



SIP SERVER SDK v8.0

TECHNICAL DOCUMENTATION

VERSION 8.0.6.8

CONTENTS

INTRODUCTION AND QUICK START 7

EXPORTED FUNCTIONS 8

SetLicenseKey()	8
GetVaxErrorCode()	9
Initialize()	10
UnInitialize()	12
OpenNetworkUDP()	13
OpenNetworkTCP()	14
OpenNetworkTLS()	15
CloseNetworkUDP()	17
CloseNetworkTCP()	18
CloseNetworkTLS()	19
SetListenPortRangeRTP()	20
AddNetworkRouteSIP(), AddNetworkRouteRTP()	21
AddUser()	23
RemoveUser()	26
RegisterUserExpiry()	28
AcceptRegister()	29
RejectRegister()	31
AuthRegister()	33
AddLine()	35
RemoveLine()	39
RegisterLine()	40
UnRegisterLine()	42
AcceptCallSession()	43
RejectCallSession()	45
CloseCallSession()	46
DialCallSession()	47
AccessAudioPCM()	49
SendAudioPCM()	52
AcceptTransferBlind()	54
AcceptTransferConsult()	56
RejectTransfer()	58
LoadWaveFile()	59
LoadWavePCM()	61
UnLoadWaveID()	63
PlayWaveStartToCallSession()	65
PlayWaveStopToCallSession()	67
PlayWaveSetVolumeToCallSession()	69
PlayWaveSetModeTypeToCallSession()	71
RecordWaveSetCacheSize()	73
RecordWaveStartToCallSession()	75
RecordWaveStopToCallSession()	77
RecordWavePauseToCallSession()	78
LoadMusicHold()	80
UnLoadMusicHold()	81
AcceptOnHoldRequest()	82

AcceptOffHoldRequest()	83
AcceptChatMessage()	84
RejectChatMessage()	85
SendChatMessageText()	86
AcceptChatStatusSubscribe()	87
RejectChatStatusSubscribe()	89
OpenConferenceRoom()	90
CloseConferenceRoom()	91
AddCallSessionToConferenceRoom()	92
RemoveCallSessionFromConferenceRoom()	93
PlayWaveStartToConferenceRoom()	94
PlayWaveStopToConferenceRoom()	95
PlayWaveSetVolumeToConferenceRoom()	96
RecordWaveStartToConferenceRoom()	97
RecordWaveStopToConferenceRoom()	99
RecordWavePauseToConferenceRoom()	100
AudioSessionLost()	101
SendInfoVM()	102
DiagnosticLogSIP()	103
StartVaxTeleTick()	104
StopVaxTeleTick()	105
SetUserAgentName()	106
GetUserAgentName()	107
SetSessionNameSDP()	108
GetSessionNameSDP()	109
SendDigitDTMF()	110
DetectDigitDTMF()	111
DialToneToCallSession()	114
SplitCallSession()	115
MoveCallSession()	116
VideoCOMM()	118
GetCallSessionTxCodec()	119
GetCallSessionRxCodec()	120
CallSessionMuteVoice()	121
BusyLampSubscribeAccept()	122
BusyLampSubscribeReject()	123
BusyLampSendStatus()	124
AddCustomHeader()	125
RemoveCustomHeader()	126
RemoveCustomHeaderAll()	127
CallSessionDetectAMD()	128
CallSessionSendStatusResponse()	130
GetCallSessionHeaderCallId()	131
ConnectToServerREC()	132
MuteCallServerREC()	134
SetExtDataREC()	136
SendReqTransferBlind()	137
SendReqTransferCallConsult()	138
AddRingGroup()	140
RemoveRingGroup()	142
AddRingGroupAgent()	143
RemoveRingGroupAgent()	144
SetRingGroupProcessMode()	145

SetRingGroupAgentPriority()	146
AddCallSessionToRingGroup()	147
AddCallPickUpGroup()	148
RemoveCallPickUpGroup()	150
AddCallPickUpGroupMember()	151
RemoveCallPickUpGroupMember()	152
PickUpCall()	153
ParkCallSession()	154
ConnectToParkedCallSession()	156
AddQueue()	158
RemoveQueue()	159
AddQueueAgent()	160
RemoveQueueAgent()	161
SetQueueAgentPriority()	162
SetQueueProcessMode()	163
AddCallSessionToQueue()	164
AddStealthListener()	165
AdjustCallSessionVoiceType()	167
AdjustCallSessionVideoType()	169
CallSessionAttachCall()	171
CallSessionDetachCall()	173
ActivatePushSIP()	175
SetUserPushSIP()	177
AddUserRouteUID()	179
AddLineRouteUID()	180
AddRingGroupRouteUID()	181
AddCallPickUpGroupRouteUID()	182
AddQueueRouteUID()	183
AddConferenceRoomRouteUID()	184
AddCallParkRouteUID()	185
AddStealthListenRouteUID()	186
AttackDetectScanSIP()	187
AttackDetectFloodSIP()	188
AttackDetectBruteForceSIP()	189

EXPORTED EVENTS 190

OnVaxErrorLog()	190
OnCallSessionErrorLog()	191
OnCallSessionCreated()	192
OnCallSessionClosed()	193
OnCallSessionAccessAudioPCM()	194
OnRegisterUser()	196
OnRegisterUserSuccess()	198
OnRegisterUserFailed()	199
OnRegisterUserPushSIP()	200
OnUnRegisterUser()	201
OnLineRegisterTrying()	202
OnLineRegisterFailed()	203
OnLineRegisterSuccess()	204
OnLineUnRegisterTrying()	205
OnLineUnRegisterFailed()	206
OnLineUnRegisterSuccess()	207

OnIncomingCall()	208
OnCallSessionConnecting()	210
OnCallSessionFailed()	211
OnCallSessionConnected()	212
OnCallSessionLost()	213
OnCallSessionHangup()	214
OnCallSessionTimeout()	215
OnCallSessionCancelled()	216
OnCallSessionOnHold()	217
OnCallSessionOffHold()	218
OnCallSessionTransferBlind()	219
OnCallSessionTransferConsult()	221
OnCallSessionTransferring()	223
OnCallSessionTransferTimeout()	227
OnCallSessionTransferred()	228
OnSendReqTransferCallTimeout()	229
OnSendReqTransferCallAccepted()	230
OnSendReqTransferCallFailed()	231
OnDetectedDigitDTMF()	232
OnOutgoingDiagnosticLog()	233
OnIncomingDiagnosticLog()	234
OnVaxTeleTick()	235
OnSendTimeoutVM()	236
OnSendSuccessVM()	237
OnCallSessionDialToneStarted()	238
OnCallSessionDialToneEnded()	239
OnCallSessionPlayWaveDone()	240
OnConferenceRoomPlayWaveDone()	241
OnCallSessionDetectAMD()	242
OnChatMessageText()	243
OnChatMessageTyping()	245
OnChatStatusSubscribe()	247
OnChatMessageSuccess()	249
OnChatMessageFailed()	250
OnChatMessageTimeout()	251
OnBusyLampSubscribe()	252
OnBusyLampUnSubscribe()	253
OnBusyLampSubscribeSuccess()	254
OnBusyLampSubscribeFailed()	255
OnBusyLampSendStatus()	256
OnAddCallSessionToQueueSuccess()	257
OnAddCallSessionToQueueFailed()	258
OnQueuePlayWaveStarted()	259
OnQueuePlayWaveEnded()	260
OnQueueAgentConnectStarted()	261
OnQueueAgentConnectTrying()	262
OnQueueAgentConnectFailed()	263
OnQueueAgentConnectTimeout()	264
OnQueueAgentConnectSuccess()	265
OnAddCallSessionToRingGroupSuccess()	266
OnAddCallSessionToRingGroupFailed()	267
OnCallParkPlayWaveStarted()	268
OnCallParkPlayWaveEnded()	269

OnServerConnectingREC()	270
OnServerConnectedREC()	271
OnServerFailedREC()	272
OnServerTimeoutREC()	273
OnServerHungupREC()	274
OnCallSessionRecordedWaveREC()	275
OnConferenceRoomRecordedWaveREC()	277
OnAttackDetectedScanSIP()	279
OnAttackDetectedFloodSIP()	281
OnAttackDetectedBruteForceSIP()	283

WHAT IS A CALL-SESSION..... 285

Call-Session with two calls (Channel-ZERO call & Channel-ONE call)	285
Call-Session with one call (Channel-ZERO call)	285
Call-Session with one call (Channel-ONE call)	285

LIST OF ERROR CODES 287

LIST OF SIP RESPONSES (SIP RFC 3261) 290

SIP CLIENT REGISTRATION PROCESS 291

SIP PHONE TO SIP PHONE CALL FLOW 291

ACCESS AUDIO DATA PCM PROCESS..... 291

HOW TO CONNECT TO PSTN/GSM NETWORK..... 291

HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER (ITSP)..... 291

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 the VaxVoIP SDK is limited to a 30-day evaluation period. After this period, a license key is required to continue using the SDK without interruption and to remove the evaluation message box that appears during usage. License keys are provided to customers upon purchase.

To convert the trial version into a fully registered version, without any expiration or trial period limitations, you can use the SetLicenseKey() method. By invoking this method with a valid license key, the SDK will operate without any time restrictions.

Syntax

```
SetLicenseKey(LicenseKey)
```

Parameters

LicenseKey (string)

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

Return Value

No return value.

Example

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

See Also

Initialize(), GetVaxErrorCode()

GetVaxErrorCode()

The GetVaxErrorCode() method retrieves the error code associated with the last operation that failed to execute. This error code provides specific information about the nature of the failure, which can be useful for debugging and troubleshooting.

For a detailed explanation of the possible error codes and their meanings, please refer to the [LIST OF ERROR CODES](#) section.

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

OnVaxErrorLog(), OnCallSessionErrorLog()

Initialize()

The Initialize() function is responsible for setting up the VaxVoIP SIP Server COM component. This process includes allocating internal memory resources and configuring the component for integration with your application.

Once this function is executed, the component becomes fully operational, allowing it to expose its methods for external use. This setup ensures that the application can effectively interact with the component and utilize its various functionalities.

Syntax

```
boolean Initialize(DomainRealm)
```

Parameters

DomainRealm (string)

This parameter value is used internally to generate SIP URIs and is included as the "realm" field in SIP packets during the authentication of SIP clients (such as softphones and hardphones) and call processing.

Parameter Purpose: The parameter dictates the domain used for SIP URIs and authentication. It is utilized as the "realm" field in SIP packets.

Blank/Empty Value: If this parameter is left blank or set to an empty string, the SIP authentication process and SIP URI generation will use the assigned IP address instead.

Examples: With DomainRealm Set: If the DomainRealm value is set to sip.vaxvoip.com, VaxVoIP will create SIP URIs in the format sip:username@sip.vaxvoip.com.
With Empty DomainRealm: If the DomainRealm is set to an empty string and the assigned IP address is 10.3.5.66, VaxVoIP will generate SIP URIs in the format sip:username@10.3.5.66.

Note: It is not mandatory for an IP address to be assigned to the DomainRealm parameter.

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("")  
or  
Initialize("demo.vaxvoip.com")  
or  
Initialize("vaxvoip.com")
```

See Also

UnInitialize(), GetVaxErrorCode(), SetListenPortRangeRTP()

UnInitialize()

The UnInitialize() function is responsible for releasing all resources that were allocated by the Initialize() function.

Its primary purpose is to ensure a clean and orderly release of resources when they are no longer needed, helping to prevent resource leaks and maintain system stability.

Syntax

```
UnInitialize()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
UnInitialize()
```

See Also

[Initialize\(\)](#)

OpenNetworkUDP()

The OpenNetworkUDP() function initializes a UDP socket, assigns a listening port, and begins monitoring for incoming SIP requests. It also manages the sending of SIP responses over UDP. Based on the received SIP requests, the relevant COM (Component Object Model) events are triggered.

Syntax

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

Parameters

ListenIP (string)

This parameter specifies the IP address on which VaxVoIP listens for incoming SIP requests over UDP. It can either be an empty value or an IP address assigned to the computer where the VaxVoIP COM-integrated SIP server is running.

ListenPort (integer)

This parameter defines the port number on which the SIP server receives SIP requests. The standard SIP listening port is 5060.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
OpenNetworkUDP("", 5060)  
  
or  
  
OpenNetworkUDP("192.168.0.3", 5060)
```

See Also

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

OpenNetworkTCP()

The OpenNetworkTCP() function sets up a TCP socket, assigns a listening port, and begins monitoring for incoming SIP requests. It also handles sending SIP responses over TCP. Upon receiving SIP requests, the corresponding COM (Component Object Model) events are triggered.

Syntax

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

Parameters

ListenIP (string)

This parameter specifies the IP address on which VaxVoIP listens for incoming SIP requests over TCP. It can be either an empty value or an IP address assigned to the computer running the VaxVoIP COM-integrated SIP server.

ListenPort (integer)

This parameter defines the port number on which the SIP server accepts SIP requests. The default SIP listening port is 5060.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

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

OpenNetworkTLS()

The OpenNetworkTLS() function establishes a secure TLS communication channel, allocates a TLS listening port, and begins monitoring for incoming SIP requests over TLS. It also handles and manages the sending of outgoing SIP responses. Based on the received SIP requests, the relevant COM (Component Object Model) events are triggered.

