SIP SERVER SDK v8.0

TECHNICAL DOCUMENTATION

VERSION 8.0.2.6

CONTENTS

INTRODUCTION AND QUICK START

The VaxVoIP COM (Component Object Model) component, specifically the VaxTeleServerCOM.dll, comprises a set of functions and events designed for the development of SIP (Session Initiation Protocol) based servers, IP-PBX, IVR, Telemarketing systems, and other IP-Telephony related services.

With its diverse functions and events, the VaxVoIP COM component facilitates the swift creation of SIP-based servers. It is versatile enough to be employed in call centers as a telephony server, as a virtual PBX in offices to connect multiple remote locations, for providing call routing services, and offering PC-to-phone IP-Telephony services, among other applications.

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 VaxVoIP.

Return Value

No return value.

Example

```
SetLicenseKey("LicenseKey")
Initialize("sipsdk.com")
```

See Also

Initialize(), GetVaxErrorCode()

GetVaxErrorCode()

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

Please see **LIST OF ERROR CODES** for more details.

Syntax

```
integer GetVaxErrorCode()
```

Parameters

No parameters.

Return Value

The GetVaxErrorCode() returns the error code.

Example

```
SetLicenseKey("LicenseKey")

Result = Initialize("")
if(Result == 0) GetVaxErrorCode()
```

See Also

Initialize(), SetLicenseKey()

Initialize()

The Initialize() function is responsible for configuring the VaxVoIP SIP Server COM component. This involves allocating internal memory resources and preparing the component for integration with the application.

Upon execution of this function, the component is primed to expose its exported methods. This readiness enables the application to seamlessly interact with the component, making use of its various functionalities.

Syntax

```
boolean Initialize(DomainRealm)
```

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 assigned IP value is use in SIP authentication process and to create SIP URI(s).

e.g. if value of DomainRealm is sip.vaxvoip.com then VaxTele creates SIP URI sip:username@sip.vaxvoip.com

But if value of DomainRealm is "" empty string and assigned IP 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.

Return Value

Upon successful execution, this function returns a non-zero value; otherwise, it returns 0. To obtain specific error details, the GetVaxErrorCode() method can be invoked.

Example

```
Initialize("")
Initialize("sip.vaxvoip.com")
Initialize("vaxvoip.com")
```

See Also

UnInitialize(), GetVaxErrorCode(), SetListenPortRangeRTP()

OpenNetworkUDP()

The OpenNetworkUDP() function opens UDP socket/network. It allocates UDP listen port and starts listening for incoming UDP based SIP requests and triggers the COM (Component Object Model) events accordingly.

Syntax

```
boolean OpenNetworkUDP(
                        ListenIP,
                        ListenPort
                        )
```

Parameters

ListenIP (string)

The value of this parameter specifies the IP on which VaxTele listen for incoming UDP based SIP requests.
It can be empty value or an IP address assigned to the computer on which VaxTele COM integrated SIP server is running.

ListenPort (integer)

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

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 GetVaxErrorCode() method.

Example

```
OpenNetworkUDP(“”, 5060)
OpenNetworkUDP(“192.168.0.3”, 5060)
```

See Also

OpenNetworkTCP(), OpenNetworkTLS(), CloseNetworkUDP(), CloseNetworkTCP(), CloseNetworkTLS(), GetVaxErrorCode(), SetListenPortRangeRTP()

OpenNetworkTCP()

The OpenNetworkTCP() function opens TCP socket/network. It allocates TCP listen port and starts listening for incoming TCP based SIP requests and triggers the COM (Component Object Model) events accordingly.

Syntax

```
boolean OpenNetworkTCP(
                        ListenIP,
                        ListenPort
                        )
```

Parameters

ListenIP (string)
: The value of this parameter specifies the IP on which VaxTele listen for incoming TCP based SIP requests.
It can be empty value or an IP address assigned to the computer on which VaxTele COM integrated SIP server is running.

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

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 GetVaxErrorCode() method.

Example

```
OpenNetworkTCP("", 5060)
OpenNetworkTCP("192.168.0.3", 5060)
```

See Also

OpenNetworkUDP(), OpenNetworkTLS(), CloseNetworkUDP(), CloseNetworkTCP(), CloseNetworkTLS(), GetVaxErrorCode(), SetListenPortRangeRTP()

OpenNetworkTLS()

The OpenNetworkTCP() function opens TLS communication channel/network. It allocates TLS listen port and starts listening for incoming TLS based SIP requests and triggers the COM (Component Object Model) events accordingly.

Syntax

```
boolean OpenNetworkTLS(
                        ListenIP,
                        ListenPort,
                        CertPEM
                        )
```

Parameters

ListenIP (string)

The value of this parameter specifies the IP on which VaxTele listen for incoming TLS based SIP requests.
It can be empty value or an IP address assigned to the computer on which VaxTele COM integrated SIP server is running.

ListenPort (integer)

The value of this parameter specifies the port number for SIP server to receive SIP requests. Standard SIP listen port is 5061

CertPEM (string)

The value of this parameter specifies the SSL certificate (.PEM) file name or Data of (.PEM) file in the form of string/text value.
It can be empty value and VaxTele COM starts using the default VaxVoIP SSL certificate.

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 GetVaxErrorCode() method.

Example

```
OpenNetworkTLS(“”, 5061, “”)
OpenNetworkTLS(“192.168.0.3”, 5061, “”)
```

See Also

OpenNetworkUDP(), OpenNetworkTCP(), CloseNetworkUDP(), CloseNetworkTCP(), CloseNetworkTLS(), GetVaxErrorCode(), SetListenPortRangeRTP()

CloseNetworkUDP()

The CloseNetworkUDP() function closes UDP socket/network.

Syntax

```
CloseNetworkUDP()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkUDP("", 5060)
CloseNetworkUDP()
```

See Also

OpenNetworkTUDP(), OpenNetworkTCP(), OpenNetworkTLS(), CloseNetworkTCP(), CloseNetworkTLS(), GetVaxErrorCode(), SetListenPortRangeRTP()

CloseNetworkTCP()

The CloseNetworkTCP() function closes TCP socket/network.

Syntax

```
CloseNetworkTCP()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkTCP("", 5060)
CloseNetworkTCP()
```

See Also

OpenNetworkTUDP(), OpenNetworkTCP(), OpenNetworkTLS(), CloseNetworkUDP(), CloseNetworkTLS(), GetVaxErrorCode(), SetListenPortRangeRTP()

CloseNetworkTLS()

The CloseNetworkTLS() function closes TLS communication channel/network.

Syntax

```
CloseNetworkTLS()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkTLS(“”, 5061, “”)
CloseNetworkTLS()
```

See Also

OpenNetworkTUDP(), OpenNetworkTCP(), OpenNetworkTLS(), CloseNetworkUDP(), CloseNetworkTCP(), GetVaxErrorCode(), 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)
: This parameter's value designates the starting port of the specified range.

ListenEndPort (integer)
: This parameter's value designates the ending port of the specified 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 GetVaxErrorCode() method.

Example

```
Result = Initialize("192.168.0.125")
if(Result == 0) GetVaxErrorCode()

SetListenPortRangeRTP(8088, 32406)
```

See Also

Initialize(), GetVaxErrorCode()

AddNetworkRouteSIP(), AddNetworkRouteRTP()

The AddNetworkRouteSIP() & AddNetworkRouteRTP() functions functions enable the addition of supplementary IP addresses to facilitate the routing of SIP and RTP packets, respectively.

In cases where a VaxVoIP integrated application is positioned behind a firewall, router, or NAT, with port forwarding enabled from the router's end, it is recommended to initialize VaxTele with the private IP address assigned to the computer.

Additionally, use the AddNetworkRouteSIP() and AddNetworkRouteRTP() functions to incorporate the public IP address assigned to the router, ensuring proper routing in this network configuration.

Syntax

```
boolean AddNetworkRouteSIP(AssignedIP, RouterIP)
boolean AddNetworkRouteRTP(AssignedIP, RouterIP)
```

Parameters

AssignedIP (string)
: This parameter specifies the IP address assigned to the computer.

RouterIP (string)
: This parameter specifies the IP address assigned to the router.

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 GetVaxErrorCode() method.

Example

```
Result = Initialize("192.168.0.125")
if(Result = 0) GetVaxErrorCode()

AddNetworkRouteSIP("192.168.0.25", "66.77.88.99")
AddNetworkRouteRTP("192.168.0.25", "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(), GetVaxErrorCode()

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.

Add user by AddUser() function so that user login and password can be used in any SIP based softphone or hardphone to connect and register with VaxTele based SIP server.

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 GetVaxErrorCode() method.

Example

```
SetLicenseKey("LicenseKey")
Initialize("192.168.0.125")

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

See Also

RemoveUser(), GetVaxErrorCode()

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 the expiration interval value that will be added by the VaxVoIP SIP SERVER during the SIP client registration process. This setting defines how long the registration information for a SIP client will remain valid before requiring renewal.

If the value of -1 is set, the VaxVoIP-based SIP SERVER will accept the Expiration Interval requested by the SIP client during the registration process. On the other hand, if a value other than -1 is set, the SIP SERVER will ignore the Expiration Interval requested by the SIP client and instead send the time specified by calling the RegisterUserExpiry() function. This allows for customized control over the expiration interval during the registration process.

If the RegisterUserExpiry() function is not explicitly called, the VaxVoIP-based SIP SERVER will use the default value of 30 seconds for the expiration interval, which is then communicated to the SIP client(s) during the registration process.

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 GetVaxErrorCode() 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** for more details.

Syntax

```
boolean AcceptRegister(RegId)
```

Parameters

RegId (integer)
: This parameter specifies a unique identification of a registration 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 GetVaxErrorCode() method.

Example

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

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** for more details.

Syntax

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

Parameters

RegId (integer)
: This parameter specifies a unique identification of a registration session.

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 GetVaxErrorCode() method.

Example

```
OnRegisterUser(UserLogin, Domain, UserAgentName, FromIP, FromPort,
                    RegId)
{
        RejectRegister(RegId, 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.

Use 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** for more details.

Syntax

```
boolean AuthRegister(RegId)
```

Parameters

RegId (integer)
: This parameter specifies a unique identification of a registration 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 GetVaxErrorCode() method.

Example

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

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 **HOW TO CONNECT TO PSTN/GSM NETWORK** for more details.

Please see **HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP)** for more details.

Syntax

```
boolean AddLine(
                    LineName,
                    LineType
                    DisplayName,
                    UserName,
                    AuthUser
                    AuthPwd,
                    DomainRealm,
                    ServerAddr,
                    ServerPort,
                    AudioCodecList
                    )
```

Parameters

LineName (string)
: This parameter value specifies the unique line name to identify a specific line.

LineType (integer)
This parameter value specifies the line type.

0 = Line Type UDP
1 = Line Type TCP
2 = Line Type TLS

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.

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

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

AudioCodecList (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 GetVaxErrorCode() method.

Example

```
AddLine("LineNameA", 0, "8034", "8034", "8034", "9341", "sipsdk.com",
        "192.168.0.3", 5060, "32")
```

See Also

RemoveLine(), RegisterLine(), UnRegisterLine(), GetVaxErrorCode()

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 GetVaxErrorCode() method.

Example

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

See Also

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

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 GetVaxErrorCode() method.

Example

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

See Also

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

AddUserRouteUID()

The AddUserRouteUID() function adds unique Route-Id as DialIn-No to a specific user.

Syntax

```
boolean AddUserRouteUID(UserName, RouteUID)
```

Parameters

UserName (string)
: This parameter value specifies the user's login.

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-No assigned to Route-Types 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 GetVaxErrorCode() method.

Example

```
AddUserRouteUID("6868", "9090")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddLineRouteUID()

The AddLineRouteUID() function adds unique Route-Id as DialIn-Prefix to a specific line.

Syntax

```
boolean AddLineRouteUID(LineName, RouteUID, DialOutPrefix)
```

Parameters

LineName (string)
: This parameter value specifies the unique line name to identify a specific line.

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-Prefix assigned to Route-Types LINE.

