, 1800)  
}  
  
OnBusyLampSubscribeFailed(SubScribId, UserName)  
{  
  
}
```

**See Also**

OnBusyLampSubscribeSuccess(), OnBusyLampSubscribe(),  
OnBusyLampUnSubscribe()



## OnCallSessionCreated()

The OnCallSessionCreated() event triggers when VaxTele creates a call-session internally.

### Syntax

```
OnCallSessionCreated(  
    SessionId,  
    ReasonCode  
)
```

### Parameters

SessionId (integer)

This parameter specifies the unique identification for a call-session.

ReasonCode(integer)

This parameter specifies the reason due to which the call-session is created.

- INCOMING\_CALL      1001
- OUTGOING\_CALL     1002
- MERGED             1003
- TRANSFERED        1004

### Example

```
OnCallSessionCreated(SessionId, ReasonCode)  
{  
    CCustomCallSessionData objData = new CCustomCallSessionData()  
    SetCallSessionData(SessionId, objData)  
}
```

### See Also

OnCallSessionClosed(), SetCallSessionData()

## OnCallSessionClosed()

The OnCallSessionClosed() event triggers when VaxTele close a call-session internally.

### Syntax

```
OnCallSessionClosed(SessionId, CallType, ReasonCode)
```

### Parameters

SessionId (integer)

This parameter specifies a unique identification of a call-session.

CallType (integer)

This parameter value specifies the call that is responsible for closing the call-session.

201 = From-Call

202 = To-Call

ReasonCode (integer)

This parameter specifies the reason due to which the call-session is closed.

- |                |      |
|----------------|------|
| - HANGUP       | 2001 |
| - SESSION_LOST | 2002 |
| - MERGED       | 2003 |
| - TRANSFERED   | 2004 |
| - REJECTED     | 2005 |
| - FAILED       | 2006 |
| - CANCELLED    | 2007 |
| - TIMEOUT      | 2008 |
| - CLOSED       | 2009 |

### Example

```
OnCallSessionClosed(SessionId, CallType, ReasonCode)
{
    CCustomCallSessionData objData = GetCallSessionData(SessionId)
    objData = nil
}
```

### See Also

OnCallSessionCreated(), SetCallSessionData(), GetCallSessionData()

## OnCallSessionRecordedPCM()

The OnCallSessionRecordedPCM() event starts triggering when recording on a specific call-session starts by using RecordWaveStartToCallSession() with value nil or 0 to the second parameter.

e.g. RecordWaveStartToCallSession(SessionId, 0)

OnCallSessionRecordedPCM() can be used to integrate third-party SAPI (Speech recognition) engines or libraries. SAPI engines work with voice data and generate text data. In this event, VaxTele delivers PCM data of format (8000Hz, 16bit, Mono) to the application, application pass it to SAPI engine and SAPI engine generates text sentence/data. For more details about SAPI engines, please contact to the vendor of SAPI engine.

OnCallSessionRecordedPCM() can also be used to record or save conversation into .mp3 or other media files by using third-party libraries. In this event, application receives voice data PCM (8000Hz, 16bit, Mono), pass it to third-party library and library stores it in the required media file format. Please contact the third-party vendor for further details.

## Syntax

```
OnCallSessionRecordedPCM(SessionId, DataPCM, SizePCM)
```

## Parameters

SessionId (integer)

This parameter specifies a unique identification of a call-session.

DataPCM (array)

The value of this parameter specifies the voice PCM data.

SizePCM (integer)

The value of this parameter specifies the size of voice PCM data.

## Example

```
OnCallSessionRecordedPCM(SessionId, DataPCM, SizePCM)
{
    // Pass PCM data to SAPI engine.
}
```

## See Also

RecordWaveStartToCallSession()

## OnConferenceRoomRecordedPCM()

The OnConferenceRoomRecordedPCM() event starts triggering when recording on a specific conference room starts by using RecordWaveStartToConferenceRoom() with value nil or 0 to the second parameter.

e.g. RecordWaveStartToConferenceRoom(RoomName, 0)

OnConferenceRoomRecordedPCM() can be used to integrate third-party SAPI (Speech recognition) engines or libraries. SAPI engines work with voice data and generate text data. In this event, VaxTele delivers PCM data of format (8000Hz, 16bit, Mono) to the application, application pass it to SAPI engine and SAPI engine generates text sentence/data. For more details about SAPI engines, please contact to the vendor of SAPI engine.

OnConferenceRoomRecordedPCM() can also be used to record or save conversation into .mp3 or other media files by using third-party libraries. In this event, receive voice data PCM (8000Hz, 16bit, Mono), pass it to third-party library and library stores it in the required media file format. Please contact the third-party vendor for further details.

## Syntax