Syntax

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

Parameters

ListenIP (string)

This parameter specifies the IP address on which VaxVoIP listens for incoming SIP requests over TLS. It can be an empty value or an IP address assigned to the computer running the VaxVoIP COM-integrated SIP server.

ListenPort (integer)

This parameter defines the port number on which the SIP server receives SIP requests over TLS. The standard SIP listening port for TLS is 5061.

CertPEM (string)

This parameter specifies the filename or data of the SSL certificate in PEM format. It can be provided as a string or text value. If it is empty, VaxVoIP COM uses the default VaxVoIP SSL certificate.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
OpenNetworkTLS("", 5061, "")  
  
or  
  
OpenNetworkTLS("192.168.0.3", 5061, "")
```

See Also

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

CloseNetworkUDP()

The CloseNetworkUDP() function closes a UDP socket, terminating its operation and releasing all associated network resources. This function ensures that no further data transmission or reception occurs on the socket, effectively disconnecting it from the network.

Syntax

```
CloseNetworkUDP()
```

Parameters

No parameters.

Return Value

No return value.

Example

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

See Also

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

CloseNetworkTCP()

The CloseNetworkTCP() function closes a TCP socket, terminating the connection and releasing all associated resources. This function ensures a proper shutdown of the TCP connection, preventing any further data transmission or reception. By closing the socket, the function helps to free up system resources and maintain network stability.

Syntax

```
CloseNetworkTCP()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkUDP("", 5060)
OpenNetworkTCP("", 5060)

CloseNetworkTCP()
CloseNetworkUDP()
```

See Also

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

CloseNetworkTLS()

The CloseNetworkTLS() function securely terminates a TLS communication channel, closing the underlying socket and releasing all associated resources. This function ensures that the secure connection is properly shut down, preventing any further encrypted data transmission or reception. By closing the TLS channel, the function helps maintain the security and integrity of the network while freeing up system resources.

Syntax

```
CloseNetworkTLS()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkUDP("", 5060)
OpenNetworkTCP("", 5060)
OpenNetworkTLS("", 5061, "")

CloseNetworkTLS()
CloseNetworkTCP()
CloseNetworkUDP()
```

See Also

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

SetListenPortRangeRTP()

The SetListenPortRangeRTP() function sets the specified port range for VaxVoIP to allocate and open the RTP listen port.

The listen port must comply with SIP RFC 2327, requiring it to be within the range of 1024 to 65535 and an even number for RTP compliance.

Syntax

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

Parameters

ListenStartPort (integer)

This parameter defines the starting port of the specified range for the RTP listen port.

ListenEndPort (integer)

This parameter defines the ending port of the specified range for the RTP listen port.

Return Value

The function returns a non-zero value upon successful execution; otherwise, it returns 0. In case of failure, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

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

OpenNetworkUDP("", 5060)

SetListenPortRangeRTP(20000, 40000)
```

See Also

Initialize(), GetVaxErrorCode()

AddNetworkRouteSIP(), AddNetworkRouteRTP()

The AddNetworkRouteSIP() and AddNetworkRouteRTP() functions allow the addition of supplementary IP addresses to facilitate the routing of SIP and RTP packets, respectively. These functions are particularly useful in network configurations where the VaxVoIP integrated Server SIP is positioned behind a firewall, router, or NAT with port forwarding enabled.

In such scenarios, it is recommended to initialize VaxVoIP with the private IP address assigned to the computer. By doing so, the server can communicate effectively within the local network.

To ensure proper routing of SIP and RTP packets through the public network, the AddNetworkRouteSIP() and AddNetworkRouteRTP() functions should be used to incorporate the public IP address assigned to the router.

This setup ensures that the VaxVoIP server can handle incoming and outgoing traffic correctly, maintaining seamless communication despite the presence of NAT or other network barriers.

Syntax

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

Parameters

AssignedIP (string)

This parameter specifies the private IP address assigned to the computer within the local network.

RouterIP (string)

This parameter specifies the public IP address assigned to the router, used for external network communication.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

Result = OpenNetworkUDP("192.168.0.25", 5060)
if(Result = 0) GetVaxErrorCode()

Result = OpenNetworkTCP("192.168.0.25", 5060)
if(Result = 0) GetVaxErrorCode()

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

SetListenPortRangeRTP(10000, 20000)
```

To enable port forwarding and ensure the SIP server functions correctly behind a firewall or router, follow these steps in your router's settings:

Forward Inbound UDP Ports 10000 to 20000: Configure the router to forward incoming UDP traffic on ports 10000 to 20000 to the computer running the SIP server.

Enable Outbound UDP Ports 1024 to 65535: Allow outbound UDP traffic on ports 1024 to 65535 to ensure that the SIP server can communicate with external networks.

Assign IP Addresses: Ensure that the IP address 192.168.0.25 is assigned to the computer behind the router and 66.77.88.99 is the public IP address assigned to the router.

This setup will enable the SIP server to operate effectively behind the firewall, allowing proper routing of SIP and RTP packets.

See Also

Initialize(), SetListenPortRangeRTP(), GetVaxErrorCode(),
OpenNetworkUDP(), OpenNetworkTCP(), OpenNetworkTLS()

AddUser()

The AddUser() function adds users to the SIP server developed using the VaxVoIP COM component (VaxTeleServerCOM.dll). This function is essential for managing SIP user accounts, as it integrates users into the server's internal system.

When the AddUser() function is invoked, it internally creates and maintains a list of users that the SIP server uses to handle SIP registration and call requests. This user list is crucial for processing incoming connection requests and ensuring the SIP server can properly authenticate and manage user sessions.

By using the AddUser() function, you enable users to log in and register with the VaxVoIP-based SIP server. Usernames and passwords can then be used by SIP-based softphones or hardphones to connect to and interact with the server, facilitating effective communication and integration within the SIP network.

Syntax

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

Parameters

UserName (string)

This parameter specifies the user's login name.

Password (string)

This parameter specifies the password associated with the user's login.

AudioCodecList (string)

This parameter specifies the list of audio codecs to be used in the call request.

0 = GSM

1 = iLBC

2 = G711 A-Law

3 = G711 U-Law

4 = G729

"03" indicates that only GSM and G711 U-Law codecs are supported for the call, with GSM having a higher priority.

"4213" indicates that the supported codecs for the call request are G729, G711 A-Law, iLBC, and G711 U-Law. In this case, G729 has the highest priority (4), and G711 U-Law has the lowest priority (3).

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result == 0) GetVaxErrorCode()

// 3 = G711 U-Law, 2 = G711 A-Law, 4 = G729
// (G711 U-Law has the highest priority, G729 has the lowest priority)

Result = AddUser("9090", "123", "324")
if(Result == 0) GetVaxErrorCode()

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

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

OnRegisterUser(UserName, Domain, UserAgentName, FromIP, FromPort,
               RegId)
{
    // Fetch UserName details from storage (RecordDB)

    Result = AddUser(UserName, RecordDB.Password,
                    RecordDB.AudioCodec)

    if(Result == 0) GetVaxErrorCode()

    AuthRegister(RegId)
}

OnRegisterUserFailed(UserName, FromIP, FromPort, RegId)
{
    RemoveUser(UserName)
}

OnUnRegisterUser(UserName)
{
    RemoveUser(UserName)
}
```

See Also

RemoveUser(), RejectRegister(), AcceptRegister(), AuthRegister(),
GetVaxErrorCode(), OnRegisterUser(), OnUnRegisterUser(),
OnRegisterUserFailed(), OnRegisterUserSuccess()

RemoveUser()

The RemoveUser() function deletes a user that was previously added using the AddUser() function.

This function is used to remove a user account from the SIP server, ensuring that the user can no longer access or interact with the server.

By calling RemoveUser(), you effectively clean up the user list and free up resources associated with that user account.

Syntax

```
RemoveUser(Username)
```

Parameters

Username (string)

This parameter specifies the user's login name.

Return Value

No return value.

Example

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

OnRegisterUser(Username, Domain, UserAgentName, FromIP, FromPort,
               RegId)
{
    // Retrieve Username account details from storage (RecordDB)

    // 2 = G711a, 4 = G729, 3 = G711u
    // (G711a has the highest priority, G711u has the lowest priority)

    Result = AddUser(Username, RecordDB.AccountPassword, "243")
    if(Result == 0) GetVaxErrorCode()

    AuthRegister(RegId)
}

OnRegisterUserFailed(Username, FromIP, FromPort, RegId)
{
    RemoveUser(Username)
}

OnUnRegisterUser(Username)
{
    RemoveUser(Username)
}
```

See Also

AddUser(), RejectRegister(), AcceptRegister(), AuthRegister(),
GetVaxErrorCode(), OnRegisterUser(), OnUnRegisterUser(),
OnRegisterUserFailed(), OnRegisterUserSuccess()

RegisterUserExpiry()

The RegisterUserExpiry() function configures the expiration interval for SIP client registrations in the VaxVoIP SIP server. This setting determines how long the registration information for a SIP client remains valid before it requires renewal.

If the value of -1 is set, the SIP server accepts the expiration interval requested by the SIP client during the registration process.

If a value other than -1 is set, the SIP server ignores the client's requested expiration interval and uses the value specified by RegisterUserExpiry() instead. This allows for customized control over the expiration interval.

If RegisterUserExpiry() is not called explicitly, the SIP server defaults to an expiration interval of 30 seconds, which it communicates to the SIP client(s) during registration.

The SIP packet's Expires header designates the expiration interval of the registration.

Syntax

```
boolean RegisterUserExpiry(Expiry)
```

Parameters

Expiry (integer)

This parameter specifies the time interval, in seconds, for the expiration of the registration.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

RegisterUserExpiry(1800)
```

See Also

AddUser(), OnRegisterUser(), AcceptRegister(), AuthRegister()

AcceptRegister()

The AcceptRegister() function processes and accepts registration requests received from SIP clients, such as softphones and hardphones.

During the SIP registration process, SIP clients send registration requests to the SIP server. The server is responsible for authorizing the provided login credentials and then accepting these requests.

When VaxVoIP receives a registration request, it triggers the OnRegisterUser() event. Within this event, calling the AcceptRegister() function accepts the registration request without performing any authorization of the login credentials.

In essence, the AcceptRegister() function bypasses the login authorization process, directly accepting the registration request as-is. This function is used when you want to allow registration requests to be accepted without validating user credentials.

For additional information, refer to the [SIP CLIENT REGISTRATION PROCESS](#)

Syntax

```
boolean AcceptRegister(RegId)
```

Parameters

RegId (integer)

This parameter identifies a unique registration session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result == 0) GetVaxErrorCode()

OnRegisterUser(Username, Domain, UserAgentName, FromIP, FromPort,
               RegId)
{
    // Fetch Username details from storage (RecordDB)

    Result = AddUser(Username, RecordDB.Password,
                    RecordDB.AudioCodec)

    if(Result == 0) GetVaxErrorCode()

    AcceptRegister(RegId)
}

OnRegisterUserFailed(Username, FromIP, FromPort, RegId)
{
    RemoveUser(Username)
}

OnUnRegisterUser(Username)
{
    RemoveUser(Username)
}
```

See Also

AddUser(), RemoveUser(), RejectRegister(), AuthRegister(),
GetVaxErrorCode(), OnRegisterUser(), OnUnRegisterUser(),
OnRegisterUserFailed(), OnRegisterUserSuccess()

RejectRegister()

The RejectRegister() function processes and rejects registration requests from SIP clients directed to VaxVoIP.

When VaxVoIP receives a registration request, it triggers the OnRegisterUser() event. If the RejectRegister() function is invoked during this event, it denies the registration request, preventing the client from registering with the SIP server.

This function ensures that the SIP server can effectively handle invalid or unauthorized registration attempts.

For further details, refer to the [LIST OF SIP RESPONSES](#)

Syntax

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

Parameters

- RegId (integer)
This parameter specifies a unique identifier for a registration session.
- StatusCode (integer)
This parameter specifies the SIP response status code.
- ReasonPhrase (string)
This parameter specifies the SIP response reason phrase, such as Unauthorized or Not Found.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

OnRegisterUser(Username, Domain, UserAgentName, FromIP, FromPort,
               RegId)
{
    // Fetch Username details from storage (RecordDB)
    if(RecordDB not found)
    {
        RejectRegister(Username, 404, "Not Found")
    }
    else
    {
        AddUser(Username, RecordDB.Password, "32")
        AuthRegister(RegId)
    }
}

OnRegisterUserFailed(Username, FromIP, FromPort, RegId)
{
    RemoveUser(Username)
}

OnUnRegisterUser(Username)
{
    RemoveUser(Username)
}
```

See Also

AddUser(), RemoveUser(), AuthRegister(), AcceptRegister(),
GetVaxErrorCode(), OnRegisterUser(), OnUnRegisterUser(),
OnRegisterUserFailed(), OnRegisterUserSuccess()

AuthRegister()

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

SIP clients, including softphones, hardphones, and Wi-Fi phones, send SIP REGISTER requests to connect and register with SIP servers.

When VaxVoIP receives a registration request, it triggers the OnRegisterUser() event and starts the authentication process by calling the AuthRegister() function.