DialOutPrefix (string)
: This parameter value specifies the prefix to be replaced with DialIn-Prefix/Route-Id.

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 GetVaxErrorCode() method.

Example

```
AddLineRouteUID("LineServiceProvider", “00”, “001”)
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddRingGroupRouteUID()

The AddRingGroupRouteUID() function adds unique Route-Id as DialIn-No to a specific ring-group.

Syntax

```
boolean AddRingGroupRouteUID(GroupName, RouteUID)
```

Parameters

GroupName (string)
: This parameter value specifies the unique name to identify a specific ring-group.

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-No assigned to Route-Types RING-GROUP.

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 GetVaxErrorCode() method.

Example

```
AddRingGroupRouteUID("RingGroupSales", “40044”)
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddCallPickUpGroupRouteUID()

The AddCallPickUpGroupRouteUID() function adds unique Route-Id as DialIn-Prefix to activate pickup-call.

Syntax

```
boolean AddCallPickUpGroupRouteUID(RouteUID)
```

Parameters

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-Prefix assigned to Route-Types PICKUP-CALL.

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 GetVaxErrorCode() method.

Example

```
AddCallPickUpGroupRouteUID("#55#")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddQueueRouteUID()

The AddQueueRouteUID() function adds unique Route-Id as DialIn-No to a specific queue-name.

Syntax

```
boolean AddQueueRouteUID(QueueName, RouteUID)
```

Parameters

QueueName (string)
: This parameter value specifies the unique name to identify a specific queue.

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-No assigned to Route-Types QUEUE.

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 GetVaxErrorCode() method.

Example

```
AddQueueRouteUID("QueueSupport", "001415000001")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddConferenceRoomRouteUID()

The AddConferenceRoomRouteUID() function adds unique Route-Id as DialIn-No to a specific room-name.

Syntax

```
boolean AddConferenceRoomRouteUID(RoomName, RouteUID)
```

Parameters

RoomName (string)
: This parameter value specifies the unique name to identify a specific room.

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-No assigned to Route-Types 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 GetVaxErrorCode() method.

Example

```
AddConferenceRoomRouteUID ("RoomTalk", "200011")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddCallParkRouteUID()

The AddCallParkRouteUID() function adds unique Route-Id as DialIn-Prefix to activate call-parking.

Syntax

```
boolean AddCallParkRouteUID(RouteUID)
```

Parameters

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-Prefix assigned to Route-Types CALL-PARKING.

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 GetVaxErrorCode() method.

Example

```
AddCallParkRouteUID("#1#")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

AddStealthListenRouteUID()

The AddStealthListenRouteUID() function adds unique Route-Id as DialIn-Prefix to activate stealth-listening.

Syntax

```
boolean AddStealthListenRouteUID(RouteUID)
```

Parameters

RouteUID (string)
: This parameter specifies the unique Route-Id as DialIn-Prefix assigned to Route-Types STEALTH-LISTEN.

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 GetVaxErrorCode() method.

Example

```
AddStealthListenRouteUID("#3#")
```

See Also

AcceptCallSession(), AcceptTransferBlind(), OnIncomingCall(), OnCallSessionTransferBlind()

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 Channel-ZERO call and Channel-ONE call. A Call-Session may have one call i.e. Channel-ZERO call or Channel-ONE call or two calls i.e. Channel-ZERO call and Channel-ONE call (incoming and outgoing call).

Please see **WHAT IS CALL-SESSION** for more details.

The OnIncomingCall() 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,
                          ToPeerName,
                          RoutUID,
                          Timeout
                          )
```

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.

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).

RouteUID (string)
: This parameter specifies the name/Unique RouteId assigned to Queue, RingGroup, Line, User, StealthListen, Room etc. For further details, please have a look at AddQueueRouteUID(), AddUserRouteUID(), AddLineRouteUID(), AddStealthListenRouteUID() methods.

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, CallerName, CallerId, DialNo,
    DialNo, RouteUID, 20)
}

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, “”, “”, “”, “”, “”, 20)
}
```

See Also

RejectCallSession(), CloseCallSession(), DialCallSession(), GetVaxErrorCode(), OnIncomingCall()

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

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        RejectCallSession(SessionId)
}
```

See Also

AcceptCallSession(), CloseCallSession(), OnIncomingCall(), DialCallSession()

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 use CloseCallSession() method and the call in that session gets 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 GetVaxErrorCode() method.

Example

```
CloseCallSession(SessionId)
```

See Also

RejectCallSession(), AcceptCallSession(), DialCallSession(), OnCallSessionClosed()

DialCallSession()

A Call-Session consists of either one or two calls where the calls are identified as Channel-ZERO call and Channel-ONE call.

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

Please see **WHAT IS CALL-SESSION** for more details.

One of the typical scenario for use of DialCallSession() 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 DialCallSession(
                        CallerName,
                        CallerId,
                        DialNo,
                        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.

ToPeerName (string)
: The value of this parameter specifies the name of Peer (LineName or UserName).

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 GetVaxErrorCode().

Example

```
DialCallSession("8025", "8025", "8034", "UserName", 20)
```

See Also

RejectCallSession(), CloseCallSession(), AcceptCallSession()

AcceptTransferBlind()

The AcceptTransferBlind() function processes 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;

- Generates an outgoing call to TransferTo parameter's value and puts that call at the free/available channel of Transferer call-session.

Syntax

```
boolean AcceptTransferBlind(
                              TransfererSessionId,
                              CallerName,
                              CallerId,
                              TransferTo,
                              ToPeerName,
                              RouteUID,
                              Timeout
                              )
```

Parameters

TransfererSessionId (integer)
: This parameter specifies a unique identification of a Transferer Call-Session.

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.

ToPeerName (string)
: The value of this parameter specifies the name of To-Peer.

RouteUID (string)
: The value of this parameter specifies the UniqueId assigned as route. Please have a look at OnCallSessionTransferBlind() event.

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

On successful execution this function returns non-zero value otherwise it returns 0. To get extended error information, call GetVaxErrorCode().

Example

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererChannelId,
                           Transferer, Transferee, TransferTo,
                           RouteUID, RouteType, RouteName)
{
    AcceptTransferBlind(TransfererSessionId, Transferee, Transferee,
    TransferTo, TransferTo, RouteUID)
}

OnCallSessionTransferBlind(TransfererSessionId, TransfererChannelId,
                           Transferer, Transferee, TransferTo,
                           RouteUID, RouteType, RouteName)
{
    AcceptTransferBlind(TransfererSessionId, Transferee, Transferee,
    "0016148448545", "LineUSA", "")
}
```

See Also

AcceptTransferConsult(), RejectTransfer(), OnCallSessionTransferBlind(), GetVaxErrorCode()

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;

- Removes Transferee call from Transferer call-session.
- Disconnects Transferer calls from Transferer & TransferTo call-sessions.
- Closes Transferer call-session.
- Moves Transferee call at available/free channel of TransferTo call-session.

Syntax

```
boolean AcceptTransferConsult(
                                TransfererSessionId,
                                TransferToSessionId
                                )
```

Parameters

TransfererSessionId (integer)
: This parameter specifies a unique identification of a Transferer Call-Session.

TransferToSessionId (integer)
: This parameter specifies a unique identification of a TransferTo Call-Session.

Return Value

On successful execution this function returns non-zero value otherwise it returns 0. To get extended error information, call GetVaxErrorCode().

Example

```
OnCallSessionTransferConsult(TransfererSessionId, TransfererChannelId,
                             TransferToSessionId, TransferToChannelId,
                             Transferer, Transferee, TransferTo)
{

        AcceptTransferConsult(TransfererSessionId, TransferToSessionId)
}
```

See Also

AcceptTransferBlind(), RejectTransfer(), OnCallSessionTransferConsult(), GetVaxErrorCode()

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, TransfererChannelId,
                           Transferer, Transferee, TransferTo,
                           RouteUID, RouteType, RouteName)
{
    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, ChannelId, WaveId, Loop, PlayPos)
- PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
- UnLoadWaveID(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 GetVaxErrorCode() method.

Example

```
MusicWaveId = LoadWaveFile("C:\Music.wav")

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        PlayWaveStartToCallSession(SessionId, 0, MusicWaveId, 1, 0)
}
```

See Also

LoadWavePCM(), UnLoadWaveID(), PlayWaveStartToCallSession() , DialCallSession(), PlayWaveStopToCallSession(), GetVaxErrorCode()

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, ChannelId, WaveId, Loop, PlayPos)
- PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
- UnLoadWaveID(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 GetVaxErrorCode() method.

Example

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

        WaveId = LoadWavePCM(VoicePCM[], SizePCM)

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

OnCallSessionPlayWaveDone(SessionId, ChannelId, WaveId)
{
        UnLoadWaveID(WaveId)
}
```

See Also

LoadWaveFile(), UnLoadWaveID(), PlayWaveStartToCallSession() , DialCallSession(), PlayWaveStopToCallSession(), GetVaxErrorCode()

UnLoadWaveID()

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

Syntax

```
UnLoadWaveID(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) GetVaxErrorCode()

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        PlayWaveStartToCallSession(SessionId, 0, MusicWaveId, 1, 10)
}

OnCallSessionClosed(SessionId, ChannelId, ReasonCode)
{
        PlayWaveStopToCallSession(SessionId, 0, MusicWaveId)
}
```

See Also

LoadWavePCM(), LoadWaveFile(), PlayWaveStartToCallSession(), DialCallSession(), PlayWaveStopToCallSession()

PlayWaveStartToCallSession()

The PlayWaveStartToCallSession() function starts playing the wave file (.wav) data to the specific call in 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,
                                    ChannelId,
                                    WaveId,
                                    Loop,
                                    PlayPosPCM
                                    )
```

Parameters

SessionId (integer)
: This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

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.

PlayPosPCM (integer)
: This parameter value specifies the Position a wave to play. Either from start or Middle.

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 GetVaxErrorCode() method.

Example

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

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

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        PlayWaveStartToCallSession(SessionId, 0, GreetingWaveId, 1, 0)

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

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#22#")
        {
                PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
                PlayWaveStartToCallSession(SessionId, 0, HoldWaveId, 1, 0)
        }
}
```

See Also

LoadWaveFile(), UnLoadWaveID(), DialCallSession(), GetVaxErrorCode(), PlayWaveStopToCallSession(), DetectDigitDTMF()

PlayWaveStopToCallSession()

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

Syntax

```
boolean PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
```

Parameters

SessionId (integer)
This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

WaveId (integer)
This parameter value specifies the unique identification of wave data to 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 GetVaxErrorCode() method.

Example

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

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        PlayWaveStartToCallSession(SessionId, GreetingWaveId, 1)

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

```
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#22#")
        {
                PlayWaveStopToCallSession(SessionId, 0, WaveId)
        }
}
```

See Also

LoadWaveFile(), UnLoadWaveID(), DialCallSession(), GetVaxErrorCode(), PlayWaveStopToCallSession(), DetectDigitDTMF()

PlayWaveSetVolumeToCallSession()

The PlayWaveSetVolumeToCallSession() set volume of a playing wave data to a particular call in a call-session.

Syntax

```
boolean PlayWaveSetVolumeToCallSession (SessionId, ChannelId, WaveId, Volume)
```

Parameters

SessionId (integer)
: This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

WaveId (integer)
: This parameter value specifies the unique identification of wave data to be played.

Volume (integer)
: This parameter value specifies the volume.

  0 = Not Change
  100 = High Volume
  -100 = Low Volume

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 GetVaxErrorCode() method.

Example

```
WaveId = LoadWaveFile("C:\Greeting.wav")
if (WaveId = -1) GetVaxErrorCode()

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)

        PlayWaveStartToCallSession(SessionId, 0, WaveId, 1)
        PlayWaveSetVolumeToCallSession (SessionId, 0, WaveId, 100)

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


OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#22#")
        {
                PlayWaveSetVolumeToCallSession (SessionId, 0, WaveId, 0)
        }
        if(DigitDTMF = "#23#")
        {
                PlayWaveSetVolumeToCallSession (SessionId, 0, WaveId, -100)
        }