```
OnConferenceRoomRecordedPCM(RoomName, DataPCM, SizePCM)
```

## Parameters

RoomName (string)

This parameter specifies the name of a conference room.

DataPCM (array)

The value of this parameter specifies the voice PCM data.

SizePCM (integer)

The value of this parameter specifies the size of voice PCM data.

## Example

```
OnConferenceRoomRecordedPCM(RoomName, DataPCM, SizePCM)
{
    // Pass PCM data to SAPI engine.
}
```

## See Also

RecordWaveStartToConferenceRoom()

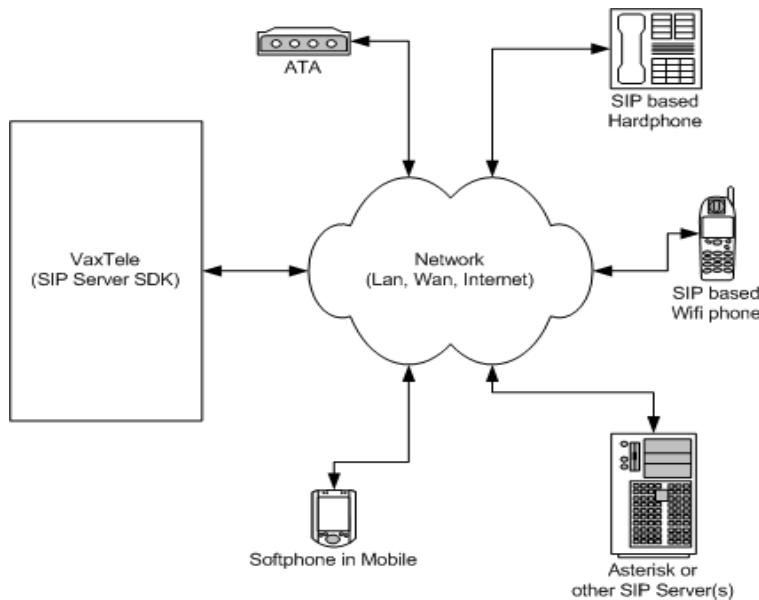
## **SIP CLIENT REGISTRATION FLOW**

SIP client send a registration request.
<b>Event:</b> OnRegisterUser()
<b>Method:</b> AddUser()
<b>Method:</b> AuthRegister()
<b>Event:</b> OnRegisterUserSuccess()
SIP client successfully registered.
Registered client can successfully sends/receives/process call requests.
SIP client sends the unregister request.
<b>Event:</b> OnUnregisterUser()
<b>Method:</b> RemoveUser()
SIP client Un-registered successfully.

## **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 softphones, hardphones and devices 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.

Other SIP servers can also be connected as SIP client to VaxTele and can forward the calls to VaxTele SIP Server.



### **CALL BETWEEN TWO SIP CLIENTS**

SIP client-A sends a call request.

**Event:** OnCallSessionOpen()

**Method:** AcceptCallSession()

**Event:** OnCallSessionConnecting() // CallType = To-Call(SIP client B)

At SIP client-B incoming call appears.

SIP client-B accepts the incoming call.

Call-Session established between SIP client-A and SIP client-B.

**Event:** OnCallSessionConnected()

Voice streaming starts between SIP client-A and SIP client-B.

SIP client-A disconnect the Call.

**Event:** OnCallSessionHangUp() CallType = From-Call (if SIP client A hang up phone)  
CallType = To-Call (if SIP client B hang up phone)

**Event:** OnCallSessionClosed()

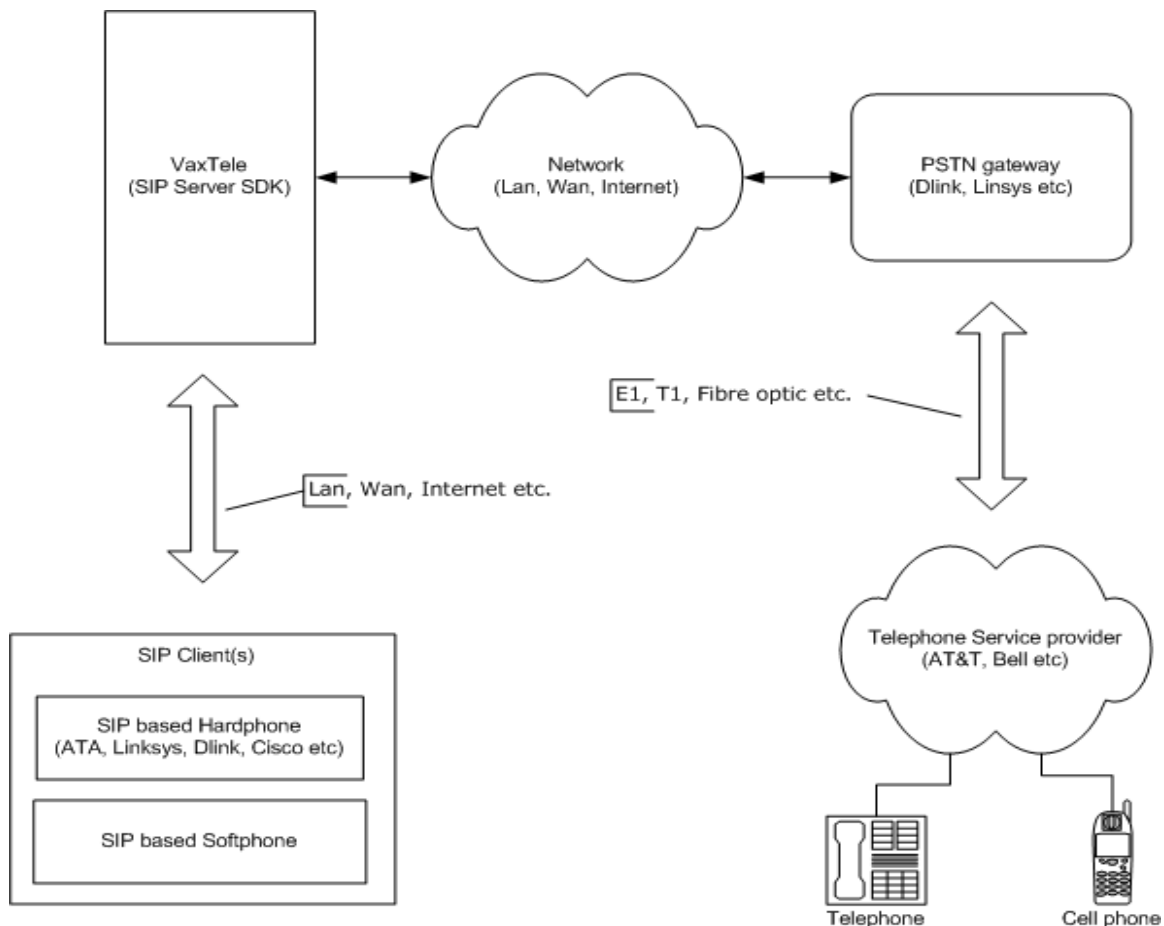
## **SIP PHONE TO PSTN CALL FLOW**

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 available in the market, 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 used 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

PSTN gateway is configure 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

<b>Method:</b> AddUser()	//User Name for SIP client
<b>Method:</b> AddUser()	// User Name for PSTN gateway