Upon calling the AuthRegister() function, VaxVoIP:

1. Starts the SIP authentication process.
2. Completes the authentication process.
3. Accepts the registration request.
4. Triggers either the OnRegisterUserSuccess() event if authentication is successful or the OnRegisterUserFailed() event if authentication fails.

Please see [SIP CLIENT REGISTRATION PROCESS](#) 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

OnRegisterUser(Username, Domain, UserAgentName, FromIP, FromPort,
               RegId)
{
    // Fetch Username details from storage (RecordDB)
    if(RecordDB not found)
    {
        RejectRegister(Username, 404, "Not Found")
    }
    else
    {
        AddUser(Username, RecordDB.Password, "32")
        AuthRegister(RegId)
    }
}

OnRegisterUserSuccess(Username, FromIP, FromPort, RegId)
{
}

OnRegisterUserFailed(Username, FromIP, FromPort, RegId)
{
    RemoveUser(Username)
}

OnUnRegisterUser(Username)
{
    RemoveUser(Username)
}
```

See Also

AddUser(), RemoveUser(), RejectRegister(), AcceptRegister(),
GetVaxErrorCode(), OnRegisterUser(), OnUnRegisterUser(),
OnRegisterUserFailed(), OnRegisterUserSuccess()

AddLine()

The AddLine() function integrates third-party SIP server account settings into VaxVoIP, allowing it to interact with other SIP servers. This functionality expands VaxVoIP's capabilities by enabling it to work seamlessly with external SIP servers and devices.

VaxVoIP is a SIP-based server SDK that can connect to other SIP servers and devices using the SIP protocol. By using the AddLine() function, you can configure VaxVoIP to send and receive call requests from these external SIP entities.

For instance, if you want VaxVoIP to interface with another SIP server, create a user account on that server and then use the AddLine() function to add the account settings to VaxVoIP. This setup enables VaxVoIP to manage calls with the external SIP server effectively.

Additionally, the AddLine() function allows VaxVoIP to connect to IP Telephony Service Providers (ITSPs) and supports both PSTN (Public Switched Telephone Network) and mobile calls. By configuring the appropriate settings with AddLine(), VaxVoIP can facilitate communication with ITSPs for VoIP services and manage calls over traditional PSTN and mobile networks.

Note: Before integrating SIP account settings from IP-Telephony Service Providers (ITSPs) into VaxVoIP, it is advisable to first test these settings with a softphone. This preliminary step helps verify that the SIP account settings are correctly configured and operational. Ensure that you can both make and receive calls successfully with the softphone. This confirmation guarantees that the settings are functioning as expected and will work reliably when integrated into VaxVoIP.

Please see [HOW TO CONNECT TO PSTN/GSM NETWORK](#) for more details.

Please see [HOW TO CONNECT TO IP-TELEPHONY SERVICE PROVIDER \(ITSP\)](#) for further details.

Syntax

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

Parameters

LineName (string)

Specifies the unique name assigned to a line for identifying it within the system.

LineType (integer)

Indicates the type of line being used:

0 = Line Type UDP

1 = Line Type TCP

2 = Line Type TLS

DisplayName (string)

Provides the display name for the user, as specified by the IP-Telephony Service Provider or third-party SIP server.

UserName (string)

Specifies the username provided by the IP-Telephony Service Provider or third-party SIP server.

AuthUser (string)

Indicates the user login name supplied by the IP-Telephony Service Provider or third-party SIP server.

AuthPwd (string)

Specifies the password provided by the IP-Telephony Service Provider or third-party SIP server.

DomainRealm (string)

Represents the domain realm information provided by the IP-Telephony Service Provider or third-party SIP server.

ServerAddr (string)

Defines the address of the SIP server as provided by the IP-Telephony Service Provider or third-party SIP server.

ServerPort (integer)

Specifies the port number of the SIP server, as provided by the IP-Telephony Service Provider or third-party SIP server.

AudioCodecList (string)

This parameter specifies the list of audio codecs to be used in the call request.

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

"03" indicates that only GSM and G711 U-Law codecs are supported for the call, with GSM having a higher priority.

"4213" indicates that the supported codecs for the call request are G729, G711 A-Law, iLBC, and G711 U-Law. In this case, G729 has the highest priority (4), and G711 U-Law has the lowest priority (3).

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

AddLine("LineDial", 0, "", "", "8034", "9341", "", "192.168.0.3", 5060, "32")
RegisterLine("LineDial", 1800)

OnLineRegisterTrying(LineName)
{
}

OnLineRegisterFailed(LineName, StatusCode, ReasonPhrase)
{
}

OnLineRegisterSuccess(LineName)
{
}
```

See Also

RemoveLine(), RegisterLine(), UnRegisterLine(), GetVaxErrorCode(),
OnLineRegisterFailed(), OnLineRegisterSuccess(), OnLineRegisterTrying(),
OnLineUnRegisterSuccess(), OnLineUnRegisterFailed()

RemoveLine()

The RemoveLine() function deletes a specified line that was previously added using the AddLine() function.

When you call RemoveLine(), it removes the line configuration from VaxVoIP, which was previously integrated into the system through AddLine(). This action effectively disconnects the SIP account or connection associated with that line, ceasing any interactions or communications that were facilitated by it.

This function is useful for managing and updating your SIP line configurations by allowing the removal of obsolete or unwanted lines.

Syntax

```
RemoveLine(LineName)
```

Parameters

LineName (string)

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

Return Value

No return value.

Example

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

AddLine("LineDial", 0, "", "", "8034", "9341", "", "192.168.0.3", 5060, "32")
RegisterLine("LineDial", 1800)

RemoveLine("LineDial")
```

See Also

AddLine(), RegisterLine(), UnRegisterLine(), GetVaxErrorCode(),
OnLineRegisterFailed(), OnLineRegisterSuccess(), OnLineRegisterTrying(),
OnLineUnRegisterSuccess(), OnLineUnRegisterFailed()

RegisterLine()

The RegisterLine() function initiates the registration of a line with an external third-party SIP server or IP-Telephony Service Provider (ITSP). During the registration process, VaxVoIP triggers various events related to the registration status, such as OnLineRegisterTrying() and OnLineRegisterSuccess().

Before calling RegisterLine(), ensure that the line has already been added using the AddLine() function. This prerequisite step is necessary for the registration process to proceed, as it provides the essential line configuration details required for successful communication with the external SIP server or ITSP.

Syntax

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

Parameters

LineName (string)

This parameter specifies the unique name assigned to identify a specific line within the system.

Expire (integer)

This parameter defines the time interval (in seconds) after which the registration with the server will be refreshed. This ensures that the server remains updated about the current status of the client, maintaining an active and consistent connection.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

AddLine("LineDial", 0, "", "", "8034", "9341", "", "192.168.0.3", 5060, "32")
RegisterLine("LineDial", 1800)

OnLineRegisterTrying(LineName)
{
}

OnLineRegisterFailed(LineName, StatusCode, ReasonPhrase)
{
}

OnLineRegisterSuccess(LineName)
{
}
```

See Also

AddLine(), RemoveLine(), UnRegisterLine(), GetVaxErrorCode(),
OnLineRegisterFailed(), OnLineRegisterSuccess(), OnLineRegisterTrying(),
OnLineUnRegisterSuccess(), OnLineUnRegisterFailed()

UnRegisterLine()

The UnRegisterLine() function removes the registration of a specified line that was previously registered using the RegisterLine() function.

During the unregistration process, VaxVoIP triggers events related to the status of this process, such as OnLineUnRegisterTrying() and OnLineUnRegisterSuccess().

These events help track and manage the progress and outcome of the unregistration, ensuring that the line is properly disconnected from the external SIP server or IP-Telephony Service Provider (ITSP).

Syntax

```
boolean UnRegisterLine(LineName)
```

Parameters

LineName (string)

Specifies the unique name assigned to identify a particular line within the system.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
UnRegisterLine("LineDial")
```

See Also

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

AcceptCallSession()

The AcceptCallSession() function accepts an incoming call and generates an outgoing call to establish a connection between the originating (From-Peer) and receiving (To-Peer) parties.

In the VaxVoIP server, a Call-Session is defined as a collection of one or two calls, which are referred to as Channel-ZERO and Channel-ONE calls. A Call-Session may consist of either a single call Channel-ZERO or Channel-ONE or two calls, representing both the incoming and outgoing connections.

The OnIncomingCall() event is triggered when the VaxVoIP-based SIP Server receives an incoming call request. In this event, use the AcceptCallSession() method to handle the incoming call request and initiate the appropriate connections.

In addition to handling traditional SIP calls, the VaxVoIP server SDK can be utilized to develop an IVR (Interactive Voice Response) system. In this scenario, the server accepts the incoming call, and once the call is connected, it can play a wave file and wait for DTMF (Dual-Tone Multi-Frequency) input from the caller.

For further details, [WHAT IS CALL-SESSION](#)

Syntax

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

Parameters

- SessionId (integer)
This parameter specifies a unique identifier for a call session.
- CallerName (string)
This parameter specifies the name of the caller.
- CallerId (string)
This parameter specifies a unique identifier for the caller.
- DialNo (string)
This parameter value specifies the number to be dialed.

ToPeerName (string)

Specifies the name of the recipient (To-Peer). If left as an empty string (""), the function will accept only the incoming call without generating an outgoing call. This is particularly useful for IVR systems that accept calls and play pre-recorded audio.

RouteUID (string)

Specifies the unique identifier or name assigned to various routing entities such as a Queue, RingGroup, Line, User, StealthListen, Room, etc. For further details, refer to the methods AddQueueRouteUID(), AddUserRouteUID(), AddLineRouteUID(), AddStealthListenRouteUID(), and others.

Timeout (integer)

Specifies the time interval (in seconds) for which VaxVoIP waits for the Call-Session to be connected or established. If the Call-Session is not connected within the specified time, a timeout occurs, triggering the OnCallSessionTimeout() event.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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(), OnCallSessionCreated(),
OnCallSessionClosed()

RejectCallSession()

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

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

StatusCode (integer)

This parameter specifies the SIP response status.

For further details, refer to the [LIST OF SIP RESPONSES](#)

ReasonPhrase (string)

Specifies the SIP response reason phrase, such as "Unauthorized" or "Not Found."

Return Value

Upon successful execution, the function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

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

See Also

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

CloseCallSession()

The CloseCallSession() function terminates the connection, causing all calls within that session to be disconnected.

This function is particularly useful in various scenarios where the VaxVoIP SDK-integrated SIP server needs to disconnect a Call-Session. For example, if during a call, the SIP server detects that a user's balance is insufficient, it can invoke the CloseCallSession() method to disconnect the call and terminate the session.

Syntax

```
boolean CloseCallSession(SessionId)
```

Parameters

SessionId (integer)
specifies a unique identifier for a call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
CloseCallSession(SessionId)
```

See Also

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

DialCallSession()

A Call-Session consists of either one or two calls, identified as Channel-ZERO and Channel-ONE calls.

The DialCallSession() function creates a new Call-Session by initiating an outgoing call and designating this call as the Channel-ONE call within the session.

For further details, please refer to [WHAT IS CALL-SESSION](#)

A typical use case for the DialCallSession() function is in telemarketing. In this scenario, the SIP server uses the function to dial out to a number. Once the call is connected, the server plays a pre-recorded wave file containing the message.

This function supports various applications, including automated messaging and interactive telephony services.

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 specifies the number to be dialed.

ToPeerName (string)
This parameter specifies the name of the peer, which can be a LineName or UserName.

Timeout (integer)
This parameter specifies the time interval (in seconds) for which VaxVoIP waits for the outgoing call to connect. If the outgoing call does not connect within the specified time, a timeout occurs, and the OnCallSessionTimeout() event is triggered.

Return Value

If the function succeeds, it returns a new SessionId. If the function fails, it returns -1. To obtain extended error information, call the GetVaxErrorCode() method.

Example

```
SessionId = DialCallSession("8025", "8025", "8034", "LineDial", 20)

If (SessionId = -1) GetVaxErrorCode()
```

See Also

CloseCallSession(), OnCallSessionCreated(), OnCallSessionTimeout(),
OnCallSessionConnecting(), OnCallSessionClosed()

AccessAudioPCM()

AccessAudioPCM() provides real-time access to raw audio PCM data. There are two methods for accessing this data during a VoIP call: through events or through named pipes.

This capability is particularly useful for applications integrated with VaxVoIP, such as IVR systems, IP-PBX, and SIP servers, where real-time audio processing, AI and machine learning tasks, speech recognition, language translation, text-to-speech, and speech-to-text operations are performed.

The supported raw digital audio PCM format is uncompressed 8kHz, 16bit, and Mono.

Named-Pipe Method: A named pipe facilitates data communication between a pipe server and a pipe client. Named pipes can be used for communication both between processes on the same computer and across different computers over a network. For more details, refer to the named-pipe documentation on the Microsoft MSDN website.

In this method, VaxVoIP internally creates a Pipe-Server using the name provided in the PipeName parameter. The application then creates a Pipe-Client and connects it to the Pipe-Server. Once connected, the Pipe-Server begins delivering PCM audio data to the Pipe-Client. This method is recommended for large-scale audio processing in multi-threaded environments, such as IP-PBX and SIP servers managing a high volume of VoIP calls.