}
```

See Also

LoadWaveFile(), UnLoadWaveID(), DialCallSession(), GetVaxErrorCode(), PlayWaveStopToCallSession(), DetectDigitDTMF()

PlayWaveSetModeTypeToCallSession()

VaxTele allows to play multiple wave datas on a specific call of a call-session. PlayWaveSetModeTypeToCallSession() adjusts that how a wave data should be played either mix or excusively.

Syntax

```
boolean PlayWaveSetModeTypeToCallSession(
                                        SessionId,
                                        ChannelId,
                                        PlayModeType
                                        )
```

Parameters

SessionId (integer)
: This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

PlayModeType (integer)
: This parameter value specifies the ModeType of a playing wave.

  0 = Wave Mode Exclusive
  1 = Wave Mode Normal

  Exclusive mode allows to play an announcement to the agent in a callcenter scenario.

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 GetVaxErrorCode() method.

Example

```
WaveIdA = LoadWaveFile("C:\Greeting1.wav")
if (WaveIdA = -1) GetVaxErrorCode()

WaveIdB = LoadWaveFile("C:\Greeting2.wav")
if (WaveIdB = -1) GetVaxErrorCode()

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)

        PlayWaveStartToCallSession(SessionId, 0, WaveIdA, 1)
        PlayWaveSetModeTypeToCallSession(SessionId, 0, 0)
        PlayWaveStartToCallSession(SessionId, 0, WaveIdB, 1)

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

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#22#")
        {
                PlayWaveStartToCallSession(SessionId, 0, WaveId, 1)
                PlayWaveSetModeTypeToCallSession (SessionId, 0, 1)

        }
}
```

See Also

LoadWaveFile(), UnLoadWaveID(), DialCallSession(), GetVaxErrorCode(), PlayWaveStopToCallSession(), DetectDigitDTMF()

RecordWaveStartToCallSession()

The RecordWaveStartToCallSession() function starts recording the voice stream of a particular call in a 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
                                        ChannelId,
                                        FileName
                                        )
```

Parameters

SessionId (integer)
: This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

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 GetVaxErrorCode() method.

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId, 0, “Record.wav”)
}

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

See Also

RecordWaveStopToCallSession(), RecordWavePauseToCallSession(), DialCallSession(), GetVaxErrorCode()

RecordWaveStopToCallSession()

The RecordWaveStopToCallSession() function stops the recording of voice stream on a call of a call-session.

Syntax

```
boolean RecordWaveStopToCallSession(SessionId, ChannelId)
```

Parameters

SessionId (integer)
This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

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 GetVaxErrorCode() method.

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId,  0, "Record.wav")
}

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

See Also

RecordWaveStartToCallSession(), RecordWavePauseToCallSession(), DialCallSession(), GetVaxErrorCode()

RecordWavePauseToCallSession()

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

Syntax

```
boolean RecordWavePauseToCallSession(SessionId, ChannelId, Enable)
```

Parameters

SessionId (integer)
This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

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 GetVaxErrorCode() method.

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId , 0, "Record.wav")

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

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        If(Digit = "#220")
                RecordWavePauseToCallSession(SessionId, 0, 1)
        Else
                RecordWavePauseToCallSession(SessionId, 0, 0)
}
```

See Also

RecordWaveStartToCallSession(), RecordWaveStopToCallSession(), DialCallSession(), GetVaxErrorCode()

LoadMusicHold()

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

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 GetVaxErrorCode() method.

Example

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

See Also

UnLoadMusicHold(), GetVaxErrorCode()

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(), GetVaxErrorCode()

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 use of 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 GetVaxErrorCode() method.

Example

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

See Also

AcceptOffHoldRequest(), OnCallSessionOnHold(), GetVaxErrorCode()

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 use 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 GetVaxErrorCode() method.

Example

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

See Also

AcceptOnHoldRequest(), OnCallSessionOffHold(), GetVaxErrorCode()

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,
                              ToPeerName
                              )
```

Parameters

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

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 GetVaxErrorCode() method.

Example

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

See Also

RejectChatMesage(), GetVaxErrorCode()

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.
LIST OF SIP RESPONSES

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

AcceptChatMesage(), GetVaxErrorCode()

SendChatMessageText()

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

Syntax

```
integer SendChatMessageText(
                                MsgFrom,
                                MsgTo,
                                MsgText,
                                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.

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 GetVaxErrorCode() method.

Example

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

See Also

AcceptChatMesage(), GetVaxErrorCode()

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(
                                    SubscrbId,
                                    MsgFrom,
                                    MsgTo,
                                    ToPeerName
                                    )
```

Parameters

SubscrbId (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.

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 GetVaxErrorCode() method.

Example

```
OnChatstatusSubscribe(SubscrbId, MsgFrom, MsgTo, FromPeerType,
                              FromPeerName, FromIP, FromPort)

{
        AcceptChatStatusSubscribe(SubscrbId, "8021", "8095", "8095")
}
```

See Also

AcceptChatStatusSubscribe(), OnChatStatusSubscribe(), GetVaxErrorCode()

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(
                        SubscrbId,
                        MsgFrom,
                        MsgTo
                        )
```

Parameters

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

See Also

AcceptChatStatusSubscribe, OnChatStatusSubscribe, GetVaxErrorCode()

OpenConferenceRoom()

The OpenConferenceRoom() function creates a conference room to add multiple call-sessions. The added call-sessions 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 GetVaxErrorCode() method.

Example

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

See Also

CloseConferenceRoom(), GetVaxErrorCode()

CloseConferenceRoom()

The CloseConferenceRoom() function closes a 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(), GetVaxErrorCode()

AddCallSessionToConferenceRoom()

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

Syntax

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

Parameters

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

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 GetVaxErrorCode() method.

Example

```
AddCallSessionToConferenceRoom("RoomName", SessionId)
```

See Also

RemoveCallSessionFromConferenceRoom(), GetVaxErrorCode()

RemoveCallSessionFromConferenceRoom()

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

Syntax

```
RemoveCallSessionFromConferenceRoom(SessionId)
```

Parameters

SessionId (integer)
This parameter value specifies a unique identification of a call-session.

Return Value

No return value.

Example

```
RemoveCallSessionFromConferenceRoom()
```

See Also

AddCallSessionToConferenceRoom(), GetVaxErrorCode()

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,
                                        PlayPosPCM
                                        )
```

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.

PlayPosPCM (integer)
: This parameter value specifies the Position a wave to play. Either from start or Middle.

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 GetVaxErrorCode() method.

Example

```
PlayWaveStartToConferenceRoom("RoomName", WaveId, 1, 0)
```

See Also

PlayWaveStopToConferenceRoom(), GetVaxErrorCode()

PlayWaveStopToConferenceRoom()

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

Syntax

```
boolean PlayWaveStopToConferenceRoom(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.

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 GetVaxErrorCode() method.

Example

```
PayWaveStopToConferenceRoom("RoomName ", WaveId)
```

See Also

PlayWaveStartToConferenceRoom(), GetVaxErrorCode()

PlayWaveSetVolumeToConferenceRoom()

The PlayWaveSetVolumeToConferenceRoom() set volume of a playing specific wave data to a particular conference room.

Syntax

```
boolean PlayWaveSetVolumeToConferenceRoom(
                                                    RoomName,
                                                    WaveId,
                                                    Volume
                                                    )
```

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.

Volume (integer)
: This parameter value specifies the volume of a call.

  0 = Not Change
  100 = High Volume
  -100 = Low Volume

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 GetVaxErrorCode() method.

Example

```
PlayWaveSetVolumeToConferenceRoom("RoomName", WaveId, 100)
```

See Also

LoadWaveFile(), UnLoadWaveID(), PlayWaveStartToConferenceRoom(), GetVaxErrorCode(), PlayWaveStopToConferenceRoom(), DetectDigitDTMF()

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 GetVaxErrorCode() method.

Example

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

See Also

RecordWaveStopToConferenceRoom(), RecordWavePauseToConferenceRoom, GetVaxErrorCode()

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 GetVaxErrorCode() method.

Example

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

See Also

RecordWaveStartToConferenceRoom(), RecordWavePauseToConferenceRoom, GetVaxErrorCode()

RecordWavePauseToConferenceRoom()

The RecordWavePauseToConferenceRoom() function pauses 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 GetVaxErrorCode() method.

Example

```
RecordWavePauseToConferenceRoom( "RoomName", 1)
```

See Also

RecordWaveStartToConferenceRoom(), RecordWaveStopToConferenceRoom, GetVaxErrorCode()

AudioSessionLost()

The AudioSessionLost() 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 AudioSessionLost(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 GetVaxErrorCode() method.

Example

```
AudioSessionLost(1, 20)
```

See Also

OnCallSessionLost(), GetVaxErrorCode()

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 GetVaxErrorCode().

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 GetVaxErrorCode() method.

Example

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

See Also

OnOutgoingDiagnosticLOg(), OnIncomingDiagnosticLog(), GetVaxErrorCode()

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 timer tick 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.

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 GetVaxErrorCode() method.

Example

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

StartVaxTeleTick(2001, 8000)
```

See Also

StopVaxTeleTick(), OnVaxTeleTick(), GetVaxErrorCode()

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

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 GetVaxErrorCode() method.

Example

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

StartVaxTeleTick(2001, 8000)

StopVaxTeleTick(2001)
```

See Also

StartVaxTeleTick(), OnVaxTeleTick(), GetVaxErrorCode()

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 GetVaxErrorCode() method.

Example

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

See Also

GetUserAgentName(), GetVaxErrorCode()

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 SDP part 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 GetVaxErrorCode() method.

Example

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

See Also

GetSessionNameSDP(), GetVaxErrorCode()

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()

SendDigitDTMF()

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

Syntax

```
boolean SendDigitDTMF(
                        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 (RFC2833)
  1 = SIP based (INFO)
  2 = Inband (Audio Tone)

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 GetVaxErrorCode() method.

Example

```
SendDigitDTMF(SessionId, 0, 3)
```

See Also

DetectDigitDTMF(), OnDetectedDigitDTMF()

DetectDigitDTMF()

The DetectDigitDTMF() function enables/disables the detection of DTMF digit at Channel-ZERO or Channel-ONE call of a Call-Session.

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

For example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        DetectDigitDTMF(SessionId, 0, 0, 1, 2000)  // Enable RFC-2833

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

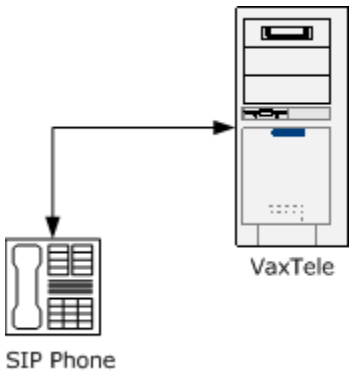
- Enables RTP based detection at Channel-ZERO call for two seconds time-out.
- VaxTele receives first digit. (e.g. 4)
- VaxTele waits for other digits with time-out interval.
- With two seconds interval, remote person press another digit (e.g. 8)
- VaxTele receives another digit. (e.g. 8)
- Waits for more digits within two seconds time-out interval.
- VaxTele receives no digit within two seconds time-out interval.
- VaxTele triggers **OnDetectedDigitDTMF(SessionId, 0, "48", 0)**

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

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

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,
                        ChannelId,
                        TypeDTMF,
                        Enable,
                        MilliSecTimeout
                        )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

TypeDTMF (integer)
The value of this parameter specifies mode of DTMF detection.

0 = RTP based (RFC2833)
1 = SIP based (INFO)
2 = Inband (Audio Tone)

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, RouteUID, RouteType, RouteName,
                  UserAgentName, FromIP, FromPort)
{
       AcceptCallSession(SessionId, CallerName, CallerId, DialNo,
       "", RouteUID, 20)

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

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                     FromPeerName, UserAgentName, FromIP, FromPort)
{
       AcceptCallSession(SessionId, "", "", "", "", "", 20)

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

See Also

OnDetectedDigitDTMF(), SendDigitDTMF(), GetVaxErrorCode()

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 GetVaxErrorCode() method.