PSTN gateway and SIP client registers successfully.	
SIP client(softphone/hardphone/ATA etc) dials phone number.	
<b>Event:</b> OnCallSessionOpen()	// FromPeerType = USER, FromPeerName = Client User Name
<b>Method:</b> AcceptCallSession()	// ToPeerType = USER, ToPeerName = Gateway User Name.
<b>Event:</b> OnCallSessionConnecting()	// CallType = To-Call (Gateway user)
VaxTele sends dial call request to PSTN gateway.	
Mobile/PSTN phone starts ringing.	
Mobile/PSTN phone user accepts the incoming call.	
<b>Event:</b> OnCallSessionConnected()	
Voice streaming starts between SIP client and Mobile/PSTN phone user.	
SIP client OR Mobile/PSTN phone user disconnects the Call.	
<b>Event:</b> OnCallSessionHangUp()	// CallType = From-Call (if SIP client hang up phone) CallType = To-Call (if Gateway user hang up phone)
<b>Event:</b> OnCallSessionClosed()	

## RECEIVE CALL FROM PSTN GATEWAY

<b>Method:</b> AddUser()	//User Name for SIP client
<b>Method:</b> AddUser()	// User Name for PSTN gateway
PSTN gateway and SIP client registers successfully.	
Mobile/PSTN phone user dials phone number.	
<b>Event:</b> OnCallSessionOpen()	// FromPeerType = USER, FromPeerName = Gateway User Name
<b>Method:</b> AcceptCallSession()	// ToPeerType = USER, ToPeerName = Client User Name.
VaxTele sends dial call request to SIP client(softphone/hardphone/ATA etc).	
<b>Event:</b> OnCallSessionConnecting()	// CallType = To-Call (SIP client)
At SIP client ringing starts.	
SIP client accepts the incoming call.	
<b>Event:</b> OnCallSessionConnected()	
Voice streaming starts between SIP client and Mobile/PSTN phone user.	
SIP client OR Mobile/PSTN phone user disconnects the Call.	
<b>Event:</b> OnCallSessionHangUp()	//CallType = From-Call (if PSTN user hang up phone) CallType = To-Call (if SIP client user hang up phone)
<b>Event:</b> OnCallSessionClosed()	

## PSTN GATEWAY AS DIRECT IP TO IP COMMUNICATION

PSTN gateway is configured in a way that it directly receives call requests on listen IP and send call requests directly to the VaxTele SIP Server IP.

### DIAL CALL TO PSTN GATEWAY

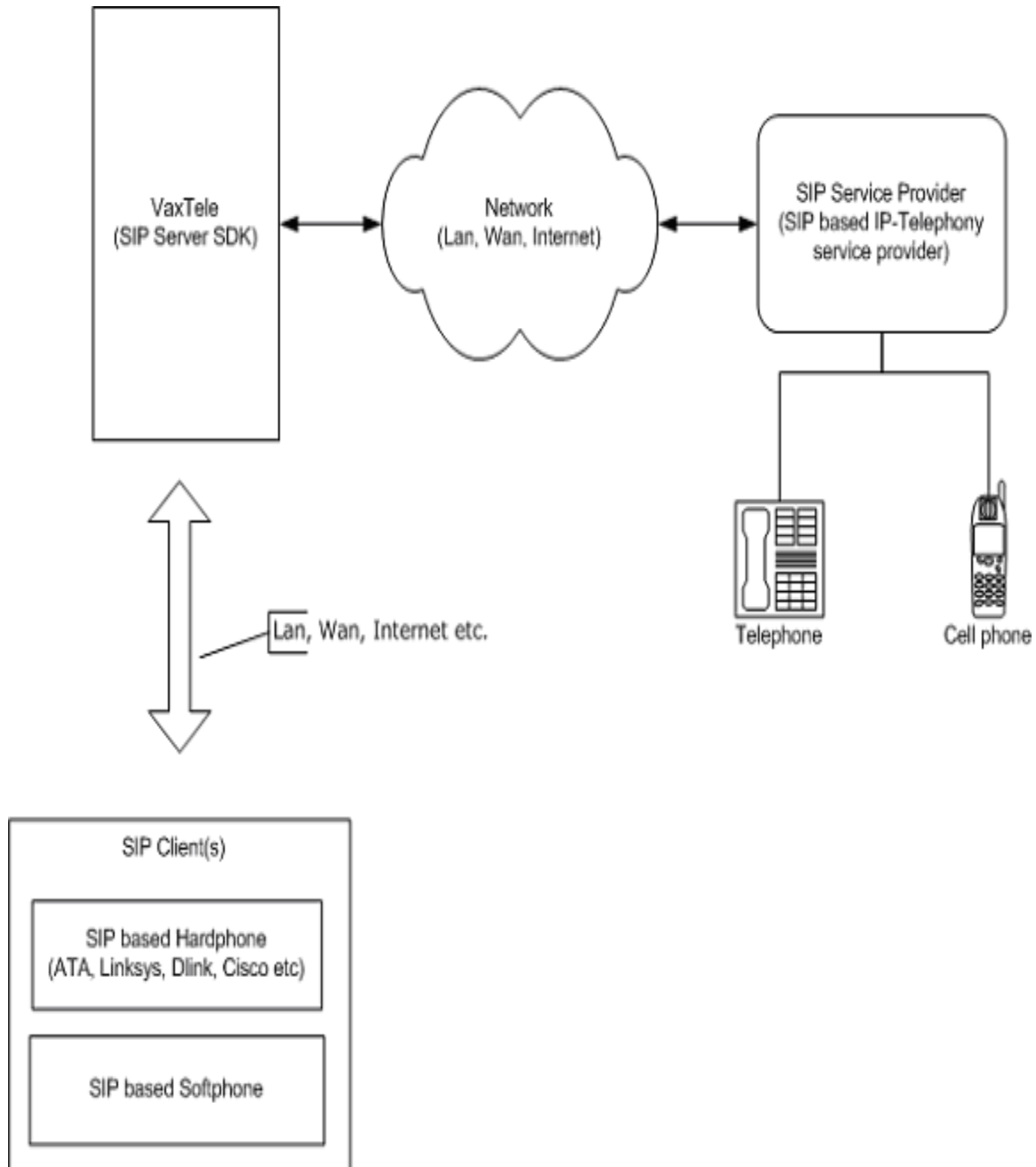
<b>Method:</b> AddUser()	//User Name for SIP client
<b>Method:</b> AddDirectProxySIP()	//User Name for PSTN gateway
PSTN gateway and SIP client registers successfully.	
SIP client (softphone/hardphone/ATA etc) dials phone number.	