Event-Driven Method: The event-based method offers a straightforward way to access audio PCM data. By activating event-based audio access for a specific channel of a SessionId, VaxVoIP triggers the OnCallSessionAccessAudioPCM() event, passing the PCM audio data to the application via the main thread. However, extensive processing on the main thread can lead to delays and affect the responsiveness of other events. Therefore, this method is best suited for scenarios involving a smaller number of calls and less extensive audio processing.

AI and Machine Learning Applications: With access to raw PCM data, applications can leverage AI and machine learning for advanced audio analytics. This includes training AI models for tasks such as speech recognition (converting spoken language into text), language translation (translating spoken language into another language), and text-to-speech (generating spoken language from text). Additionally, AI can enhance voice quality, detect anomalies, and provide personalized responses in real-time. This integration supports a wide range of applications, from improving customer interactions in IVR systems to enabling real-time language translation and adaptive learning systems.

Please refer to the [ACCESS AUDIO DATA PCM](#) section for further details.

Syntax

```
boolean AccessAudioPCM(  
    SessionId,  
    ChannelId,  
    PipeName,  
    AccessType  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

PipeName (string)

The value of this parameter specifies the name of Pipe.

AccessType (integer)

The value of this parameter specifies the access type.

-1 = Access Type All None

0 = Access Type Event None

1 = Access Type Event Inbound

2 = Access Type Event Outbound

3 = Access Type Pipe None

4 = Access Type Pipe Inbound

5 = Access Type Pipe Outbound

6 = Access Type Duplex Pipe Inbound

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the `GetVaxErrorCode()` method.

Example

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

OnCallSessionAccessAudioPCM(SessionId, ChannelId, AccessType, DataPCM,
                             SizePCM, TypePCM)
{
    // Access the DataPCM array and process it for audio analysis.
}
```

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
               FromPeerName, RouteUID, RouteType, RouteName,
               UserAgentName, FromIP, FromPort)
{
    PipeName = "\\.\pipe\PipeServer"+ ConvertToText(SessionId)

    AccessAudioPCM(SessionId, 0, PipeName, 4)
    AcceptCallSession(SessionId, "", "", "", DialNo, DialNo, 20)

    // MSDN Microsoft Help: Create PipeClient and
    // Connect to PipeServer (PipeName)
}
```

See Also

OnCallSessionAccessAudioPCM(), SendAudioPCM()

SendAudioPCM()

The SendAudioPCM() method is designed to transmit raw PCM (Pulse Code Modulation) audio data to a specific VoIP call. This functionality is particularly useful for developing features such as Text-to-Speech (TTS), where textual input is converted into audible speech.

In addition, SendAudioPCM() facilitates seamless communication between VaxVoIP-based IP-PBX systems and various telephony hardware interfaces. This includes integration with PSTN (Public Switched Telephone Network) and GSM (Global System for Mobile Communications) telephony cards, TAPI (Telephony Application Programming Interface) devices, and other telephony hardware.

Audio Format: The method supports raw digital audio in PCM format, specifically uncompressed audio with a sample rate of 8kHz, 16bit depth, and mono channel configuration. This ensures high-quality audio transmission for various applications.

Text-to-Speech Development: By sending PCM audio data directly to a VoIP call, developers can integrate advanced text-to-speech features, allowing for real-time conversion of text into spoken audio.

Telephony Integration: The method enables communication between VaxVoIP-based systems and traditional telephony hardware, making it easier to bridge modern VoIP solutions with legacy telephony infrastructures.

Overall, SendAudioPCM() is a versatile tool for both advanced audio processing and integrating VoIP systems with traditional telephony equipment.

Syntax

```
boolean SendAudioPCM(  
    SessionId,  
    ChannelId,  
    DataPCM,  
    SizePCM,  
    TypePCM  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

DataPCM (array)

This parameter contains the audio PCM data.

SizePCM (integer)

This parameter specifies the size of the audio PCM data in bytes.

TypePCM (integer)

This parameter is reserved and should be set to 0.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the `GetVaxErrorCode()` method.

Example

```
SendAudioPCM(SessionId, 0, aDataPCM, 16000, 0);
```

See Also

`OnCallSessionAccessAudioPCM()`, `AccessAudioPCM()`

AcceptTransferBlind()

The AcceptTransferBlind() function processes an incoming call transfer request from a specific user, enabling the feature of "unannounced" or "blind" call transfer. This feature allows a call to be transferred without notifying the recipient or party about the impending transfer.

When the VaxVoIP based server receives a blind transfer request from a user, it triggers the OnCallSessionTransferBlind() event. This event, in turn, invokes the AcceptTransferBlind() function to handle the transfer request.

The AcceptTransferBlind() function generates an outgoing call to the target specified by the TransferTo parameter and assigns this call to a free or available channel within the transferring call session.

Syntax

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

Parameters

- TransfererSessionId (integer)
This parameter specifies a unique identifier for the Transferer Call-Session.
- CallerName (string)
This parameter specifies the name of the caller.
- CallerId (string)
This parameter specifies the caller ID.
- TransferTo (string)
This parameter specifies the number to be dialed for the call transfer.
- ToPeerName (string)
This parameter specifies the name of the peer to which the call is being transferred.
- RouteUID (string)
This parameter specifies the unique identifier assigned to the route.

Timeout (integer)

This parameter specifies the time interval, in seconds that VaxVoIP waits for the transferred call to connect. If the transferred call does not connect within this specified time, a timeout occurs, and the OnCallSessionTransferTimeout() event is triggered.

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", "LineDial", "")  
}
```

See Also

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

AcceptTransferConsult()

The AcceptTransferConsult() function handles an incoming transfer call request from a specific user, enabling the "announced" or "consult" call transfer feature. This feature allows for notifying the desired party or extension of the impending call by putting the caller on hold while dialing the desired party or extension.

When the server receives a consult transfer request from a user, it triggers the OnCallSessionTransferConsult() event. This event then invokes the AcceptTransferConsult() function to process the request accordingly.

Upon invocation, AcceptTransferConsult() performs the following actions:

1. Removes the Transferee call from the Transferer call session.
2. Disconnects the Transferer calls from both the Transferer and TransferTo call sessions.
3. Closes the Transferer call session.
4. Moves the Transferee call to an available or free channel within the TransferTo call session.

Syntax

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

Parameters

TransfererSessionId (integer)

This parameter specifies a unique identifier for the Transferer Call-Session.

TransferToSessionId (integer)

This parameter specifies a unique identifier for the 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 handles and rejects an incoming transfer call request. This function is used to deny or cancel the transfer request, ensuring that the call is not processed further.

Syntax

```
RejectTransfer (SessionId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Return Value

No return value.

Example

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

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

See Also

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

LoadWaveFile()

The LoadWaveFile() function loads wave (.wav) data into memory and returns a unique identifier for the loaded wave data, known as WaveId. This WaveId can then be used with other functions to manage and play the wave data in a call session:

- PlayWaveStartToCallSession(SessionId, ChannelId, WaveId, Loop, PlayPos)
- PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
- UnLoadWaveID(WaveId)

To optimize performance and avoid media stream type and format conversion, LoadWaveFile() only supports uncompressed wave files (.wav) recorded in the format of 8KHz, 16bit, Mono.

Other media file types, such as MP3 or GSM, are not supported. If necessary, third-party libraries can be used to convert files from formats like MP3 to .wav before using them with the LoadWaveFile() function.

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

```
SetLicenseKey("LicenseKey")

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

Result = OpenNetworkUDP("", 5060)
if(Result = 0) GetVaxErrorCode()

MusicWaveId = LoadWaveFile("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, 0)
}
```

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

    MusicWaveId = LoadWaveFile("MusicIVR.wav")
    if(MusicWaveId = -1) GetVaxErrorCode()

    PlayWaveStartToCallSession(SessionId, 0, MusicWaveId, 0, 0)

    //Unloads wave data after playback or when call disconnects

    UnLoadWaveID(MusicWaveId);
}

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

See Also

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

LoadWavePCM()

The LoadWavePCM() function allows you to pass voice PCM data to VaxVoIP and returns a unique play wave identification (WaveId). This WaveId can be used with other functions to play the loaded wave data in a call session.

- PlayWaveStartToCallSession(SessionId, ChannelId, WaveId, Loop, PlayPos)
- PlayWaveStopToCallSession(SessionId, ChannelId, WaveId)
- UnLoadWaveID(WaveId)

To minimize media stream type and format conversions and conserve CPU resources, LoadWavePCM() supports only uncompressed PCM format (8KHz, 16bit, Mono).

You can use LoadWavePCM() to integrate third-party Text-To-Speech (TTS) engines. These engines generate PCM audio data at 8KHz, 16bit, Mono, which can be passed to VaxVoIP using LoadWavePCM(). For TTS engine details, contact the TTS vendor.

LoadWavePCM() can also be used with third-party libraries to convert MP3 and other audio formats to PCM data (8KHz, 16bit, Mono). For this, use a third-party library to convert compressed audio to PCM format and pass it to VaxVoIP. Contact the third-party vendor for further assistance.

Syntax

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

Parameters

DataPCM (array)

This parameter specifies the voice PCM data to be loaded into VaxVoIP.

SizePCM (integer)

This parameter specifies the size of the voice PCM data in bytes.

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)
    if(WaveId = -1) GetVaxErrorCode()

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

    //Unloads wave data after playback or when call disconnects

    UnLoadWaveID(WaveId);
}

OnCallSessionPlayWaveDone(SessionId, ChannelId, WaveId)
{
}

}
```

See Also

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

UnLoadWaveID()

The UnLoadWaveID() function is used to unload and free the memory associated with a specific Wave-Id voice data. When you no longer need to use the voice data identified by a particular Wave-Id, calling this function releases the resources allocated for that Wave-Id.

This helps to manage memory efficiently and ensures that the system does not hold onto unused audio data.

The UnLoadWaveID() function is particularly useful for cleaning up after audio playback or when dynamic loading and unloading of audio data are required. It is an essential part of maintaining optimal performance and resource management in applications that involve audio processing.

Syntax

```
UnLoadWaveID(WaveId)
```

Parameters

WaveId (integer)

This parameter specifies the unique identifier for the voice data to be played. Each WaveId corresponds to a specific set of audio data that has been loaded into memory.

Return Value

No return value.

Example

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

    MusicWaveId = LoadWaveFile("MusicIVR.wav")
    if(MusicWaveId = -1) GetVaxErrorCode()

    PlayWaveStartToCallSession(SessionId, 0, MusicWaveId, 0, 0)

    //Unloads wave data after playback or when call disconnects

    UnLoadWaveID(MusicWaveId);
}
```

See Also

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

PlayWaveStartToCallSession()

The PlayWaveStartToCallSession() function initiates the playback of a wave file (.wav) data to a specific call within a Call-Session.

VaxVoIP SDK utilizes buffered-based compression technology to optimize CPU usage and minimize processing load.

When playing wave file data, VaxVoIP encodes the wave data into a voice codec only once. This encoded data is then cached, allowing it to be played repeatedly without re-encoding. This approach significantly reduces the CPU load and enhances performance during playback.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

WaveId (integer)

This parameter specifies the unique identification of the 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 specifies the playback position of the wave file. It can indicate either the start or the middle of the wave data.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

GreetingWaveId = LoadWaveFile("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() function stops the playback of a specific wave file (.wav) for a particular call within a call session.

This function halts the audio output associated with the specified WaveId on the designated SessionId and ChannelId.

It ensures that the wave data ceases immediately, and the call session resumes normal audio operations without the interruption of the previously played wave data.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

WaveId (integer)

This parameter specifies the unique identification of the wave data to be played. If set to -1, it stops any currently playing wave data in the call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
GreetingWaveId = LoadWaveFile("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()

PlayWaveSetVolumeToCallSession() adjusts the volume of the wave data currently playing in a specific call within a call session.

This function allows you to control the audio level of the wave data being played, providing the ability to tailor the audio experience according to the needs of the call.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for the call session.

ChannelId (integer)

This parameter identifies the call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

WaveId (integer)

This parameter specifies the unique identifier for the wave data to be played.

Volume (integer)

This parameter specifies the volume level for the wave data.