Example

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

See Also

OnCallSessionDialToneStarted(), OnCallSessionDialToneEnded()

SplitCallSession()

The SplitCallSession() function removes a specfic call from a CallSession and moves that call into a new Call-Session.

The Call-Session is a collection of either one or two calls where the participating calls are identified as Channel-ZERO call and Channel-ONE call.

Please see **WHAT IS CALL-SESSION** for more details.

Syntax

```
integer SplitCallSession(SessionId, ChannelId)
```

Parameters

SessionId (integer)
This parameter value specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

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 GetVaxErrorCode() method.

Example

```
OnCallSessionTimeout(SessionIdA, 0)
{
        SessionIdB = SplitCallSession(SessionIdA, 0)

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

See Also

DialCallSession(), MoveCallSession()

MoveCallSession()

The MoveCallSession() function move the specfic call from source Call-Session to Destination Call-Session.

Please see **WHAT IS CALL-SESSION** for more details.

One of the typical scenario for use of MoveCallSession() 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 moves remote party call to agent's call-session.

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

- SessionIdA = DialCallSession(CallerName, CallerId, DialNo, ToPeerName, Timeout)
- SessionIdB = DialCallSession(CallerName, CallerId, DialNo, ToPeerName, Timeout)
- MoveCallSession(SessionIdA, ChannelId, SessionIdB)

Syntax

```
boolean MoveCallSession(
                        SrcSessionId,
                        SrcChannelId,
                        DscSessionId
                        )
```

Parameters

SrcSessionId (integer)
This parameter specifies a unique identification of source call-session.

SrcChannelId (integer)
This parameter value identifies a call in a source call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

DscSessionId
This parameter specifies a unique identification of destination 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 GetVaxErrorCode() method.

Example

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

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

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
        SessionIdA = SessionId;

        AcceptCallSession(SessionId, "", "", "", "", "", 20)
        PlayWaveStartToCallSession(SessionId, 0, GreetingWaveId, 1, 10)

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

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#22#")
        {
            SessionIdB = DialCallSession("82", "82", "8034", 3001, "8034",20)
        }

        MoveCall(SessionIdA, ChannelId, SessionIdB)
}
```

See Also

DialCallSession(), SplitCallSession()

VideoCOMM()

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

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 GetVaxErrorCode() method.

Example

```
Result = Initialize("sipsdk.com")
if(Result == 0) GetVaxErrorCode()

VideoCOMM(1)
```

See Also

Initialize(), GetVaxErrorCode()

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, ChannelId)
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE 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 GetVaxErrorCode().

Audio Codec Numbers:

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

Example

```
OnCallSessionConnected(SessionId)
{
	TxAudioCodec = GetCallSessionTxCodec(SessionId, 0)
}
```

See Also

GetCallSessionRxCodec(), GetVaxErrorCode()

GetCallSessionRxCodec()

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

Syntax

```
integer GetCallSessionRxCodec(SessionId, ChannelId)
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE 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 GetVaxErrorCode().

Audio Codec Numbers:

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

Example

```
OnCallSessionConnected(SessionId)
{
	RxAudioCodec = GetCallSessionRxCodec(SessionId, 1)
}
```

See Also

GetCallSessionTxCodec(), GetVaxErrorCode()

CallSessionMuteVoice()

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

Syntax

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

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Listen (boolean)
This parameter specifies to mutes the listening.

Speak (boolean)
This parameter specifies to mutes 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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", "", "", 20)

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

See Also

OnIncomingCall(), GetVaxErrorCode()

BusyLampSubscribeAccept()

The BusyLampSubscribeAccept() function accepts incoming BLF subscribe request. VaxTele receives BLF subscribe request and triggers OnBusyLampSubscribe() event, use 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 GetVaxErrorCode() 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

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 GetVaxErrorCode() method.

Example

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

See Also

BusyLampSubscribeAccept(), OnBusyLampSubscribe(), OnBusyLampSubscribeSuccess()

BusyLampSendStatus()

The BusyLampSendStatus() function forwards the BLF status from a specific user to a specific user. VaxTele receives BLF status from a user and triggers OnBusyLampSendStatus() event, use BusyLampSendStatus() to forward the BLF status.

Syntax

```
boolean BusyLampSendStatus(FromUserName, ToUserName, StateId)
```

Parameters

FromUserName (string)
: This parameter specifies sender's user name.

ToUserName (string)
: This parameter specifies receipient's user name.

StateId (integer)
: This parameter specifies the status-Id.

  0 = FREE
  1 = CONNECTING
  2 = CONNECTED
  3 = OFFLINE

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 GetVaxErrorCode() method.

Example

```
OnBusyLampSendStatus(FromUserName, ToUserName, StateId)
{
        BusyLampSendStatus(FromUserName, ToUserName, StateId)
}
```

See Also

OnBusyLampSendStatus()

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
  2 = CANCEL
  3 = BYE

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 GetVaxErrorCode() method.

Example

```
Result = Initialize("sipsdk.com")
if(Result = 0) GetVaxErrorCode()

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
2 = CANCEL
3 = BYE

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 GetVaxErrorCode() 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
2 = CANCEL
3 = BYE

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 GetVaxErrorCode() 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,
                                 ChannelId,
                                  Enable,
                                  AnalysisTime,
                                  SilenceTime,
                                  SilenceCount
                                  )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE 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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, RouteUID, RouteType, RouteName,
                    UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, “”, “”, “”, “”, “”, 30);
        CallSessionDetectAMD(SessionId, 0, 1, 6000, 300, 2)
}
```

See Also

OnCallSessionDetectAMD(), GetVaxErrorCode()

CallSessionSendStatusResponse()

The CallSessionSendStatusResponse() function is used to send SIP responses (Trying, Ringing, Not found etc.) to an incoming call in a call-session.

Syntax

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

Parameters

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

StatusCode (integer)
: This parameter specifies SIP response status.
LIST OF SIP RESPONSES

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    IncomingSessionId = SessionId
    CallSessionSendStatusResponse(SessionId, 180, "Ringing")
    AcceptCallSession(IncomingSessionId, "", "", "", "", "", 30)
}
```

See Also

StartVaxTeleTick(), AcceptCallSession()

GetCallSessionHeaderCallId()

The GetCallSessionHeaderCallId() returns the unique field/header CallId value for the specifc call. This is actually a SIP packet unique CallId.

Syntax

```
string GetCallSessionHeaderCallId (SessionId, ChannelId)
```

Parameters

SessionId (integer)
This parameter specifies the unique identification of a call.

ChannelId (integer)

This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Return Value

If the function succeeds, the return value is header call-id field otherwise, it returns empty string. To get extended error information, call GetVaxErrorCode().

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    HeaderId = GetCallSessionHeaderCallId(SessionId, ChannelId)
}
```

See Also

DialCallSession(), OnCallSessionCreated()

ConnectToServerREC()

The ConnectToServerREC() is connects a specific call identified by ChannelId to a SIP-REC protocol based call recording server.

Syntax

```
boolean ConnectToServerREC(
                                SessionId,
                                ChannelId,
                                CallerName,
                                CallerId,
                                DialNo,
                                ToPeerName,
                                Timeout
                                )
```

Parameters

SessionId (integer)
: This parameter specifies the unique identification of a call.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

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.

ToPeerName (string)
: This parameter specifies the name of To-Peer.

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 OnServerTimeoutREC() 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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, RouteUID, RouteType, RouteName,
                    UserAgentName, FromIP, FromPort)
{

        AcceptCallSession(SessionId, “”, “”, DialNo, DialNo, “”, 20)
        ConnectToServerREC()
}
```

See Also

RejectCallSession(), CloseCallSession(), DialCallSession(), GetVaxErrorCode(), OnIncomingCall()

SetExtDataREC()

The SetExtDataREC() set recording data into the SIP packet.

Syntax

```
boolean SetExtDataREC(SessionId, ChannelId, ExtData)
```

Parameters

SessionId (integer)
This parameter specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

ExtData (string)
This prameter value put data in ex data rec 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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{
    SetExDataRec (SessionId, ChannelId, RecData)
}
```

See Also

DialCallSession(), GetVaxErrorCode()

SendReqTransferBlind()

The SendReqTransferBlind() sends blind call transfer request of a specific call identified by ChannelId to the remote party/server.

Syntax

```
boolean SendReqTransferBlind(SessionId, ChannelId, ToUserName)
```

Parameters

SessionId (integer)
: This parameter specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

ToUserName (string)
: The parameter value specifies the transfer-to user 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 GetVaxErrorCode() method.

Example

```
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        SendReqTransferBlind(SessionId, ChannelId, "101")
}

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        SendReqTransferBlind(SessionId, ChannelId, DigitDTMF, TypeDTMF)
}
```

See Also

OnSendReqTransferCallTimeout(), OnSendReqTransferCallAccepted(), OnSendReqTransferCallFailed()

SendReqTransferCallConsult()

The SendReqTransferConsult() sends consult call transfer request of a specific call identified by ChannelId to the remote party/server. Remote third party server joins the both calls and starts voice streaming between those calls.

Syntax

```
boolean SendReqTransferCallConsult(
                                    SessionId,
                                    ChannelId,
                                    ToSessionId,
                                    ToChannelId,
                                    )
```

Parameters

SessionId (integer)
This parameter specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

ToSessionId (integer)
This parameter specifies transfer-to call-session.

ToChannelId (integer)
This parameter value identifies a transfer-to call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

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 GetVaxErrorCode() method.

Example

```
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        SendReqTransferConsult(SessionIdA, ChannelId, SessionIdB, 1)
}
```

See Also

OnSendReqTransferCallTimeout(), OnSendReqTransferCallAccepted(), OnSendReqTransferCallFailed()

AddRingGroup()

The AddRingGroup() function adds a RingGroup. This function along with other RingGroup functions develops RingGroup functionality.

RingGroup functionality flow:

- AddRingGroup("RingGroup")
- AddRingGroupAgent("RingGroup", "UserA")
- AddRingGroupAgent("RingGroup", "UserB")

Incoming call receives:

- OnIncomingCall(SessionId, CallerName, CallerId) event triggers.
- AddCallSessionToRingGroup(SessionId, CallerName, CallerId)
- Phones/extentions on both UserA & UserB side starts ringing.
- UserA or UserB pickup the call and conversation starts.

Syntax

```
boolean AddRingGroup(GroupName)
```

Parameters

GroupName (string)
The value of this parameter specifies group 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 GetVaxErrorCode() method.

Example

```
AddRingGroup("RG")

AddRingGroupAgent("RG", "UserA")
AddRingGroupAgent("RG", "UserB")

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{

    AddCallSessionToRingGroup(SessionId, CallerName, CallerId, "RG")

}
```

See Also

AddRingGroupAgent(), AddCallSessionToRingGroup(), RemoveRingGroup(), SetRingGroupProcessMode(), OnAddCallSessionToRingGroupSuccess(), GetVaxErrorCode()

RemoveRingGroup()

This function removes a RingGroup previously added by using AddRingGroup().

Syntax

```
RemoveRingGroup(GroupName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

Return Value

No return value

Example

```
RemoveRingGroup("RG")
```

See Also

AddRingGroup(), AddRingGroupAgent(), RemoveRingGroupAgent()

AddRingGroupAgent()

The AddRingGroupAgent() adds a user as RingGroup agent. User must be added previously by using AddUser() function.

Syntax

```
boolean AddRingGroupAgent(GroupName, UserName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

UserName (string)
The value of this parameter specifies the user 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 GetVaxErrorCode() method.

Example

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

Result = AddRingGroup("RG")
if(Result == 0) GetVaxErrorCode()

Result = AddRingGroupAgent("RG",  "9090")
if(Result == 0) GetVaxErrorCode()
```

See Also

AddRingGroup(), RemoveRingGroupAgent(), RemoveRingGroup()

RemoveRingGroupAgent()

This function removes an agent from a specific RingGroup.