<b>Event:</b> OnCallSessionOpen()	// FromPeerType = USER, FromPeerName = Client User Name
<b>Method:</b> AcceptCallSession()	// ToPeerType = DIRECT_PROXY, ToPeerName = Direct Proxy Name
VaxTele sends dial call request to PSTN gateway.	
<b>Event:</b> OnCallSessionConnecting()	// CallType = To-Call(PSTN Gateway)
Mobile/PSTN phone starts ringing.	
Mobile/PSTN phone user accepts the incoming call.	
<b>Event:</b> OnCallSessionConnected()	
Voice streaming starts between SIP client and Mobile/PSTN phone user.	
SIP client OR Mobile/PSTN phone user disconnects the Call.	
<b>Event:</b> OnCallSessionHangUp()	//CallType = From-Call (if SIP client hang up phone) CallType = To-Call (if PSTN gateway disconnects call)
<b>Event:</b> OnCallSessionClosed()	

RECEIVE CALL FROM PSTN GATEWAY

<b>Method:</b> AddUser() //User Name for SIP client
<b>Method:</b> AddDirectProxySIP() // User Name for PSTN gateway
PSTN gateway and SIP client registers successfully.
Mobile/PSTN phone user dials phone number.
<b>Event:</b> OnCallSessionOpen() // FromPeerType = DIRECT_PROXY, FromPeerName = Direct Proxy Name
<b>Method:</b> AcceptCallSession() // ToPeerType = USER, ToPeerName = Client User Name
VaxTele sends dial call request to SIP client(softphone/hardphone/ATA etc).
<b>Event:</b> OnCallSessionConnecting() // CallType = To (SIP client)
At SIP client ringing starts.
SIP client accepts the incoming call.
<b>Event:</b> OnCallSessionConnected()
Voice streaming starts between SIP client and Mobile/PSTN phone user.
SIP client OR Mobile/PSTN phone user disconnects the Call.
<b>Event:</b> OnCallSessionHangUp() CallType = From-Call(if PSTN gateway user hang up) CallType = To-Call (if SIP client hang up phone)
<b>Event:</b> OnCallSessionClosed()

## **CONNECT VAXTELE WITH IP-TELEPHONY SERVICE PROVIDER (ITSP)**

There are many ITSP (IP-Telephony Service providers) 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 & landline phones.



### DIAL CALL TO SERVICE PROVIDER (ITSP)

<b>Method:</b> AddUser()	//User Name for SIP client
<b>Method:</b> AddLine()	// User Name for IP-Telephony service provider (ITSP)
<b>Method:</b> RegisterLine()	
VaxTele server and SIP client registers successfully.	
SIP client dials phone number.	
<b>Event:</b> OnCallSessionOpen()	// FromPeerType = USER, FromPeerName = Client User Name