-100 = Silence

0 = No Change

+100 = Maximum Volume

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
WaveId = LoadWaveFile("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, 0)
    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()

VaxVoIP supports playing multiple wave files on a specific call within a call-session. The PlayWaveSetModeTypeToCallSession() function determines how the wave data should be played, either mixed with other audio or played exclusively, overriding any other audio.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session:

0 = Channel-ZERO call

1 = Channel-ONE call

PlayModeType (integer)

This parameter specifies the mode type for playing wave data:

0 = Wave Mode Exclusive

1 = Wave Mode Normal

In Exclusive mode, the wave file plays as an announcement to the agent, such as in a call center scenario, overriding any other audio.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
WaveId = LoadWaveFile("Music.wav")
if (WaveId = -1) GetVaxErrorCode()

WaveIdA = LoadWaveFile("Greeting1.wav")
if (WaveIdA = -1) GetVaxErrorCode()

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

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

    PlayWaveSetModeTypeToCallSession(SessionId, 0, 0)

    PlayWaveStartToCallSession(SessionId, 0, WaveIdA, 1, 0)
    PlayWaveStartToCallSession(SessionId, 0, WaveIdB, 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#")
    {
        PlayWaveSetModeTypeToCallSession(SessionId, 0, 1)
        PlayWaveStartToCallSession(SessionId, 0, WaveId, 1, 0)
    }
}
```

See Also

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

RecordWaveSetCacheSize()

The RecordWaveSetCacheSize() function informs the VaxVoIP library (DLL) about the amount of memory to allocate for buffering recorded audio data before it is written to the storage file.

This setting helps optimize the performance of audio recording by ensuring that the library has sufficient memory to handle the data efficiently during the recording process.

Syntax

```
boolean RecordWaveStartToCallSession(CacheSize)
```

Parameters

CacheSize (integer)

This parameter specifies the number of bytes allocated for the cache. The default value is -1, which means there is no cache size limit. In this case, all recorded data is stored in memory (RAM) and is only written to the file when the call disconnects or the recording stops.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
RecordWaveSetCacheSize(1000000) // 1MB cache size

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

```
RecordWaveSetCacheSize(-1) // No-limit cache size

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

See Also

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

RecordWaveStartToCallSession()

The RecordWaveStartToCallSession() function begins recording the audio stream of a specific call within a Call-Session and saves it to a wave file (.wav).

If the FileName parameter is provided as an empty string, the RecordWaveStartToCallSession() function records the audio data directly into memory (RAM) instead of saving it as a file. When the call disconnects or the RecordWaveStopToCallSession() method is invoked, the recording stops. At this point, the OnCallSessionRecordedWaveREC() event is triggered, which provides the recorded wave file data as an array for further processing or analysis.

If the call disconnects or closes without explicitly stopping the recording using the RecordWaveStopToCallSession() method, VaxVoIP automatically stops the recording. It will then create a wave file using the specified FileName value, or if no FileName is provided, it will trigger the OnCallSessionRecordedWaveREC() event and pass the recorded wave file data as an array to the VaxVoIP-integrated application.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

FileName (string)

This parameter specifies the name of the file where the audio will be recorded. If an empty string is provided, the audio data is stored in memory (RAM) instead of being saved to a file.

TypeREC (integer)

This parameter specifies the format and type of the recorded wave.
The following values are supported:

- 0 = REC TYPE MONO PCM
- 1 = REC TYPE STEREO PCM
- 2 = REC TYPE MONO G711U
- 3 = REC TYPE STEREO G711U
- 4 = REC TYPE MONO G711A
- 5 = REC TYPE STEREO G711A

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

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

```
OnCallSessionConnected(SessionId)
{
    RecordWaveStartToCallSession(SessionId, 0, "", 2)
}

OnCallSessionRecordedWaveREC(SessionId, ChannelId, DataREC, SizeREC,
                              TypeREC)
{
    // DataREC represents a WAV file with header and audio data.
    // Save DataREC to a file with any desired name and .wav extension.
}
```

See Also

`RecordWaveSetCacheSize()`, `OnCallSessionRecordedWaveREC()`,
`RecordWaveStopToCallSession()`, `RecordWavePauseToCallSession()`,
`DialCallSession()`, `GetVaxErrorCode()`

RecordWaveStopToCallSession()

The RecordWaveStopToCallSession() function stops the recording of audio from a specific call within a call-session. Although this function can be used to manually stop the recording, it is not necessary to use it to create the wave file. When a call ends or disconnects, VaxVoIP automatically completes and saves the recorded wave file, even if RecordWaveStopToCallSession() is not called.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

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

RecordWavePauseToCallSession()

The RecordWavePauseToCallSession() function pauses the recording of audio for a specific call within a call-session. This allows for temporary suspension of the recording process, which can be resumed later or stopped completely. The paused recording is saved in memory and can be continued or finalized as needed

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Enable (Boolean)

This parameter specifies whether to pause or resume the recording process. Assign a value of 1 to pause the recording, or 0 to resume it.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a wave file that will be used as music on hold in the system.

This function enables the integration of custom audio for holding calls, allowing callers to listen to music or announcements while they are on hold.

The file specified by the FileName parameter is loaded into the system, making it available for playback during hold periods.

Syntax

```
boolean LoadMusicHold(FileName)
```

Parameters

FileName (string)

This parameter value specifies the name of the wave file to be used for music on hold.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
LoadMusicHold("HoldMusic.wav")
```

See Also

UnLoadMusicHold(), GetVaxErrorCode()

UnLoadMusicHold()

The UnLoadMusicHold() function unloads the wave file that was previously set for music on hold. This function removes the specified music file from the system, stopping any playback of the hold music and freeing up resources.

Syntax

```
UnLoadMusicHold()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
UnLoadMusicHold()
```

See Also

LoadMusicHold(), GetVaxErrorCode()

AcceptOnHoldRequest()

The `AcceptOnHoldRequest()` function is used to accept an on-hold request for a specific call session.

When VaxVoIP receives a hold request, it triggers the `OnCallSessionOnHold()` event. Within the event handler for this event, calling `AcceptOnHoldRequest()` confirms and accepts the on-hold request, thereby placing the call session on hold as requested.

Syntax

```
boolean AcceptOnHoldRequest(SessionId)
```

Parameters

`SessionId` (integer)

This parameter specifies a unique identifier for a call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

See Also

`AcceptOffHoldRequest()`, `OnCallSessionOnHold()`, `GetVaxErrorCode()`

AcceptOffHoldRequest()

The AcceptOffHoldRequest() function is used to accept an off-hold request for a specific call session.

When VaxVoIP receives an off-hold request, it triggers the OnCallSessionOffHold() event. In response to this event, AcceptOffHoldRequest() is called to confirm and accept the off-hold request, thereby resuming the call session from its held state

Syntax

```
boolean AcceptOffHoldRequest(SessionId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

RejectChatMessage(), GetVaxErrorCode()

RejectChatMessage()

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

VaxVoIP 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

AcceptChatMessage(), 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.

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

Syntax

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

Parameters

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

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

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

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

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

AcceptChatStatusSubscribe(), OnChatStatusSubscribe(), GetVaxErrorCode()

RejectChatStatusSubscribe()

The RejectChatStatusSubscribe() rejects the chat status subscribe request.

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

Syntax

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

Parameters

SubscribId (integer)

This parameter value specifies a unique identification of status subscription.

MsgFrom (string)

This parameter specifies the From user.

MsgTo (string)

The value of this parameter specifies the To user.

Return Value

No return value.

Example

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

See Also

AcceptChatStatusSubscribe, OnChatStatusSubscribe, GetVaxErrorCode()

OpenConferenceRoom()

The OpenConferenceRoom() function establishes a conference room, allowing multiple call sessions to be added to it.

Once added, participants within these call sessions can communicate with one another, with each session granted the ability to speak and listen within the conference.

This function facilitates multi-party communication, making it ideal for meetings, group discussions, and collaborative activities.

Syntax

```
boolean OpenConferenceRoom(RoomName)
```

Parameters

RoomName (string)

This parameter specifies the name of the conference room.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
OpenConferenceRoom("TechRoom")
```

See Also

CloseConferenceRoom(), GetVaxErrorCode()

CloseConferenceRoom()

The CloseConferenceRoom() function closes an existing conference room.

Syntax

```
CloseConferenceRoom(RoomName)
```

Parameters

RoomName (string)

This parameter specifies the name of a conference room.

Return Value

No return value.

Example

```
CloseConferenceRoom("TechRoom")
```

See Also

OpenConferenceRoom(), GetVaxErrorCode()

AddCallSessionToConferenceRoom()

The AddCallSessionToConferenceRoom() method integrates a Call-Session into a conference room, enabling the participants of that session to engage in the conference by both listening and speaking.

This functionality facilitates real-time communication among multiple users within the same conference environment.

Syntax

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

Parameters

RoomName (string)

This parameter specifies the name of conference room.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

RemoveCallSessionFromConferenceRoom(), GetVaxErrorCode()

RemoveCallSessionFromConferenceRoom()

The RemoveCallSessionFromConferenceRoom() method removes a specified call session from the conference room. It updates the conference room's participant list, ensuring the call session is no longer part of the conference. The call session may remain active if connected to other sessions or calls.

Syntax

```
RemoveCallSessionFromConferenceRoom(SessionId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

PlayWaveStopToConferenceRoom(), GetVaxErrorCode()

PlayWaveStopToConferenceRoom()

The PlayWaveStopToConferenceRoom() stops particular 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
PayWaveStopToConferenceRoom("TechRoom", WaveId)
```

See Also

PlayWaveStartToConferenceRoom(), GetVaxErrorCode()

PlayWaveSetVolumeToConferenceRoom()

The PlayWaveSetVolumeToConferenceRoom() method adjusts the volume level of a specific wave file currently being played in a conference room. It takes as input the volume setting and applies it to the audio playback for the targeted wave data.

The volume adjustment affects all participants in the conference room, ensuring a consistent listening experience.

Syntax

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

Parameters

RoomName (string)

This parameter specifies the name of conference room.

WaveId (integer)

This parameter specifies the unique identifier for the wave data to be played.

Volume (integer)

This parameter specifies the volume level for the wave data.

-100 = Silence

0 = No Change

+100 = Maximum Volume

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

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

RecordWaveStartToConferenceRoom()

The RecordWaveStartToConferenceRoom() function starts recording the voice stream from a specific conference room, with the recorded audio data being saved as a wave file (.wav).

For recording audio data directly into memory (RAM) instead of saving to a file, the RecordWaveStartToCallSession() method should be used with an empty string for the FileName parameter. This method begins recording and stores the audio data in memory.

Upon stopping the recording either by disconnecting the call or invoking the RecordWaveStopToConferenceRoom() method the OnConferenceRoomRecordedWaveREC() event is triggered. This event provides the recorded wave file data as an array, which can be used for further processing or analysis of the conference room audio.

Syntax

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

Parameters

RoomName (string)

This parameter specifies the name of conference room.

FileName (string)

This parameter designates the file name where the audio will be recorded. If an empty string is supplied, the audio data is instead stored directly in memory (RAM) rather than being saved to a file.

TypeREC (integer)

The parameter specifies the type of wave file.

REC TYPE MONO PCM	0
REC TYPE STEREO PCM	1
REC TYPE MONO G711U	2
REC TYPE STEREO G711U	3
REC TYPE MONO G711A	4
REC TYPE STEREO G711A	5

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

```
OpenConferenceRoom("TechRoom")
RecordWaveStartToConferenceRoom("TechRoom", "Record.wav", 0)

OnCallSessionConnected(SessionId)
{
    AddCallSessionToConferenceRoom("TechRoom", SessionId)
}

RecordWaveStopToConferenceRoom("TechRoom")
```

```
OpenConferenceRoom("TechRoom")
RecordWaveStartToConferenceRoom("TechRoom", "", 4)

OnCallSessionConnected(SessionId)
{
    AddCallSessionToConferenceRoom("TechRoom", SessionId)
}

RecordWaveStopToConferenceRoom("TechRoom")

OnConferenceRoomRecordedWaveREC(RoomName, DataREC, SizeREC,
                                TypeREC)
{
    // DataREC is a WAV file containing the header & audio data.
    // Write the DataREC to a file with any name and a .wav extension.
}
```

See Also

`OnConferenceRoomRecordedWaveREC()`, `RecordWaveSetCacheSize()`,
`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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
RecordWaveStopToConferenceRoom("TechRoom")
```

See Also

RecordWaveSetCacheSize(), 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
RecordWavePauseToConferenceRoom( " TechRoom", 1)
```

See Also

RecordWaveSetCacheSize(), RecordWaveStartToConferenceRoom(),
RecordWaveStopToConferenceRoom(), GetVaxErrorCode()

AudioSessionLost()

The AudioSessionLost() method detects when an audio session is lost. VaxVoIP continuously monitors for incoming audio packets within a specified time interval. If no audio packets are received during this interval, VaxVoIP triggers the OnCallSessionLost() event, indicating that the audio session has been disrupted.

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 (seconds) for detection of voice data.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
AudioSessionLost(1, 20)

//Enabled audio session loss detection with a 20-second interval
```

See Also

OnCallSessionLost(), GetVaxErrorCode()

SendInfoVM()

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

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

Syntax

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

Parameters

UserName (string)

This parameter specifies the user's login name.

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.

VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 identifier for 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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.
}
```

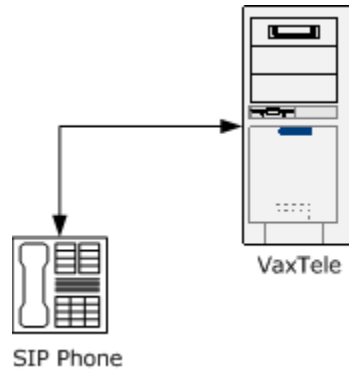
- Enables RTP based detection at Channel-ZERO call for two seconds time-out.
- VaxVoIP receives first digit. (e.g. 4)
- VaxVoIP waits for other digits with time-out interval.
- With two seconds interval, remote person press another digit (e.g. 8)
- VaxVoIP receives another digit. (e.g. 8)
- Waits for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

There are total three types of DTMF digit detection standards in IP-Telephony. VaxVoIP 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)
This parameter identifies a call within 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `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 specifies a unique identifier for 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

OnCallSessionDialToneStarted(), OnCallSessionDialToneEnded()

SplitCallSession()

The SplitCallSession() function removes a specific 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 specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 specific 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

GreetingWaveId = LoadWaveFile("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, VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP to inbound voice stream of a specific call in a call-session.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

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

See Also

OnIncomingCall(), GetVaxErrorCode()

BusyLampSubscribeAccept()

The BusyLampSubscribeAccept() function accepts incoming BLF subscribe request. VaxVoIP 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, VaxVoIP completes login authorization process then accepts the BLF request.

If Value is 0, VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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. VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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. VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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.

VaxVoIP completes the detection and triggers OnCallSessionDetectAMD() event.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 specific 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 identifies a call within 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 identifies a call within 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 VaxVoIP 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the `GetVaxErrorCode()` method.

Example

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

See Also

`MuteCallServerREC()`, `OnServerConnectingREC()`, `GetVaxErrorCode()`,
`OnServerHungupREC()`

MuteCallServerREC()

The MuteCallServerREC() function mutes the audio stream being sent to the SIP Recording (REC) Server. When invoked, this function prevents any audio data from being transmitted to the REC server, effectively silencing the recorded audio for the duration of the mute.

Syntax

```
boolean MuteCallServerREC(SessionId, ChannelId, MuteType)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

MuteType (integer)

This parameter specifies the mute types.

REC_MUTE_TYPE_NONE	-1
REC_MUTE_TYPE_SPEAK	0
REC_MUTE_TYPE_LISTEN	1
REC_MUTE_TYPE_BOTH	2

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

OnDetectedDigitDTMF(SessionId, CallerName, DigitDTMF, TypeDTMF)
{
    if(DigitDTMF = "#21#")
        MuteCallServerREC(SessionId, 0, 1)

    if(DigitDTMF = "#22#")
        MuteCallServerREC(SessionId, 0, -1)
}
```

See Also

ConnectToServerREC(), OnServerConnectingREC(), GetVaxErrorCode(),
OnServerHungupREC()

SetExtDataREC()

The SetExtDataREC() adds extra recording data into the SIP packet.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

ExtData (string)

This parameter value put data in ex data rec field.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the `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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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 extensions 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 identifier for a call session.

Return Value

On successful execution, this function returns a non-zero value. If it fails, it returns 0. For details on the specific error, call the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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(GroupNames, UserName)
```

Parameters

GroupNames (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 identifier for a call session.

UserName (string)

The value of this parameter specifies the user name.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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 is designed to park a call session at a specified unique slot number, effectively holding the session in a temporary state.

This functionality is crucial for implementing call parking features within a SIP environment, where calls can be parked and later retrieved from specific slots. By assigning a unique slot number to each parked session, the system ensures that calls are organized and easily accessible.

The ParkCallSession() function can be integrated with other call parking-related functions, allowing developers to create a comprehensive call parking system that supports seamless call management and retrieval. This enhances the flexibility and efficiency of call handling in various telephony applications

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 allows a call session to be connected to a call that was previously parked at a specified slotId using the ParkCallSession() function.

This function is essential for retrieving parked calls, enabling the seamless continuation of communication by linking the current call session to the parked call. By utilizing the slotId, the function ensures that the correct parked call is connected, facilitating efficient call management within the system.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

SlotId (integer)

The value of this parameter specifies unique slot number.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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 identifier for a call session.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the 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.

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

Syntax

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

Parameters

StealthSessionId (integer)

This parameter specifies a unique identifier for a call session.

CapturePeerName (string)

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

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

```
AddUser("0000", "123", "01") // Set user 0000 as an admin in your database
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() function modifies the voice listening type of a call within a call session, conference room, or for stealth listening purposes.

This function, along with AddStealthListener(), can be utilized to develop agent training functionality in a call center environment.

Voice Type Admin: An admin-type participant can listen to user-type, normal-type, and other admin-type participants.

Voice Type User: A user-type participant can listen to admin-type, normal-type, and other user-type participants.

Voice Type Normal: A normal-type participant can listen to user-type and other normal-type participants.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

VoiceTypeId (integer)

This parameter specifies the voice type of the participant.

0 = Normal Voice Type

1 = User Voice Type

2 = Admin Voice Type

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
               FromPeerName, UserAgentName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", "", "")
    AddStealthListener(SessionId, "4040", 2)

    // Retrieve the AgentSessionId from the database or list.
    // Set the agent's voice type to 'user'.

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

See Also

AddStealthListener(), AddCallSessionToConferenceRoom(),
AdjustCallSessionVideoType()

AdjustCallSessionVideoType()

AdjustCallSessionVideoType() configures the video reception settings for a call within a call session, conference room, or during stealth-listening functionality.

When AdjustCallSessionVideoType() is configured with the voice type set to "admin" for a call in a conference room, it enables the admin-type participant to receive video frames from user-type participants. This setting ensures that the admin has visibility of all video feeds from participants identified as users within the conference room.

Video Type Admin: In a conference room, an admin-type participant receives copies of video frames from user-type participants.

Video Type User: In a conference room, a user-type participant forwards their video frames to admin-type participants.

Video Type Normal: A normal-type participant does not receive video frames from any participant.

Syntax

```
boolean AdjustCallSessionVideoType(SessionId, ChannelId, VideoTypeId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

VideoTypeId (integer)

This parameter value specifies the video type of the participant.

0 = Normal Video Type

1 = User Video Type

2 = Admin Video Type

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

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

    if (SessionId = TypeAdmin)
    {
        AdjustCallSessionVideoType(SessionId, 0, 2)
    }
}
```

See Also

AddStealthListener(), AddCallSessionToConferenceRoom(),
AdjustCallSessionVoiceType()

CallSessionAttachCall()

The CallSessionAttachCall() function attaches two call sessions. After the attachment, the attached session begins receiving media (audio and video) streams from the attachTo call session.

This function also supports specific voice and video streaming types, determining how the media is streamed from the attachTo call session.

It can be utilized to develop agent training functionality in a call center environment, enabling supervisors to monitor or participate in ongoing calls for training purposes.

Syntax

```
boolean CallSessionAttachCall(  
    SessionId,  
    VoiceTypeId,  
    VideoTypeId,  
    AttachToSessionId,  
    AttachToChannelId  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session. In a typical call center scenario, it represents the administrator's call session.

VoiceTypeId (integer)

This parameter specifies the voice type of the participant.

- 1 = Keep Existing Voice Type
- 0 = Normal Voice Type
- 1 = User Voice Type
- 2 = Admin Voice Type

Voice Type Normal: A participant with this voice type can listen to user-type and other normal-type participants.

Voice Type User: A user-type participant can listen to admin-type, normal-type, and other user-type participants.

Voice Type Admin: An admin-type participant can listen to user-type, normal-type, and other admin-type participants.

VideoTypeId (integer)

This parameter specifies the video type of the participant.

-1 = Keep Existing Video Type

0 = Normal Video Type

1 = User Video Type

2 = Admin Video Type

Video Type Normal: A normal-type participant does not receive video frames from any participant.

Video Type User: A user-type participant forwards their video frames to admin-type participants.

Video Type Admin: An admin-type participant receives copies of video frames from user-type participants.

AttachToSessionId (integer)

This parameter specifies a unique identifier for a call session. In a typical call center scenario, it represents the agent's call session.

AttachToChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Return Value

When executed successfully, this function returns a non-zero value. If it fails, it will return zero. In case of an error, you can obtain a specific error code by invoking the `GetVaxErrorCode()` method.

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
               FromPeerName, RouteUID, RouteType, RouteName,
               UserAgentName, FromIP, FromPort)
{
    if(DialNo == "#22#")
    {
        AdjustCallSessionVoiceType(AgentSessionId, 1, 1)
        CallSessionAttachCall(SessionId, 2, -1, AgentSessionId, 1)
    }
}
```

See Also

`CallSessionDetachCall()`, `AdjustCallSessionVoiceType()`,
`AdjustCallSessionVideoType()`

CallSessionDetachCall()

The CallSessionDetachCall() function detaches a call session from the current session. After detachment, the specified participant will no longer receive media (audio and video) streams from the session.

This function can be useful in various scenarios, such as when a participant needs to exit a call or when managing call sessions in a call center environment.

Syntax

```
boolean CallSessionDetachCall(  
                                SessionId,  
                                VoiceTypeId,  
                                VideoTypeId,  
                                )
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session. In a typical call center scenario, it represents the administrator's call session.

VoiceTypeId (integer)

This parameter specifies the voice type of the attached participant.

-1 = Keep Existing Voice Type

0 = Normal Voice Type

1 = User Voice Type

2 = Admin Voice Type

Voice Type Normal: A participant with this voice type can listen to user-type and other normal-type participants.

Voice Type User: A user-type participant can listen to admin-type, normal-type, and other user-type participants.

Voice Type Admin: An admin-type participant can listen to user-type, normal-type, and other admin-type participants.

VideoTypeId (integer)

This parameter specifies the video type of the attached participant.

-1 = Keep Existing Video Type

0 = Normal Video Type

1 = User Video Type

2 = Admin Video Type

Video Type Normal: A normal-type participant does not receive video frames from any participant.

Video Type User: A user-type participant forwards their video frames to admin-type participants.

Video Type Admin: An admin-type participant receives copies of video frames from user-type participants.

Return Value

No Return Value.

Example

```
OnDetectedDigitDTMF(SessionId, ChannelId, DigitDTMF, TypeDTMF)
{
    if(DigitDTMF = "#22#") // If the admin presses #22#
    {
        CallSessionDetachCall(SessionId, -1, -1)
    }
}
```

See Also

CallSessionAttachCall(), AdjustCallSessionVoiceType(),
AdjustCallSessionVideoType()

ActivatePushSIP()

The ActivatePushSIP() method enables the PushSIP functionality within the VaxVoIP system.

PushSIP Activation: This method triggers the OnCallSessionPushSIP() event for a user who is registered with the VaxVoIP-integrated SIP server. It works by utilizing the PushSIP RFC and including ProviderPN and DevicePN details in the SIP REGISTER request. This setup facilitates push notifications for SIP clients.

WaitSeconds Parameter: The ActivatePushSIP() method includes a WaitSeconds parameter, which specifies the time interval the system should wait for a SIP response from an iOS or Android-based SIP client. If the SIP client does not respond within the specified time, VaxVoIP triggers the OnCallSessionPushSIP() event.

Notification Handling: When the response is not received within the wait time, the OnCallSessionPushSIP() event is fired to the VaxVoIP-integrated SIP server or application. This allows the application to send a notification to the device, ensuring that the user is alerted even if the initial SIP response was delayed or missed.

Syntax

```
boolean ActivatePushSIP(  
    Activate,  
    WaitSeconds  
)
```

Parameters

Activate (boolean)

This parameter can be set to either 1 or 0 (true or false). Assign a value of 1 to enable PushSIP functionality, or set it to 0 to disable it. This controls whether the PushSIP feature is active for handling push notifications.

WaitSeconds (integer)

This parameter defines the duration in seconds to wait for a SIP response from the client. It specifies the time interval during which the system will wait for the SIP client to respond before taking any further action.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

```
SetLicenseKey("TRIAL-LICENSE-KEY")  
  
Initialize("")  
  
OpenNetworkUDP("", 5060)  
  
ActivatePushSIP(true, 2)  
  
// Activates PushSIP functionality with a 2-second wait for response.
```

See Also

[OnRegisterUserPushSIP\(\)](#), [SetUserPushSIP\(\)](#), [OnCallSessionPushSIP\(\)](#)

SetUserPushSIP()

The SetUserPushSIP() method is utilized to import Push SIP (Session Initiation Protocol) information from data storage into the VaxVoIP system.

Once VaxVoIP is initialized, it is essential to invoke this method to load the Push SIP data that was gathered during the OnRegisterUserPushSIP() event and saved.

This step ensures that VaxVoIP is accurately updated with the required Push SIP details, facilitating correct system functionality and efficient user management.

Syntax

```
boolean SetUserPushSIP(  
    UserName,  
    ProviderPN,  
    ProjectPN,  
    DevicePN  
)
```

Parameters

UserName (string)

This parameter specifies the user's login name.

ProviderPN (string)

The ProviderPN parameter identifies the push notification service provider responsible for delivering push notifications. This could be a service like Apple Push Notification Service (APNs) for iOS devices or Firebase Cloud Messaging (FCM) for Android devices.

ProjectPN (string)

This parameter specifies the project unique Name or ID.

DevicePN (string)

This parameter specifies the unique device token used for identifying and sending push notifications.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution fails, it returns zero. In the case of an error, a specific error code can be retrieved by calling the GetVaxErrorCode() method.

Example

```
OnRegisterUserPushSIP(UserName, ProviderPN, ProjectPN, DevicePN)
{
    //Store ProviderPN, ProjectPN, and DevicePN to data storage.
}

+++ VaxVoIP Integrated Server Restarts +++

Initialize("")

OpenNetworkUDP("", 5060)

ActivatePushSIP(true, 2)

// Assume dataStorage is an object that interfaces
// with your data storage system.

PushSIPData data = dataStorage.readDataPushSIP();

// Pass the retrieved PushSIP data to VaxVoIP based ServerSIP.

SetUserPushSIP(data.UserName, data.ProviderPN, data.ProjectPN,
               data.DeviceToken);
```

See Also

OnRegisterUserPushSIP(), ActivatePushSIP(), OnCallSessionPushSIP()

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 specifies the user's login name.