Syntax

```
RemoveRingGroupAgent(GroupName, UserName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

UserName (string)
The value of this parameter specifies the user name.

Return Value

No Return Value.

Example

```
RemoveRingGroupAgent("RG", "9090")
```

See Also

AddRingGroup(), AddRingGroupAgent(), RemoveRingGroup()

SetRingGroupProcessMode()

This function is used to adjust RingGroup processing mode, it defines that how agents should be searched out for the processing of RingGroup calls.

Syntax

```
boolean SetRingGroupProcessMode(GroupName, ModeId, ModeType)
```

Parameters

GroupName (string)
: The value of this parameter specifies group name.

ModeId (integer)
: The value of this parameter specifies the RingGroup processing mode.

  0 = Ring All
  1 = Hunt Random
  2 = Round Robin
  3 = Least Answered
  4 = Least Talk Time
  5 = Longest Waiting
  6 = Prioritized Hunt

ModeType (integer)
: The value of this parameter specifies the RingGroup processing mode type. Either processing should be performed on only free users or all available users.

  0 = Free
  1 = Any

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 GetVaxErrorCode() method.

Example

```
SetRingGroupPrecessMode("RG", 0, 1)
```

See Also

AddRingGroupAgent(), RemoveRingGroupAgent(), RemoveRingGroup(), SetRingGroupAgentPriority()

SetRingGroupAgentPriority()

The SetRingGroupAgentPriority() adjusts the priority of an agent in a RingGroup.

Syntax

```
boolean SetRingGroupAgentPriority(
                                    GroupName,
                                    UserName,
                                    Priority
                                    )
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

UserName (string)
The value of this parameter specifies the user name.

Priority (integer)
The value of this parameter specifies the priority weight of an agent. Highest value represents highest priority.

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 GetVaxErrorCode() method.

Example

```
SetRingGroupAgentPriority("RG", "9090", 10)
SetRingGroupAgentPriority("RG", "9091", 20)  // at highest priority
SetRingGroupAgentPriority("RG", "9092", 13)
```

See Also

AddRingGroupAgent(), RemoveRingGroupAgent(), RemoveRingGroup(), SetRingGroupProcessMode()

AddCallSessionToRingGroup()

The AddCallSessionToRingGroup() adds a call-session to a RingGroup and internally generates outgoing calls to all the agents/users of that RingGroup and the phone extentions of those agents start ringing.

Syntax

```
boolean AddCallSessionToRingGroup(
                                    GroupName,
                                    CallerName,
                                    CallerId,
                                    SessionId
                                    )
```

Parameters

GroupName (integer)
: The value of this parameter specifies group name.

CallerName (string)
: This parameter specifies the caller name.

CallerId (string)
: This parameter specifies a unique identification of caller.

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, FromIP, FromPort)
{

    AddCallSessionToRingGroup(SessionId, CallerName, CallerId, "RG")

}
```

See Also

AddRingGroupAgent(), RemoveRingGroup(), SetRingGroupProcessMode()

AddCallPickUpGroup()

The AddCallPickUpGroup() function adds a call-pickup group. This function along with other pickup group functions can be used to develop call-pickup functionality.

Add members/users to a pickup group by using AddCallPickUpGroupMember() function.

If a member's phone set is ringing, other member of the same group can answer that call by picking up one's own phone set and then using the call pickup feature, instead of walking to the member’s desk.

Syntax

```
boolean AddCallPickUpGroup(GroupName)
```

Parameters

GroupName (string)
The value of this parameter specifies group 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 GetVaxErrorCode() method.

Example

```
AddCallPickUpGroup("CallPickup")

AddCallPickUpGroupMember("CallPickup", "2120")
AddCallPickUpGroupMember("CallPickup", "2121")

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, RouteUID, RouteType, RouteName,
                  UserAgentName, FromIP, FromPort)
{
        // Incoming call for 2121, DialNo value = "2121"
        // Ext-2121 starts ringing
        AcceptCallSession(SessionId, "", CallerId, DialNo, DialNo, "", 20)

        AddCallSessionToCallPickUpGroup("CallPickup", SessionId)
        // Add incoming call to pickup group
}

// Ext-2120 pickup the phone set and dials #22#2121

OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        PickupNo   = Remove first 4 digits of DigitDTMF
        PickupUser = Get last digits of DigitDTMF

        if(PickupNo = "#22#")
        {
                PickUpCall(SessionId, PickupUser)
        }
}
```

See Also

RemoveCallPickUpGroup(), AddCallPickUpGroupMember(), AddCallSessionToCallPickUpGroup(), PickUpCall()

RemoveCallPickUpGroup()

This function removes a call-pickup group.

Syntax

```
boolean RemoveCallPickUpGroup(GroupName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

Return Value

No Return Value.

Example

```
RemoveCallPickUpGroup("CallPickUp")
```

See Also

AddCallPickUpGroup(), AddCallPickUpGroupMember(), AddCallSessionToCallPickUpGroup(), PickUpCall()

AddCallPickUpGroupMember()

The AddCallPickUpGroupMember() function add a user as call-pickup group member to a call-pickup group. User must be added previously by using AddUser() function.

Syntax

```
boolean AddCallPickUpGroupMember(GroupName, UserName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

UserName (string)
The value of this parameter specifies the user 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 GetVaxErrorCode() method.

Example

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

Result = AddCallPickUpGroup("CallPickUp")
if(Result == 0) GetVaxErrorCode()

Result = AddCallPickUpGroupMember("CallPickUp",  "9090")
if(Result == 0) GetVaxErrorCode()
```

See Also

AddCallPickUpGroup(), RemoveCallPickUpGroupMember(), AddCallSessionToCallPickUpGroup(), PickUpCall()

RemoveCallPickUpGroupMember()

The RemoveCallPickUpGroupMember() removes a user from a call-pickup group.

Syntax

```
RemoveCallPickUpGroupMember(GroupName,  UserName)
```

Parameters

GroupName (string)
The value of this parameter specifies group name.

UserName (string)
The value of this parameter specifies the user name.

Return Value

No Return Value.

Example

```
RemoveCallPickUpGroupMember("CallPickUp", "9090")
```

See Also

AddCallPickUpGroup(), RemoveCallPickUpGroupMember(), AddCallSessionToCallPickUpGroup(), PickUpCall()

PickUpCall()

The PickUpCall() function connects a call-session to the member of a call-pickup group.

Syntax

```
boolean PickUpCall(SessionId, UserName)
```

Parameters

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

UserName (string)
The value of this parameter specifies the user 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 GetVaxErrorCode() method.

Example

```
// Ext-2120 pickup the phone set and dials #22#2121
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        PickupNo   = Remove first 4 digits of DigitDTMF
        PickupUser = Get last digits of DigitDTMF

        if(PickupNo = "#22#")
        {
                PickUpCall(SessionId, PickupUser)
        }
}
```

See Also

AddCallPickUpGroup(), AddCallPickUpGroupMember(), RemoveCallPickUpGroupMember()

ParkCallSession()

The ParkCallSession() function parks a call-session at a provided unique slot number. It can be used with other call-paking functions to develop a call-parking functionality.

Syntax

```
boolean ParkCallSession(SessionId, ChannelId, SlotId)
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

SlotId (integer)
: The value of this parameter specifies unique slot 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 GetVaxErrorCode() method.

Example

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererChannelId,
                           Transferer, Transferee, TransferTo,
                           RouteUID, RouteType, RouteName)
{
        ParkCallSession(TransfererSessionId, TransfererChannelId, 101)
        //Park call for a person
}

// Person pickup a phone extention and dials #22#101
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        Code  = Remove first 4 digits of DigitDTMF
        SlotId = Get last digits of DigitDTMF

        if(Code  = "#22#")
        {
                ConnectToParkedCallSession(SessionId, ChannelId, SlotId)
        }
}
```

See Also

ConnectToParkedCallSession()

ConnectToParkedCallSession()

The ConnectToParkedCallSession() function connects a call of a call-session to the call previously parked at slotId by using ParkCallSession() function.

Syntax

```
boolean ConnectToParkedCallSession(SessionId, ChannelId, SlotId)
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

SlotId (integer)
The value of this parameter specifies unique slot 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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, UserAgentName, FromIP, FromPort)
{

        Code  = Remove first 4 digits of DialNo
        SlotId = Get last digits of DialNo

        if(Code  = "#22#")
        {
                ConnectToParkedCallSession(SessionId, ChannelId, SlotId)
        }

}
```

See Also

ParkCallSession()

AddQueue()

The AddQueue() function adds a queue. With the use of other queue functions call-queue functionality can be developed.

Add queue agents to a call-queue by using AddQueueAgent() function.

Add incoming call to call-queue by using AddCallSessionToQueue() function to be answered by the available queue agent.

Syntax

```
boolean AddQueue(QueueName)
```

Parameters

QueueName (string)
The value of this parameter specifies queue 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 GetVaxErrorCode() method.

Example

```
Result = AddQueue("Queue")
if(Result == 0) GetVaxErrorCode()
```

See Also

RemoveQueue(), AddQueueAgent(), RemoveQueueAgent(), SetQueueAgentPriority(), SetQueueProcessMode(), AddCallSessionToQueue()

RemoveQueue()

This function removes a call-queue previously added by using AddQueue().

Syntax

```
RemoveQueue(QueueName)
```

Parameters

QueueName (string)
The value of this parameter specifies queue name.

Return Value

No Return Value.

Example

```
RemoveQueue ("Queue")
```

See Also

AddQueue(), AddQueueAgent(), RemoveQueueAgent(), SetQueueAgentPriority(), SetQueueProcessMode(), AddCallSessionToQueue()

AddQueueAgent()

The AddQueueAgent() adds a user as queue agent to the call-queue. User must be added previously by using AddUser() function.

Syntax

```
boolean AddQueueAgent(QueueName, UserName)
```

Parameters

QueueName (string)
The value of this parameter specifies queue name.

UserName (string)
The value of this parameter specifies the user 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 GetVaxErrorCode() method.

Example

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

Result = AddQueue("Queue")
if(Result == 0) GetVaxErrorCode()

Result = AddQueueAgent("Queue",  "9090")
if(Result == 0) GetVaxErrorCode()
```

See Also

RemoveQueue(), RemoveQueueAgent(), SetQueueAgentPriority(), SetQueueProcessMode(), AddCallSessionToQueue()

RemoveQueueAgent()

This function removes an agent from a call-queue.

Syntax

```
RemoveQueueAgent(QueueName, UserName)
```

Parameters

QueueName (string)
The value of this parameter specifies queue name.

UserName (string)
The value of this parameter specifies the user name.

Return Value

No Return Value.

Example

```
RemoveQueueAgent ("Queue", "9090")
```

See Also

RemoveQueue(), AddQueueAgent(), SetQueueAgentPriority(), SetQueueProcessMode(), AddCallSessionToQueue()

SetQueueAgentPriority()

The SetQueueAgentPriority() adjusts the priority of an agent in a call-queue.

Syntax

```
boolean SetQueueAgentPriority(QueueName, UserName, Priority)
```

Parameters

QueueName (string)
: The value of this parameter specifies queue name.

UserName (string)
: The value of this parameter specifies the user name.

Priority (integer)
: The value of this parameter specifies the priority weight of an agent. Highest value represents highest priority.

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 GetVaxErrorCode() method.

Example

```
SetQueueAgentPriority("Queue", "9090", 10)
SetQueueAgentPriority("Queue", "9091", 14) // at highest priority
SetQueueAgentPriority("Queue", "9092", 13)
```

See Also

RemoveQueue(), RemoveQueueAgent(), AddCallSessionToQueue(), SetQueueProcessMode(), AddQueue()

SetQueueProcessMode()

The function is used to adjust call-queue processing mode, it defines that how agents should be searched out for the processing of call-queue calls.

Syntax

```
boolean SetQueueProcessMode(QueueName, ModeId, ProcessModeType)
```

Parameters

QueueName (string)
: The value of this parameter specifies queue name.

ModeId (integer)
: The value of this parameter specifies the call-queue processing mode.

  0 = Ring All
  1 = Hunt Random
  2 = Round Robin
  3 = Least Answered
  4 = Least Talk Time
  5 = Longest Waiting
  6 = Prioritized Hunt

ModeType (integer)
: The value of this parameter specifies the call-queue processing mode type. Either processing should be performed on only free users or all available users.

  0 = Free
  1 = Any

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 GetVaxErrorCode() method.

Example

```
SetQueueProcessMode (“Queue”, 5, 0)
```

See Also

RemoveQueue(), RemoveQueueAgent(), AddCallSessionToQueue(), SetQueueProcessMode(), AddQueue()

AddCallSessionToQueue()

The AddCallSessionToQueue() adds a call-session to a call-queue.

Syntax

```
boolean AddCallSessionToQueue(
                                    QueueName,
                                    CallerName,
                                    CallerId,
                                    SessionId
                                    )
```

Parameters

QueueName (string)
: The value of this parameter specifies queue name.

CallerName (string)
: This parameter specifies the caller name.

CallerId (string)
: This parameter specifies a unique identification of caller.

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, RouteUID, RouteType, RouteName,
                    UserAgentName, FromIP, FromPort)
{
        AddCallSessionToQueue("Queue", CallerName, CallerId, SessionId)
        PlayWaveStartToCallSession(SessionId, 0, GreetingWaveId, 1, 0)
}
```

See Also

RemoveQueue(), RemoveQueueAgent(), SetQueueProcessMode(), SetQueueProcessMode(), AddQueue()

AddStealthListener()

The AddStealthListener() provides functionality to develop call barge-in feature.