<b>Method:</b> AcceptCallSession()	// ToPeerType = LINE, ToPeerName = Line Name
VaxTele sends dial call request to SIP service provider(ITSP)	
<b>Event:</b> OnCallSessionConnecting()	// CallType = To-Call(ITSP)
ITSP accepts the call.	
<b>Event:</b> OnCallSessionConnected()	
Voice streaming starts between SIP client and ITSP phone user.	
SIP client OR ITSP phone user disconnects the Call.	
<b>Event:</b> OnCallSessionHangUp()	//CallType = From-Call (if SIP client hang up phone) CallType = To-Call (if ITSP hang up the call)
<b>Event:</b> OnCallSessionClosed()	

RECEIVE CALL FROM SERVICE PROVIDER (ITSP)

<b>Method:</b> AddUser()	//User Name for SIP client
<b>Method:</b> AddLine()	// User Name for IP-Telephony service provider (ITSP)
<b>Method:</b> RegisterLine()	
VaxTele server and SIP client registers successfully.	
ITSP send dial call request on specific phone number.	
<b>Event:</b> OnCallSessionOpen()	// FromPeerType = LINE, FromPeerName = Line Name
<b>Method:</b> AcceptCallSession()	// ToPeerType = USER, ToPeerName = Client User Name
VaxTele dials call to respective SIP client	
At SIP client ringing starts.	
<b>Event:</b> OnCallSessionConnecting()	// CallType = To-Call (SIP client)
SIP client accepts the incoming call.	
<b>Event:</b> OnCallSessionConnected()	
Voice streaming starts between SIP client and ITSP phone user.	
SIP client OR ITSP phone user disconnects the Call.	
<b>Event:</b> OnCallSessionHangUp()	//CallType = To-Call (if ITSP hang up the call) CallType = From-Call (if SIP client hang up phone)
<b>Event:</b> OnCallSessionClosed()	

## **LIST OF SIP RESPONSES (SIP RFC 3261)**

### Provisional responses 1xx

100	Trying	180	Ringling
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



## **CALL-SESSION**

Call-Session is a collection of either one or two calls. In a Call-Session, calls are identified as From-Call and To-Call.

Call-Session with two calls (From-Call & To-Call)

From-Call	To-Call
-----------	---------

Call-Session with One call (From-Call)

From-Call	
-----------	--

Call-Session with One call (To-Call)

	To-Call
--	---------

### **Call-Session with two calls (From-Call & To-Call)**

VaxTele COM component receives SIP based call request(s), upon receiving call request(s) it internally creates a new Call-Session. It adds incoming call as From-Call to Call-Session and triggers OnCallSessionOpen() event.