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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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)

The value of this parameter specifies queue name.

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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

```
AddConferenceRoomRouteUID ("TechRoom", "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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the 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 a non-zero value. If it fails, it returns 0. The specific error code can be retrieved by using the GetVaxErrorCode() method.

Example

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

See Also

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

AttackDetectScanSIP()

The AttackDetectScanSIP() method strengthens the security of a VaxVoIP SDK-based SIP server by detecting and preventing unauthorized SIP traffic. This method accepts a domain name as a parameter and monitors incoming SIP packets for that domain by inspecting the FromURI field.

If the domain name provided as a parameter does not match the domain in the FromURI field of an incoming SIP packet, the server identifies the packet as part of a potential attack and triggers the **OnAttackDetectedScanSIP()** event. In response, the SIP server can block the attacker's IP and port using the Windows Defender Firewall APIs.

To mitigate risks, the server discards such packets without sending any SIP response, effectively neutralizing the threat without engaging the attacker.

This mechanism ensures that only legitimate SIP requests associated with the specified domain are processed, reducing the risk of malicious activities such as unauthorized port scanning or domain spoofing. By silently discarding invalid packets, the server also minimizes its exposure to further probing attempts.

Syntax

```
boolean AttackDetectScanSIP(DomainName)
```

Parameters

DomainName (string)
This parameter specifies the domain name.

Return Value

On successful execution, this function returns a non-zero value. If the function fails, it returns 0. To determine the specific error, use the GetVaxErrorCode() method.

Example

```
AttackDetectScanSIP ("demo.vaxvoip.com")
```

On the softphone or SIP client side, configure the domain "demo.vaxvoip.com" as the DomainRealm. The SIP client will then include this domain in its SIP URIs.

See Also

AttackDetectFloodSIP(), AttackDetectBruteForceSIP(),
OnAttackDetectedScanSIP()

AttackDetectFloodSIP()

The AttackDetectFloodSIP() method monitors the number of incoming SIP requests per second to detect potential flooding attacks. If the number of SIP requests per second exceeds the threshold specified as a parameter, the method identifies it as a threat and triggers the **OnAttackDetectedFloodSIP()** event.

Upon detecting such an attack, the SIP server, built using the VaxVoIP SIP SDK, can block the attacker's IP address and port by utilizing the Windows Defender Firewall APIs. This automatic blocking mechanism helps prevent further intrusion or disruption caused by the flood of requests.

By implementing this method, the SIP server enhances its resilience against Denial-of-Service (DoS) attacks and ensures smooth communication for legitimate traffic. The configurable threshold allows administrators to fine-tune the server's sensitivity to flooding attempts based on their specific requirements.

Syntax

```
boolean AttackDetectFloodSIP(ReqRecvLimit)
```

Parameters

ReqRecvLimit (integer)

This parameter specifies the threshold for the maximum number of incoming SIP requests allowed per second. If the number of requests exceeds this limit, the method triggers an attack detection event, signaling a potential flood attack.

Return Value

On successful execution, this function returns a non-zero value. If the function fails, it returns 0. To determine the specific error, use the GetVaxErrorCode() method.

Example

```
Initialize("")  
  
AttackDetectFloodSIP(1000)
```

See Also

AttackDetectScanSIP(), AttackDetectBruteForceSIP(),
OnAttackDetectedFloodSIP()

AttackDetectBruteForceSIP()

The AttackDetectBruteForceSIP() method monitors the authentication failure attempts for a specific interval of time. If attempts increase more than the value provided as first parameter then VaxVoIP SDK detects it as threat and informs application by triggering event **OnAttackDetectedBruteForceSIP()**, application can block the IP and port by using Microsoft's provided Firewall Defender APIs.

Syntax

```
boolean AttackDetectBruteForceSIP(FailureCount, FailureInterval)
```

Parameters

FailureCount (integer)

This parameter specifies the maximum number of authentication failure attempts allowed within the time interval defined by the FailureInterval. If the number of failed attempts exceeds this limit, the method will trigger the brute force attack detection.

FailureInterval: (integer)

This parameter defines the time interval (in seconds) within which the authentication failure attempts are counted. If the number of failed attempts within this interval exceeds the FailureCount, the system detects a potential brute force attack.

Return Value

On successful execution, this function returns a non-zero value. If the function fails, it returns 0. To determine the specific error, use the GetVaxErrorCode() method.

Example

```
Initialize("")  
  
AttackDetectBruteForceSIP(10, 4)
```

See Also

AttackDetectScanSIP(), AttackDetectFloodSIP(),
OnAttackDetectedBruteForceSIP()

EXPORTED EVENTS

OnVaxErrorLog()

VaxVoIP 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("")

// On failure of Initialize(), the OnVaxErrorLog() event is triggered

if (Result = false)
{
    GetVaxErrorCode()
    return
}

OnVaxErrorLog(FuncName, ErrorCode, ErrorMsg)
{
    //Log or store the method name provided in FuncName
    //Log or store the error message provided in ErrorMsg
}
```

See Also

GetVaxErrorCode(), OnCallSessionErrorLog()

OnCallSessionErrorLog()

OnCallSessionErrorLog() is an event that notifies the application when an internal function execution fails. It provides detailed information about errors encountered during call session functions, which helps in diagnosing and troubleshooting issues effectively.

Please see [LIST OF ERROR CODES](#) for more details.

Syntax

```
OnCallSessionErrorLog(SessionId, ChannelId, ErrorCode, ErrorMsg)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

ErrorCode (integer)

Specifies the error code associated with the failure.

ErrorMsg (string)

Provides a textual description of the error message.

Example

```
OnCallSessionErrorLog(SessionId, ChannelId, ErrorCode, ErrorMsg)
{
    //Log or store the error message provided in ErrorMsg
}
```

See Also

GetVaxErrorCode(), OnVaxErrorLog()

OnCallSessionCreated()

The OnCallSessionCreated() event triggers when VaxVoIP 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 VaxVoIP closes a call-session internally.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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()

OnCallSessionAccessAudioPCM()

The OnCallSessionAccessAudioPCM() event is triggered when event-based access to the audio PCM data in real-time of a call within a call session is activated using the AccessAudioPCM() method.

This event-based approach provides a simplified and efficient way to access real-time audio PCM data. When event-based audio access is enabled for a specific channel associated with a SessionId, VaxVoIP automatically triggers the OnCallSessionAccessAudioPCM() event.

During this event, the PCM audio data is passed directly to the application, allowing developers to process or manipulate the audio stream as needed.

This mechanism is particularly useful for applications requiring real-time audio analysis, recording, or other custom audio processing tasks within the VaxVoIP framework.

Syntax

```
OnCallSessionAccessAudioPCM(  
    SessionId,  
    ChannelId,  
    AccessType,  
    DataPCM,  
    SizePCM,  
    TypePCM  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

AccessType (integer)

The value of this parameter specifies the access type.

DataPCM (array)

Contains the raw audio PCM data from the call session.

SizePCM (integer)

Specifies the size of the PCM data in bytes.

TypePCM (integer)

Reserved and set to 0 for future use.

Example

```
OnCallSessionAccessAudioPCM(SessionId, ChannelId, AccessType, DataPCM,  
                             SizePCM, TypePCM)  
{  
    // Access the DataPCM array and process it for audio analysis.  
}
```

See Also

AccessAudioPCM()

OnRegisterUser()

The OnRegisterUser() event is triggered whenever VaxVoIP receives a registration request from any SIP client. This event occurs as part of the SIP registration process, where the SIP client attempts to register its information with the SIP server.

Handling this event allows the system to manage and process incoming registration requests, ensuring that SIP clients are properly authenticated and registered within the VaxVoIP environment

Please see [SIP CLIENT REGISTRATION PROCESS](#) for more details.

Syntax

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

Parameters

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

See Also

OnUnRegisterUser(), AddUser(), RemoveUser()

OnRegisterUserSuccess()

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

Please see [SIP CLIENT REGISTRATION PROCESS](#) for more details.

Syntax

```
OnRegisterUserSuccess(Username)
```

Parameters

Username (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(Username, FromIP, FromPort, RegId)
{
}
```

See Also

OnRegisterUser(), OnRegisterUserFailed()

OnRegisterUserFailed()

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

Syntax

```
OnRegisterUserFailed(Username)
```

Parameters

Username (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(Username, FromIP, FromPort, RegId)
{
    RemoveUser(Username)
}
```

See Also

OnRegisterUser(), OnRegisterUserSuccess(), RemoveUser()

OnRegisterUserPushSIP()

The OnRegisterUserPushSIP() event triggers when VaxVoIP receives register request from any SIP client.

Syntax

```
OnRegisterUserPushSIP(  
    UserName,  
    ProviderPN,  
    ProjectPN,  
    DevicePN  
)
```

Parameters

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

ProviderPN (string)
This parameter specifies the user's login name.

ProjectPN (string)
This parameter specifies the user's login name.

DevicePN (string)
This parameter specifies the user's login name.

Example

```
OnRegisterUserPushSIP(UserName, ProviderPN, ProjectPN, DevicePN)  
{  
  
}
```

See Also

OnUnRegisterUser(), AddUser(), RemoveUser()

OnUnRegisterUser()

VaxVoIP triggers the OnUnRegisterUser() event when it receives an unregister request from any SIP client.

This event occurs when a SIP client requests to deregister from the SIP server, allowing the system to handle and process the removal of the client's registration information from the VaxVoIP environment.

Please see [SIP CLIENT REGISTRATION PROCESS](#) for more details.

Syntax

```
OnUnRegisterUser(Username)
```

Parameters

Username (string)

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

Example

```
OnUnRegisterUser(Username)
{
    RemoveUser(Username)
}
```

See Also

OnRegisterUser(), RemoveUser(), AddUser()

OnLineRegisterTrying()

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

VaxVoIP 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()

VaxVoIP 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()

VaxVoIP 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()

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

To unregister or disconnect VaxVoIP 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()

VaxVoIP 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()

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

VaxVoIP 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()

OnIncomingCall()

The OnIncomingCall() event triggers when VaxVoIP 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 identifier for 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(),
AddQueueRouteUID(), AddUserRouteUID(), AddLineRouteUID(),
AddStealthListenRouteUID()

OnCallSessionConnecting()

The OnCallSessionConnecting() event triggers as soon as call connection process begins. VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP successfully established a Call-Session with SIP client/third party server.

Syntax

```
OnCallSessionConnected(SessionId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

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

See Also

SplitCallSession(), AcceptCallSession(), OnIncomingCall()

OnCallSessionLost()

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

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

Example

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

See Also

OnCallSessionOnHold()

The OnCallSessionOnHold() event triggers, when VaxVoIP 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 identifier for a call session..

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifies a call within 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 VaxVoIP 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 identifies a call within 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 identifies a call within 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()

OnCallSessionTransferring()

The OnCallSessionTransferring() event triggers when a call transfer process initiates and VaxVoIP starts receiving SIP status responses regarding transferring of a call.

Syntax

```
OnCallSessionTransferring(  
    SessionId,  
    TransferType,  
    ChannelId,  
    StatusCode,  
    ReasonPhrase  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for 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 identifies a call within 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

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

See Also

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(),
OnCallSessionTransferred(), RejectTransfer()

OnCallSessionTransferFailed()

The OnCallSessionTransferFailed() event triggers when VaxVoIP receives 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 identifier for 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 identifies a call within 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 VaxVoIP 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 identifier for 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 identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

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

See Also

OnCallSessionTransferBlind(), OnCallSessionTransferConsult(),
RejectTransfer()

OnCallSessionTransferred()

The OnCallSessionTransferred() event triggers when VaxVoIP successfully transfers a call.