Call barge-in feature allows a person to listen the conversation of another call. Such feature is very useful in call centers for quality assurance and training purposes.

Syntax

```
boolean AddStealthListener(StealthSessionId, CapturePeerName)
```

Parameters

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

CapturePeerName (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 GetVaxErrorCode() method.

Example

```
AddUser("0000", "123", "01")   // Set user 0000 in your database as admin
AddUser("9090", "123", "01")

OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                  FromPeerName, RouteUID, RouteType, RouteName,
                  UserAgentName, FromIP, FromPort)
{
        If(FromPeerType = 0 AND FromPeerName = "0000")
        {
                AcceptCallSession(SessionId, "", "", "", "", "", 20)
                AddStealthListener(SessionId, "0000")
        }
        Else
        {
                AcceptCallSession(SessionId, "", "", DialNo, DialNo, "", 20)
                AttachToStealthListener(SessionId)
        }
}
```

See Also

AcceptCallSession(), AdjustCallSessionVoiceType()

AdjustCallSessionVoiceType()

The AdjustCallSessionVoiceType() adjusts the voice listen type of a call in a call-session, conference room or in steath-listening functionality.

AdjustCallSessionVoiceType() and AddStealthListener() can also be used to develop agent training functionality in a call center.

There are three voice listen types:

TYPE_VOICE ADMIN
: The admin-type participant can listen user-type, normal-type and admin-type participants.

TYPE_VOICE USER
: The user-type participant can listen admin-type, normal-type and user-type participants.

TYPE_VOICE NORMAL
: The normal-type participant can listen user-type and normal-type participants.

Syntax

```
boolean AdjustCallSessionVoiceType(SessionId, ChannelId, VoiceTypeId)
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

VoiceTypeId (integer)
: This parameter value specifies the voice type of the participant.

  0 = Normal Voice Type
  1 = User Voice Type
  2 = Admin Voice Type

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 GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, UserAgentName, FromIP, FromPort)
{

        AcceptCallSession(SessionId, “”, “”, “”, “”, “”)
        AddStealthListener(SessionId, “4040”, 2)

        // Retrieve AgentSessionId from the database or list.
        // Adjust the voice type of agent to voice type user.

        AdjustCallSessionVoiceType(AgentSessionId, 1, 1)
}
```

See Also

AddStealthListener(), AddCallSessionToConferenceRoom()

EXPORTED EVENTS

OnVaxErrorLog()

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

Please see **LIST OF ERROR CODES** 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")

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

OnVaxErrorLog(FuncName, ErrorCode, ErrorMsg)
{

}
```

See Also

GetVaxErrorCode()

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 **HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP)** for more details.

Please see **HOW TO CONNECT TO PSTN/GSM NETWORK** 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 registration of a LINE (SIP account settings) to external SIP server or IP-Telephony service provider fails.

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**

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 successfully completes.

Please see **HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP)** for more details.

Please see **HOW TO CONNECT TO PSTN/GSM NETWORK** 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 if a provided LINE fails to un-register 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**

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 unregisters 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 executes 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.

Please see **SIP CLIENT REGISTRATION FLOW** for more details.

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** for more details.

Syntax

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

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.

UserAgentName (string)
: This parameter specifies the UserAgentName of SIP client.

FromIP (string)
: This parameter value specifies the from IP address.

FromPort (integer)
: This parameter specifies the from port number.

RegId (integer)
: This parameter specifies a unique identification of a registration session.

Example

```
OnRegisterUser(UserLogin, Domain, UserAgentName, FromIP, FromPort,
                RegId)
{
        AddUser(UserLogin, "123", "01")
        AcceptRegister(RegId)
}
```

See Also

OnUnRegisterUser(), AddUser(), RemoveUser()

OnRegisterUserSuccess()

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

Please see **SIP CLIENT REGISTRATION FLOW** for more details.

Syntax

```
OnRegisterUserSuccess(UserLogin)
```

Parameters

UserLogin (string)
: This parameter specifies the user's login of SIP client.

FromIP (string)
: This parameter value specifies the from IP address.

FromPort (integer)
: This parameter specifies the from port number.

RegId (integer)
: This parameter specifies a unique identification of a registration session.

Example

```
OnRegisterUserSuccess(UserLogin, FromIP, FromPort, RegId)
{
}
```

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.

FromIP (string)
: This parameter value specifies the from IP address.

FromPort (integer)
: This parameter specifies the from port number.

RegId (integer)
: This parameter specifies a unique identification of a registration session.

Example

```
OnRegisterUserFailed(UserLogin, FromIP, FromPort, RegId)
{
	RemoveUser(UserLogin)
}
```

See Also

OnRegisterUser(), OnRegisterUserSuccess(), RemoveUser()

OnCallSessionCreated()

The OnCallSessionCreated() event triggers when VaxTele creates/allocates a call-session internally.

Syntax

```
OnCallSessionCreated(
                        SessionId,
                        ReasonCode
                        )
```

Parameters

SessionId (integer)
: This parameter specifies the unique identification of 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)
{
        // Allocate resources for the Session
}
```

See Also

OnCallSessionClosed()

OnCallSessionClosed()

The OnCallSessionClosed() event triggers when VaxTele closes a call-session internally.

Syntax

```
OnCallSessionClosed(SessionId, ChannelId, ReasonCode)
```

Parameters

SessionId (integer)
This parameter specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

ReasonCode (integer)
This parameter specifies the reason due to which the call-session is closed.

- HANGUP 0
- SESSION_LOST 1
- MERGED 2
- TRANSFERED 3
- REJECTED 4
- FAILED 5
- CANCELLED 6
- TIMEOUT 7
- CLOSED 8

Example

```
OnCallSessionClosed(SessionId, ChannelId, ReasonCode)
{

}
```

See Also

OnCallSessionCreated()

OnIncomingCall()

The OnIncomingCall() event triggers when VaxTele receives a call request.

Syntax

```
OnIncomingCall(
        SessionId, CallerName, CallerId,
        DialNo, FromPeerType, FromPeerName,
        RouteUID, RouteType, RouteName,
        UserAgentName,  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.

  0 = User PeerType
  1 = Line PeerType

FromPeerName (string)
: This parameter value specifies the name of From-Peer.

RouteUID (string)
: This parameter specifies the name/unique RouteId assigned to Route-Types Queue, RingGroup, Line, User, StealthListen, Room etc. For further details, please have a look at AddQueueRouteUID(), AddUserRouteUID(), AddLineRouteUID(), AddStealthListenRouteUID() methods.

RouteType (integer)
: This parameter specifies the Route-Types (Queue, RingGroup, Line, User, StealthListen, Room etc).

RouteName (string)
: This parameter value specifies the Route-Name (Queue-Name, Line-Name, RingGroup-Name, Room-Name etc)

UserAgentName (string)
This parameter value specifies the name of UserAgent.

FromIP (string)
This parameter value specifies the From IP address.

FromPort (integer)
This parameter specifies the From port number.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                FromPeerName, RouteUID, RouteType, RouteName,
                UserAgentName, 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,
                        ChannelId,
                        StatusCode,
                        ReasonPhrase
                        )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

StatusCode (integer)
: This parameter specifies SIP response status code (100, 181 etc).
  LIST OF SIP RESPONSES

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

Example

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

See Also

AcceptCallSession(), OnCallSessionConnected, OnIncomingCall()

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,
                    ChannelId,
                    StatusCode,
                    ReasonPhrase
                    )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

StatusCode (integer)
This parameter specifies SIP response status code (486, 404 etc).
LIST OF SIP RESPONSES

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

Example

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

See Also

AcceptCallSession(), SplitCallSession(), OnCallSessionConnected, OnIncomingCall()

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

SplitCallSession(), AcceptCallSession(), OnIncomingCall()

OnCallSessionLost()

The OnCallSessionLost() event triggers when VaxTele does not receive voice data for define interval of time.

Syntax

```
OnCallSessionLost(
                    SessionId,
                    ChannelId
                    )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

Example

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

See Also

AudioSessionLost()

OnCallSessionHangup()

The OnCallSessionHangup() event triggers when remote party hangup the call.

Syntax

```
OnCallSessionHangup(
                    SessionId,
                    ChannelId
                    )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

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

See Also

AcceptCallSession(), SplitCallSession(), OnCallSessionConnected(), OnIncomingCall()

OnCallSessionTimeout()

The OnCallSessionTimeout() event triggers when VaxTele fails to establish a Call-Session and does not receive the response from SIP client within time period specified in AcceptCallSession() / DialCallSession() method.

Syntax

```
OnCallSessionTimeout(
                    SessionId,
                    ChannelId
                    )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

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

See Also

AcceptCallSession(), DialCallSession()

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 of a call-session.

Syntax

```
OnCallSessionOnHold(
                    SessionId,
                    ChannelId
                    )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

Example

```
OnCallSessionOnHold(SessionId, ChannelId)
{
}
```

See Also

AcceptOnHoldRequest(), OnCallSessionOffHold()

OnCallSessionOffHold()

The OnCallSessionOffHold() event triggers, when VaxTele receives call off-hold request from SIP client for specific call of a call-session.

Syntax

```
OnCallSessionOffHold(
                        SessionId,
                        ChannelId
                        )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

Example

```
OnCallSessionOffHold(SessionId, ChannelId)
{

}
```

See Also

AcceptOffHoldRequest(), OnCallSessionOnHold()

OnCallSessionTransferBlind()

The OnCallSessionTransferBlind() event triggers when VaxTele receives blind call transfer request from a SIP client.

Syntax

```
OnCallSessionTransferBlind(
                    TransfererSessionId, TransfererChannelId,
                    Transferer, Transferee, TransferTo,
                    RouteUID, RouteType, RouteName
                    )
```

Parameters

TransfererSessionId (integer)
: This parameter specifies the call-session identification of transferer.

TransfererChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE 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.

RouteUID (string)
: This parameter specifies the name/unique RouteId assigned to Route-Types Queue, RingGroup, Line, User, StealthListen, Room etc. For further details, please have a look at AddQueueRouteUID(), AddUserRouteUID(), AddLineRouteUID(), AddStealthListenRouteUID() methods.

RouteType (integer)
: This parameter specifies the Route-Types (Queue, RingGroup, Line, User, StealthListen, Room etc).