```
OnCallSessionOpen (SessionId, CallerName, CallerId, FromPeerType,
                    FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, CallerName, CallerId, DialNo,
                      ToPeerType, ToPeerName, Timeout)
}
```

AcceptCallSession() with appropriate values in the DialNo, ToPeerType, ToPeerName, sends a SIP based call request and adds outgoing call as To-Call to the same Call-Session (created upon incoming call). The AcceptCallSession() also accepts/connects the From-Call in that Call-Session.

### **Call-Session with One call (From-Call)**

VaxTele COM component receives SIP based call request(s), upon receiving call request(s) it internally creates a new Call-Session. It adds incoming call as From-Call to Call-Session and triggers OnCallSessionOpen() event.

```
OnCallSessionOpen(SessionId, CallerName, CallerId, FromPeerType,
                    FromPeerName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, CallerName, CallerId, -1, "", "", Timeout)
}
```

To only accepts/connects the incoming call without generating outgoing call  
AcceptCallSession() is called with no values in DialNo, ToPeerType and  
ToPeerName.

### **Call-Session with One call (To-Call)**

VaxTele COM component exports a function OpenCallSession(), that function  
sends/dials a SIP based call request. Such method is used to develop predictive  
dialer and telemarketing softwares.

```
OnVaxTeleTick(TickId)
{
    SessionId = OpenCallSession(CallerName, CallerId, DialNo,
                                ToPeerType, ToPeerName, Timeout)
}
```

OpenCallSession() internally creates a new Call-Session and adds that outbound call  
as To-Call to the Call-Session.

## **FUNCTION AND CALL SESSION**

VaxTele COM component exports a set of different functions/methods and events. Each method and event performs a certain action on the Call-Session.

### **Method: AcceptCallSession()**

It performs two actions depending upon the values of DialNo, ToPeerType and ToPeerName values.

- It generates To-Call and accepts the From-Call.
- It only accepts the From-Call.

### **Method: PlayWaveStartToCallSession()**

It starts playing the wave file in a Call-Session. If there are two calls connected in the Call-Session as in case of Call-Session with two call, it plays wave data on both calls and persons on both calls listen the playing wave file.

If there is only one call connected in the Call-Session as in case of Call-Session with one call, it plays wave data on that call and person on that call listen the playing wave file.

### **Method: RecordWaveStartToCallSession()**

It start recording of the voice conversation into wave file.

If there are two calls connected as in case of Call-Session with two call then it records the conversation of both calls in real-time but if there is only one call in the Call-Session as in case of Call-Session with one call then it record voice of the that call.

### **Event: OnCallSessionConnected()**

It triggers when the call(s) get connected in a Call-Session.

If there are two calls in the Call-Session then such event triggers when both calls get connected.

If there is only one call in the Call-Session then it triggers when that call gets connected.