RouteName (string)
: This parameter value specifies the Route-Name (Queue-Name, Line-Name, RingGroup-Name, Room-Name etc)

Example

```
OnCallSessionTransferBlind(TransfererSessionId, TransfererChannelId,
                           Transferer, Transferee, TransferTo,
                           RouteUID, RouteType, RouteName)
{

}
```

See Also

AcceptTransferBlind(), OnCallSessionTransferConsult(), AcceptTransferConsult()

OnCallSessionTransferConsult()

The OnCallSessionTransferConsult() event triggers when VaxTele receives consult call transfer request from a SIP client.

Syntax

```
OnCallSessionTransferConsult(
                                TransfererSessionId,
                                TransfererChannelId,
                                TransferToSessionId,
                                TransferToChannelId,
                                Transferer,
                                Transferee,
                                TransferTo
                                )
```

Parameters

TransfererSessionId (integer)
: This parameter specifies the call-session identification of transferer.

TransfererChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

TransferToSessionId (integer)
: This parameter specifies the session identification of *transfer To*.

TransferToChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE 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, TransfererChannelId,
                              TransferToSessionId, TransferToChannelId,
                              Transferer, Transferee, TransferTo)
{

}
```

See Also

AcceptTransferBlind(), AcceptTransferConsult(), OnCallSessionTransferBlind()

OnCallSessionTransfering()

The OnCallSessionTransfering() event triggers when a call transfer process initiates and VaxTele starts receiving SIP status responses regarding transfering of a call.

Syntax

```
OnCallSessionTransfering(
                        SessionId,
                        TransferType,
                        ChannelId,
                        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.

0 = Transfer-Blind
1 = Transfer-Consult

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

StatusCode (integer)
This parameter specifies SIP response status code (100, 181 etc).
LIST OF SIP RESPONSES

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

Example

```
OnCallSessionTransfering(SessionId, TransferType, ChannelId, StatusCode,
                         ReasonPhrase)
{
}
```

See Also

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(), OnCallSessionTransfered(), RejectTransfer()

OnCallSessionTransferFailed()

The OnCallSessionTransferFailed() event triggers when VaxTele recieves failure responses during call transfer process and failed to transfer the call.

Syntax

```
OnCallSessionTransferFailed(
                              SessionId,
                              TransferType,
                              ChannelId,
                              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.

0 = Transfer-Blind
1 = Transfer-Consult

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

StatusCode (integer)
This parameter specifies SIP response status code (486, 404 etc).
LIST OF SIP RESPONSES

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

Example

```
OnCallSessionTransferFailed(SessionId,TransferType, ChannelId, StatusCode,
                              ReasonPhrase)
{
}
```

See Also

OnCallSessionTransferBlind(), OnCallSessionTransferConsult(), RejectTransfer()

OnCallSessionTransferTimeout()

The OnCallSessionTransferTimeout() event triggers when call transfer process fails and VaxTele does not receive the response from SIP client (Transfer-To) within specified time limit.

Syntax

```
OnCallSessionTransferTimeout(
                                SessionId,
                                TransferType,
                                ChannelId
                                )
```

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.

0 = Transfer-Blind
1 = Transfer-Consult

ChannelId (integer)
: This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

```
OnCallSessionTransferTimeout(SessionId, TransferType, ChannelId)
{
}
```

See Also

OnCallSessionTransferBlind(), OnCallSessionTransferConsult(), RejectTransfer()

OnCallSessionTransfered()

The OnCallSessionTransfered() event triggers when VaxTele successfully transfers a call.

Syntax

```
OnCallSessionTransfered(
                        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.

0 = Transfer-Blind
1 = Transfer-Consult

Example

```
OnCallSessionTransfered(SessionId, TransferType)
{
}
```

See Also

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(), OnCallSessionTransfering()

OnSendReqTransferCallTimeout()

The OnSendReqTransferCallTimeout() event triggers when VaxTele trying to transfer a call but failed in the specfied time.

Syntax

```
OnSendReqTransferCallTimeout(
                                SessionId,
                                ChannelId
                                )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

```
OnSendReqTransferCallTimeout(SessionId, ChannelId)
{
}
```

See Also

SendReqTransferCallBlind(), SendReqTransferCallConsult()

OnSendReqTransferCallAccepted()

The OnSendReqTransferCallAccepted() event triggers and notifies that the send call transfer request is accepted.

Syntax

```
OnSendReqTransferCallAccepted(
                                SessionId,
                                ChannelId
                                )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

```
OnSendReqTransferCallAccepted(SessionId, ChannelId)
{
}
```

See Also

SendReqTransferCallBlind(), SendReqTransferCallConsult()

OnSendReqTransferCallFailed()

The OnSendReqTransferCallFailed() event triggers when transfer call request is not accepted and VaxTele receives an error response.

Syntax

```
OnSendReqTransferCallFailed(
                                SessionId,
                                ChannelId,
                                StatusCode,
                                ReasonPhrase
                                )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

StatusCode (integer)
This parameter specifies SIP response status.
LIST OF SIP RESPONSES

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

Example

```
OnSendReqTransferCallFailed(SessionId, ChannelId, StatusCode,
                                ReasonPhrase)
{
}
```

See Also

SendReqTransferCallBlind(), SendReqTransferCallConsult()

OnDetectedDigitDTMF()

The OnDetectedDigitDTMF() event triggers when VaxTele notifies about the DTMF digit receives from Channel-ZERO call or Channel-ONE call in a Call-Session.

Syntax

```
OnDetectedDigitDTMF(
                        SessionId,
                        ChannelId,
                        DigitDTMF,
                        TypeDTMF
                        )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

DigitDTMF (string)
: This parameter value specifies any digit(s) that has been pressed.

TypeDTMF (integer)
: The value of this parameter specifies mode of DTMF detection.

  0 = RTP based (RFC2833)
  1 = SIP based (INFO)
  2 = Inband (Audio Tone)

Example

```
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
        if(DigitDTMF = "#") // if user press #
        PlayWaveStartToCallSession()
}
```

See Also

DetectDigitDTMF()

OnOutgoingDiagnosticLog()

The OnOutgoingDiagnosticLog() event triggers when VaxTele sends a SIP packet. This event can be used 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 used 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.

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

SendInfoVM(), OnSendSuccessVM()

OnSendSuccessVM()

The OnSendSuccessVM() event triggers when VaxTele successfully sends voice mail related information to SIP based softphones/hardphones.

Syntax

```
OnSendSuccessVM(MsgIdVM)
```

Parameters

MsgIdVM (integer)
: This parameter specifies a unique identification of voice mail message.

Example

```
OnSendSuccessVM(MsgIdVM)
{

}
```

See Also

SendInfoVM(), OnSendSuccessVM()

OnCallSessionDialToneStarted()

The OnCallSessionDialToneStarted() event triggers when VaxTele starts playing the dial tone.

Syntax

```
OnCallSessionDialToneStarted(SessionId, ChannelId, WaveId)
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

WaveId (integer)
: This parameter value specifies the unique identification of wave data to be played.

Example

```
OnCallSessionDialToneStarted(SessionId, ChannelId, WaveId)
{

}
```

See Also

DialToneToCallSession(), OnCallSessionDialToneEnded()

OnCallSessionDialToneEnded()

The OnCallSessionDiaToneStop() event triggers when VaxTele stops playing the dial tone.

Syntax

```
OnCallSessionDialToneEnded(SessionId, ChannelId, WaveId)
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

WaveId (integer)
: This parameter value specifies the unique identification of wave data to be played.

Example

```
OnCallSessionDialToneEnded(SessionId, ChannelId, WaveId)
{

}
```

See Also

DialToneToCallSession(), OnCallSessionDialToneStarted()

OnCallSessionPlayWaveDone()

The OnCallSessionPlayWaveDone() event triggers when a wave file played successfully for a particular call of a call-session.

Syntax

```
OnCallSessionPlayWaveDone(
                              SessionId,
                              ChannelId,
                              WaveId
                              )
```

Parameters

SessionId (integer)
: This parameter specifies a unique identification of a call-session.

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

WaveId (integer)
: This parameter value specifies the unique identification of wave data to be played.

Example

```
OnCallSessionPlayWaveDone(SessionId, ChannelId, 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()

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,
                        ChannelId,
                        IsHuman
                        )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE 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, ChannelId, IsHuman)
{
}
```

See Also

CallSessionDetectAMD()

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.

  0 = User PeerType
  1 = Line PeerType

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

AcceptChatStatusSubscribe(), RejectChatStatusSubscribe()

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.

  0 = User PeerType
  1 = Line PeerType

FromPeerName (string)
: This parameter value specifies the name of From-Peer.

FromIP (string)
: This parameter value specifies the F*rom IP* address.

FromPort (integer)
: This parameter specifies the F*rom port* number.

Example

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

See Also

AcceptChatStatusSubscribe(), RejectChatStatusSubscribe()

OnChatStatusSubscribe()

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

Syntax

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

Parameters

SubscrbId (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.

0 = User PeerType
1 = Line PeerType

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(SubscrbId, 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**

ReasonPhrase (string)
: This parameter specifies SIP response reason phrase (Request Timeout, Fprbidden 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()

OnBusyLampSubscribe()

The OnBusyLampSubscribe() event triggers when VaxTele receives BLF subscribe request.

Syntax

```
OnBusyLampSubscribe(
                        SubscrbId,
                        UserName,
                        ToUserName,
                        FromIP,
                        FromPort
                       )
```

Parameters

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

See Also

OnBusyLampUnSubscribe(), BusyLampSubscribeAccept(), BusyLampSubscribeReject(), OnBusyLampSubscribeSuccess()

OnBusyLampUnSubscribe()

The OnBusyLampUnSubscribe() event triggers when VaxTele receives BLF unsubscribe request.

Syntax

```
OnBusyLampUnSubscribe(
                        SubscrbId,
                        UserName,
                        ToUserName,
                        FromIP,
                        FromPort
                       )
```

Parameters

SubscrbId (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(SubScrbId, UserName, ToUserName, FromIP,
                        FromPort)
{
}
```

See Also

OnBusyLampSubscribe()

OnBusyLampSubscribeSuccess()

The OnBusyLampSubscribeSuccess() event triggers when the BLF subscribe request completes successfully.

Syntax

```
OnBusyLampSubscribeSuccess(
                                SubscrbId,
                                UserName
                                )
```

Parameters

SubscrbId (integer)
: This parameter value specifies a unique subscribe identification.

UserName (string)
: This parameter value specifies BLF subscribe user.

Example

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

 OnBusyLampSubscribeSuccess(SubScrbId, UserName)
{

}
```

See Also

OnBusyLampSubscribe(), OnBusyLampSubscribeFailed()

OnBusyLampSubscribeFailed()

The OnBusyLampSubscribeFailed() event triggers when the BLF subscribe request fails to complete.

Syntax

```
OnBusyLampSubscribeFailed(
                              SubscrbId,
                              UserName
                              )
```

Parameters

SubscrbId (integer)
: This parameter value specifies a unique subscribe identification.

UserName (string)
: This parameter value specifies BLF subscribe user.

Example

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

OnBusyLampSubscribeFailed(SubScrbId, UserName)
{

}
```

See Also

OnBusyLampSubscribeSuccess(), OnBusyLampSubscribe(), OnBusyLampUnSubscribe()

OnBusyLampSendStatus()

The OnBusyLampSendStatus() event triggers when the BLF status notification receives.

Syntax

```
OnBusyLampSendStatus(FromUserName, ToUserName, StateId)
```

Parameters

FromUserName (string)
This parameter specifies sender's user name.

ToUserName (string)
This parameter specifies receipient's user name.

StateId (integer)
This parameter specifies the status-Id.

0 = FREE
1 = CONNECTING
2 = CONNECTED
3 = OFFLINE

Example

```
 OnBusyLampSendStatus(FromUserName, ToUserName, StateId)
{
        BusyLampSendStatus(FromUserName, ToUserName, StateId)
}
```

See Also

BusyLampSendStatus()

OnCallSessionRecordedPCM()

The OnCallSessionRecordedPCM() event starts triggering when recording on a specific call-session starts by using RecordWaveStartToCallSession() with value empty string to the second parameter.

e.g. RecordWaveStartToCallSession(SessionId, “”)

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()

OnAddCallSessionToQueueSuccess()

The OnAddCallSessionToQueueSuccess() event trigers when VaxTele adds a call-session in a queue successfully.

Syntax

```
OnAddCallSessionToQueueSuccess(QueueName, SessionId)
```

Parameters

QueueName (string)
: The value of this parameter specifies queue name.

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

Example

```
OnAddCallSessionToQueueSuccess(QueueName, SessionId)
{

}
```

See Also

AddCallSessionToQueue()

OnAddCallSessionToQueueFailed()

The OnAddCallSessionToQueueFailed() event trigers when add call-session to queue process fails.

Syntax

```
OnAddCallSessionToQueueFailed(QueueName, SessionId)
```

Parameters

QueueName (string)
: The value of this parameter specifies queue name.

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

Example

```
OnAddCallSessionToQueueFailed(QueueName, SessionId)
{

}
```

See Also

AddCallSessionToQueue()

OnAddCallSessionToRingGroupSuccess()

The OnAddCallSessionToRingGroupSuccess() event trigers when VaxTele adds a call-session to ring group successfully.

Syntax

```
OnAddCallSessionToRingGroupSuccess(GroupName, SessionId)
```

Parameters

GroupName (string)
: The value of this parameter specifies group name.

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

Example

```
OnAddCallSessionToRingGroupSuccess(GroupName, SessionId)
{

}
```

See Also

AddCallSessionToRingGroup()

OnAddCallSessionToRingGroupFailed()

The OnAddCallSessionToRingGroupFailed() event trigers when the process of adding call-session to ring group fails.

Syntax

```
OnAddCallSessionToRingGroupFailed(GroupName, SessionId)
```

Parameters

GroupName (string)
: The value of this parameter specifies group name.

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

Example

```
OnAddCallSessionToRingGroupFailed(GroupName, SessionId)
{

}
```

See Also

AddCallSessionToRingGroup()

OnServerConnectingREC()

The OnServerConnectingREC() event triggers when VaxTele starts connecting a call of a call-session to SIP REC protocol supported recording server.

Syntax

```
OnServerConnectingREC(
                        SessionId,
                        ChannelId,
                        StatusCode,
                        ReasonPhrase
                       )
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

StatusCode (integer)
This parameter specifies SIP response status code (486, 404 etc).
LIST OF SIP RESPONSES

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

Example

```
OnServerConnectingREC(SessionId, ChannelId, StatusCode, ReasonPhrase)
{
}
```

See Also

ConnectToServerREC(), OnIncomingCall()

OnServerConnectedREC()

The OnServerConnectedRec() event triggers when a call of a call-session connects to the SIP REC protocol supported recording server successfully.

Syntax

```
OnServerConnectedREC(
                        SessionId,
                        ChannelId,
                        )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

Example

```
OnServerConnectedREC(SessionId, ChannelId)
{
}
```

See Also

ConnectToServerREC(), OnIncomingCall()

OnServerFailedREC()

The OnServerFailedREC() event triggers when VaxTele fails to connect a specific call of a call-session to the SIP REC recording server.

Syntax

```
OnServerFailedREC(
                    SessionId,
                    ChannelId,
                    StatusCode,
                    ReasonPhrase
                    )
```

Parameters

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

ChannelId (integer)
: This parameter value identifies a call in a call-session.

  0 = Channel-ZERO call
  1 = Channel-ONE call

StatusCode (integer)
: This parameter specifies SIP response status code (486, 404 etc).
  LIST OF SIP RESPONSES

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

Example

```
OnServerFailedREC(SessionId, ChannelId, StatusCode, ReasonPhrase)
{
}
```

See Also

ConnectToServerREC(), OnIncomingCall()

OnServerTimeoutREC()

The OnServerTimeoutREC() event triggers when remote SIP REC server disconnects a specific call of a call-session.

Syntax

```
OnServerTimeoutREC(SessionId, ChannelId)
```

Parameters

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

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

```
OnServerTimeoutREC(SessionId, ChannelId)
{
}
```

See Also

ConnectToServerREC(), OnIncomingCall()

OnServerHangUpREC()

The OnServerHangUpREC() event triggers when any party close the call.

Syntax

```
OnServerHangUpREC(SessionId, ChannelId)
```

Parameters

SessionId (integer)
This parameter specifies a unique identification of a call-session.

ChannelId (integer)
This parameter value identifies a call in a call-session.

0 = Channel-ZERO call
1 = Channel-ONE call

Example

```
OnServerHangUpREC(SessionId, ChannelId)
{
}
```

See Also

OnIncomingCall() ConnectToServerREC()

WHAT IS CALL-SESSION

Call-Session is a collection of either one or two calls. In a Call-Session, calls are identified as Channel-ZERO call and Channel-ONE call.

Call-Session with two calls (Channel-ZERO call & Channel-ONE call)

Channel-ZERO call	Channel-ONE call

Call-Session with one call (Channel-ZERO call)

Channel-ZERO call	

Call-Session with one call (Channel-ONE call)

	Channel-ONE call

Call-Session with two calls (Channel-ZERO call & Channel-ONE call)

VaxTele COM component receives SIP based call request(s), upon receiving call request(s) it internally creates/allocates a Call-Session. It adds incoming call as Channel-ZERO call to the Call-Session and triggers OnIncomingCall() event.

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, RouteUID, RouteType, RouteName,
                     UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, CallerName, CallerId, DialNo,
                              ToPeerName, RouteUID, Timeout)
}
```

AcceptCallSession() with appropriate values in the DialNo, ToPeerName, initiates a outgoing call request and adds outgoing call as Channel-ONE call to the same Call-Session (created upon incoming call). The AcceptCallSession() also accepts/connects the Channel-ZERO (incoming call) call in that Call-Session.

Call-Session with one call (Channel-ZERO call)

VaxTele COM component receives SIP based call request(s), upon receiving call request(s) it internally creates/allocates a Call-Session. It adds incoming call as Channel-ZERO call to Call-Session and triggers OnIncomingCall() event.

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
                    FromPeerName, RouteUID, RouteType, RouteName,
                    UserAgentName, FromIP, FromPort)
{
        AcceptCallSession(SessionId, “”, “”, “”, “”, “”, 20);
}
```

To only accepts/connects the incoming call without generating outgoing call AcceptCallSession() is called with no values in DialNo, ToPeerName and RoutID.

Call-Session with one call (Channel-ONE call)

VaxTele COM component exports a function DialCallSession(), that function sends/dials a SIP based call request. Such method is used to develop predictive dialer and telemarketing softwares.

```
OnVaxTeleTick(TickId)
{
        SessionId =  DialCallSession(CallerName, CallerId, DialNo,
                                        ToPeerName, 20)
}
```

DialCallSession() internally creates a new Call-Session and adds that outbound call as Channel-ONE call to the Call-Session.

LIST OF ERROR CODES

200	Failed to initialize RTP Socket.
201	Failed to allocate RTP port or provided value of RTP listen IP is incorrect.
202	Failed to create RTP task manager.
203	Failed to start media thread.
204	Unable to create RTP communication manager.
205	Unable to run 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	Provided wave id is not valid.
210	Unable to start recording manager.
211	Voice session is not connected or started yet.
212	Failed to initialize VaxVoIP Library.
213	Unable to construct SIP SDP body.
214	Unable to construct SIP INVITE request.
215	Error to initialize SIP communication layer.
216	Failed to open SIP listen port or provided SIP listen IP is incorrect.
217	Failed to create SIP task manager.
218	Unable to create SIP REGISTER request.
219	Unable to create SIP UN-REGISTER request.
220	Unable to create SIP BYTE request.
221	Unable to create SIP CANCEL request.
222	Provided SessionId does not exist.
223	Voice session does not exist.
224	Provided login does not exist.
225	Login length is incorrect.
226	Provided line does not exist.
227	Can't send SIP response.
228	Invalid SIP response code, please check SIP RFC 3261.
229	Error to create SIP response.
230	Provided line already exist.
231	Incorrect RegId or RegId does not exist.
232	Error to create SIP response.
233	User is not registered.
234	Invalid codec OR no codec found for voice streaming.
235	Invalid DTMF type.
236	Remote party does not support RFC2833 DTMF digits.
237	Error to create REFER request to transfer the call.
238	Error to create event handler.
239	License key is not valid or expired.
240	Invalid digit for DTMF.
241	Invalid proxy URI.
242	Line is not registered.
243	Invalid SIP To-URI.

244	Desired operation can only be performed on connected session.
245	Can't perform desired operation on connected session.
246	Direct communication is not supported.
247	One of the provided parameter(s) value is not correct.
248	Invalid chat message-Id.
249	Invalid subscribe-Id.
250	Failed to generate a unique-Id.
251	Provided unique-Id is not valid.
252	Provided unique-Id or value already exist.
253	Provided unique-Id or value does not exist.
254	Failed to create session.
255	Failed to enable DTMF detection.
256	Error to process transfer request.
257	Provided ListenIP is already binded.
258	Provided ListenIP does not exist in Server Key.
259	Provided SessionId has No Free Channel.
260	Both users (pickup and ringing) should be members of same pickup group.
261	Crypto audio or video media mismatched.
262	Unable to create socket dispatcher.
263	Unable to post message to socket dispatcher.
264	Provided addr is not valid.
265	Unable to bind socket addr or IP.
266	Failed to allocate socket port or provided value of listen IP or port is already in use.
267	General socket or network failure.
268	Failed to add wait-event to socket dispatcher.
269	Unable to create socket (timeout).
270	Line catch-all can't be used.
271	Route prefix conflicts with another route.
272	Unable to find provided SessionId.
273	Unable to find incoming call or inbound call does not exist.
274	Inbound call already connected.
275	Unable to connect socket. Provided address or port is not valid.
276	Unable to capture socket. Provided address or port is not valid.
277	User already attached.
278	Unable to create thread dispatcher.
279	Unable to post message to thread dispatcher.
280	Unable to process to thread dispatcher.
281	Terminating abnormally thread dispatcher.
282	Unable to create pipe dispatcher.
283	Unable to post message to pipe dispatcher.
284	Unable to process to pipe dispatcher.
285	Failed to add wait-event to pipe dispatcher.
286	General pipe communication failure.
287	Unable to create pipe (timeout).

LIST OF SIP RESPONSES (SIP RFC 3261)

Provisional responses 1xx			
100	Trying	180	Ringing
181	Call Is Being Forwarded	182	Queued
183	Session Progress		
Redirection 3xx			
300	Multiple Choices	301	Moved Permanently
302	Moved Temporarily	305	Use Proxy
380	Alternative Service		
Request Failure 4xx			
400	Bad Request	401	Unauthorized
402	Payment Required	403	Forbidden
404	Not Found	405	Method Not Allowed
406	Not Acceptable	407	Proxy Authentication Required
408	Request Timeout	410	Gone
413	Request Entity Too Large	414	Request-URI Too Long
415	Unsupported Media Type	416	Unsupported URI Scheme
420	Bad Extension	421	Extension Required
423	Interval Too Brief	480	Temporarily Unavailable
481	Call/Transaction Does Not Exist	482	Loop Detected
483	Too Many Hops	484	Address Incomplete
485	Ambiguous	486	Busy Here
487	Request Terminated	488	Not Acceptable Here
491	Request Pending	493	Undecipherable
Server Failure 5xx			
500	Server Internal Error	501	Not Implemented
502	Bad Gateway	503	Service Unavailable
504	Server Time-out	505	Version Not Supported
513	Message Too Large		
Global Failures 6xx			
600	Busy Everywhere	603	Decline
604	Does Not Exist Anywhere	606	Not Acceptable

SIP CLIENT REGISTRATION FLOW

Please see SIP CLIENT REGISTRATION FLOW document for further details.

SIP PHONE TO SIP PHONE CALL FLOW

Please see SIP PHONE TO SIP PHONE CALL FLOW document for further details.

HOW TO CONNECT TO PSTN/GSM NETWORK

Please see HOW TO CONNECT TO PSTN/GSM NETWORK document for more details.

HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP)

Please see HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP) document for more details.