Syntax

```
OnCallSessionTransferred(  
    SessionId,  
    TransferType  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for 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

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

See Also

OnCallSessionTransferBlind(), AcceptTransferBlind(), AcceptTransferConsult(),
OnCallSessionTransferring()

OnSendReqTransferCallTimeout()

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

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP receives an error response.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 VaxVoIP 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 VaxVoIP 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 VaxVoIP 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 VaxVoIP starts playing the dial tone.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 OnCallSessionDialToneStop() event triggers when VaxVoIP stops playing the dial tone.

Syntax

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

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 *From IP* address.

FromPort (integer)

This parameter specifies the *From port* number.

Example

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

See Also

AcceptChatStatusSubscribe(), RejectChatStatusSubscribe()

OnChatStatusSubscribe()

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

Syntax

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

Parameters

SubscribId (integer)

This parameter value specifies a unique identification of status subscription.

MsgFrom (string)

This parameter specifies the *From user*.

MsgTo (string)

This parameter specifies the *To user*.

FromPeerType (integer)

This parameter value specifies the type of From-Peer.

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

See Also

AcceptChatStatusSubscribe(), RejectChatStatusSubscribe()

OnChatMessageSuccess()

The OnChatMessageSuccess() event triggers when VaxVoIP 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 VaxVoIP 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, Forbidden etc).

Example

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

See Also

OnChatMessageSuccess(), AcceptChatMessage(), RejectChatMessage()

OnChatMessageTimeout()

The OnChatMessageTimeout() event triggers when VaxVoIP 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 VaxVoIP receives BLF subscribe request.

Syntax

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

Parameters

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

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

ToUserName (string)
This parameter value specifies BLF monitor user.

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

FromPort (integer)
This parameter value specifies the from Port number.

Example

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

See Also

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

OnBusyLampUnSubscribe()

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

Syntax

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

Parameters

- SubscribId (integer)
This parameter value specifies a unique subscribe identification.
- UserName (string)
This parameter value specifies BLF subscribe user.
- ToUserName (string)
This parameter value specifies BLF monitor user.
- FromIP (string)
This parameter value specifies the from IP address.
- FromPort (integer)
This parameter value specifies the from Port number.

Example

```
OnBusyLampUnSubscribe(SubscribId, UserName, ToUserName, FromIP,  
    FromPort)  
{  
}
```

See Also

OnBusyLampSubscribe()

OnBusyLampSubscribeSuccess()

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

Syntax

```
OnBusyLampSubscribeSuccess(  
                                SubscribId,  
                                UserName  
                                )
```

Parameters

SubscribId (integer)

This parameter value specifies a unique subscribe identification.

UserName (string)

This parameter value specifies BLF subscribe user.

Example

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

See Also

OnBusyLampSubscribe(), OnBusyLampSubscribeFailed()

OnBusyLampSubscribeFailed()

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

Syntax

```
OnBusyLampSubscribeFailed(  
    SubscribId,  
    UserName  
)
```

Parameters

SubscribId (integer)

This parameter value specifies a unique subscribe identification.

UserName (string)

This parameter value specifies BLF subscribe user.

Example

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

See Also

OnBusyLampSubscribeSuccess(), OnBusyLampSubscribe(),
OnBusyLampUnSubscribe()

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()

OnAddCallSessionToQueueSuccess()

The OnAddCallSessionToQueueSuccess() event notifies application that the process of adding call initiated by using AddCallSessionToQueue() method is completed successfully and caller is waiting for the queue agent to get connected.

Syntax

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

Parameters

QueueName (string)

The value of this parameter specifies queue name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnAddCallSessionToQueueSuccess(QueueName, SessionId)
{
}
}
```

See Also

OnAddCallSessionToQueueFailed(), OnQueuePlayWaveStarted(),
OnQueuePlayWaveEnded(), AddCallSessionToQueue(),
OnQueueAgentConnectStarted(), OnQueueAgentConnectSuccess()

OnAddCallSessionToQueueFailed()

The OnAddCallSessionToQueueFailed() event notifies the application that the process of adding call initiated by using AddCallSessionToQueue() method is failed.

Syntax

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

Parameters

QueueName (string)

The value of this parameter specifies queue name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnAddCallSessionToQueueFailed(QueueName, SessionId)
{
}
}
```

See Also

OnAddCallSessionToQueueSuccess(), OnQueuePlayWaveStarted(),
OnQueuePlayWaveEnded(), AddCallSessionToQueue(),
OnQueueAgentConnectStarted(), OnQueueAgentConnectSuccess()

OnQueuePlayWaveStarted()

The OnQueuePlayWaveStarted() event informs application that which type of music should be played.

It informs the application with event type "PlayMusic" to play music because call is in queue and waiting for the agent. If agent found and starts connecting to agent then such event informs the application with event type "PlayRing" to play ring to the queue call because agent is found and ringing on agent side.

Syntax

```
OnQueuePlayWaveStarted(EventType, QueueName, SessionId)
```

Parameters

EventType (integer)

This parameter specifies which type of tone or music should be played on queued call.

EVENT TYPE PLAY MUSIC	0
EVENT TYPE PLAY RING	1

QueueName (string)

The value of this parameter specifies queue name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnQueuePlayWaveStarted(EventType, QueueName, SessionId)
{
    if (EventType = 0)
        PlayWaveStartToCallSession(SessionId, -1, MusicWaveId, 1, 0)

    if (EventType = 1)
        PlayWaveStartToCallSession(SessionId, -1, RingWaveId, 1, 0)
}
```

See Also

OnQueuePlayWaveEnded(), AddCallSessionToQueue(),
OnQueueAgentConnectStarted(), OnQueueAgentConnectSuccess(),
PlayWaveStartToCallSession()

OnQueuePlayWaveEnded()

The OnQueuePlayWaveEnded() event triggers to notify the application that the parked call is now connected and in an active conversation. Consequently, the application should stop the playback of the music or .wav file initiated during the call parking process.

Syntax

```
OnQueuePlayWaveEnded(EventType, SessionId, ChannelId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

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

    if (DialNo = "0822")
        ConnectToParkedCallSession(SessionId, 0, 0822)
}

OnCallParkPlayWaveEnded(SessionId, ChannelId)
{
    PlayWaveStopToCallSession(SessionId, -1, -1)
}
```

See Also

OnCallParkPlayWaveStarted(), ParkCallSession(),
ConnectToParkedCallSession()

OnQueueAgentConnectStarted()

The OnQueueAgentConnectStarted() event triggers.

Syntax

```
OnQueueAgentConnectStarted(QueueName, AgentName, SessionId)
```

Parameters

QueueName (string)

The parameter specifies queue name.

AgentName (string)

The parameter identifies the user name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnQueueAgentConnectStarted(QueueName, AgentName, SessionId)
{
    PlayWaveStopToCallSession(SessionId, -1, -1)
}
```

See Also

OnQueueAgentConnectTrying(), OnQueueAgentConnectFailed(),
OnQueueAgentConnectTimeout(), OnQueueAgentConnectSuccess()

OnQueueAgentConnectTrying()

The OnQueueAgentConnectTrying() event triggers.

Syntax

```
OnQueueAgentConnectTrying(  
    QueueName,  
    AgentName,  
    SessionId,  
    StatusCode,  
    ReasonPhrase  
)
```

Parameters

QueueName (string)

The parameter specifies queue name.

AgentName (string)

The parameter identifies the user name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

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.

Example

```
OnQueueAgentConnectTrying(QueueName, AgentName, SessionId,  
                           StatusCode, ReasonPhrase)  
{  
    PlayWaveStopToCallSession(SessionId, -1, -1)  
}
```

See Also

OnQueueAgentConnectStarted(), OnQueueAgentConnectFailed(),
OnQueueAgentConnectTimeout(), OnQueueAgentConnectSuccess()

OnQueueAgentConnectFailed()

The OnQueueAgentConnectFailed() event triggers.

Syntax

```
OnQueueAgentConnectFailed(  
    QueueName,  
    AgentName,  
    SessionId,  
    StatusCode,  
    ReasonPhrase  
)
```

Parameters

QueueName (string)

The parameter specifies queue name.

AgentName (string)

The parameter identifies the user name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

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.

Example

```
OnQueueAgentConnectFailed(QueueName, AgentName, SessionId,  
    StatusCode, ReasonPhrase)  
{  
    PlayWaveStopToCallSession(SessionId, -1, -1)  
}
```

See Also

OnQueueAgentConnectStarted(), OnQueueAgentConnectTrying(),
OnQueueAgentConnectTimeout(), OnQueueAgentConnectSuccess()

OnQueueAgentConnectTimeout()

The OnQueueAgentConnectTimeout() event triggers.

Syntax

```
OnQueueAgentConnectTimeout(QueueName, AgentName, SessionId)
```

Parameters

QueueName (string)

The parameter specifies queue name.

AgentName (string)

The parameter identifies the user name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnQueueAgentConnectTimeout(QueueName, AgentName, SessionId)
{
    PlayWaveStopToCallSession(SessionId, -1, -1)
}
```

See Also

OnQueueAgentConnectStarted(), OnQueueAgentConnectTrying(),
OnQueueAgentConnectFailed(), OnQueueAgentConnectSuccess()

OnQueueAgentConnectSuccess()

The OnQueueAgentConnectSuccess() event triggers.

Syntax

```
OnQueueAgentConnectSuccess(QueueName, AgentName, SessionId)
```

Parameters

QueueName (string)

The parameter specifies queue name.

AgentName (string)

The parameter identifies the user name.

SessionId (integer)

This parameter specifies a unique identifier for a call session.

Example

```
OnQueueAgentConnectSuccess(QueueName, AgentName, SessionId)
{
    PlayWaveStopToCallSession(SessionId, -1, -1)
}
```

See Also

OnQueueAgentConnectStarted(), OnQueueAgentConnectTrying(),
OnQueueAgentConnectFailed(), OnQueueAgentConnectTimeout()

OnAddCallSessionToRingGroupSuccess()

The OnAddCallSessionToRingGroupSuccess() event triggers when VaxVoIP 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 identifier for a call session.

Example

```
OnAddCallSessionToRingGroupSuccess(GroupName, SessionId)
{
}
}
```

See Also

AddCallSessionToRingGroup()

OnAddCallSessionToRingGroupFailed()

The OnAddCallSessionToRingGroupFailed() event triggers 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 identifier for a call session.

Example

```
OnAddCallSessionToRingGroupFailed(GroupName, SessionId)
{
}
}
```

See Also

AddCallSessionToRingGroup()

OnCallParkPlayWaveStarted()

The OnCallParkPlayWaveStarted() event notifies the application that the call parking process initiated by the ParkCallSession() method has been completed.

This means that the application, which is based on the VaxVoIP DLL, can now play music or a .wav file. To play the audio, use the PlayWaveStartToCallSession() method.

Syntax

```
OnCallParkPlayWaveStarted(SessionId, ChannelId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, DialNo, FromPeerType,
               FromPeerName, RouteUID, RouteType, RouteName,
               UserAgentName, FromIP, FromPort)
{
    AcceptCallSession(SessionId, "", "", "", "", "", 20)
    ParkCallSession(SessionId, 0, 0822) // Parks the call in slot 0822
}

OnCallParkPlayWaveStarted(SessionId, ChannelId)
{
    WaveId = LoadWaveFile("Music.wav")
    PlayWaveStartToCallSession(SessionId, -1, WaveId, 1, 0)

    // Unloads the wave when the call disconnects
    UnLoadWaveID(WaveId)
}
```

See Also

OnCallParkPlayWaveEnded(), ParkCallSession(),
ConnectToParkedCallSession()

OnCallParkPlayWaveEnded()

The OnCallParkPlayWaveEnded() event triggers to notify the application that the parked call is now connected and in an active conversation. Consequently, the application should stop the playback of the music or .wav file initiated during the call parking process.

Syntax

```
OnCallParkPlayWaveEnded(SessionId, ChannelId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

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

    if (DialNo = "0822")
        ConnectToParkedCallSession(SessionId, 0, 0822)
}

OnCallParkPlayWaveEnded(SessionId, ChannelId)
{
    PlayWaveStopToCallSession(SessionId, -1, -1)
}
```

See Also

OnCallParkPlayWaveStarted(), ParkCallSession(),
ConnectToParkedCallSession()

OnServerConnectingREC()

The OnServerConnectingREC() event triggers when VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 VaxVoIP 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within 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 identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

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

See Also

ConnectToServerREC(), OnIncomingCall()

OnServerHungupREC()

The OnServerHungupREC() event triggers when any party close the call.

Syntax

```
OnServerHungupREC(SessionId, ChannelId)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

Example

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

See Also

OnIncomingCall(), ConnectToServerREC()

OnCallSessionRecordedWaveREC()

The OnCallSessionRecordedWaveREC() event is triggered when a recorded wave file of a specific call is available. This event provides the recorded audio data in the form of an array, which represents the wave file.

To initiate recording, use the RecordWaveStartToCallSession() method with an empty string for the FileName parameter. This method begins recording and stores the audio data directly in memory (RAM).

When the recording needs to be stopped, either due to the call being disconnected or by invoking the RecordWaveStopToCallSession() method, the recording is halted. Subsequently, the OnCallSessionRecordedWaveREC() event is triggered, delivering the recorded wave file data as an array for further processing or analysis.

Syntax

```
OnCallSessionRecordedWaveREC(  
    SessionId,  
    ChannelId,  
    DataREC,  
    SizeREC,  
    TypeREC  
)
```

Parameters

SessionId (integer)

This parameter specifies a unique identifier for a call session.

ChannelId (integer)

This parameter identifies a call within a call session.

0 = Channel-ZERO call

1 = Channel-ONE call

DataREC (array)

This parameter specifies the complete wave file.

SizeREC (integer)

This parameter specifies the size of wave file.

TypeREC (integer)

The parameter specifies the type of wave file.

REC TYPE MONO PCM	0
REC TYPE STEREO PCM	1
REC TYPE MONO G711U	2
REC TYPE STEREO G711U	3
REC TYPE MONO G711A	4
REC TYPE STEREO G711A	5

Example

```
OnCallSessionConnected(SessionId)
{
    RecordWaveStartToCallSession(SessionId, 0, "", 2)
}

OnCallSessionRecordedWaveREC(SessionId, ChannelId, DataREC, SizeREC,
                              TypeREC)
{
    // DataREC is a WAV file containing the header & audio data.
    // Write the DataREC to a file with any name and a .wav extension.
}
```

See Also

RecordWaveStartToCallSession()

OnConferenceRoomRecordedWaveREC()

The OnConferenceRoomRecordedWaveREC() event is triggered when a recorded wave file of a specific conference room is available. This event delivers the recorded audio data in the form of an array, representing the wave file of the conference session.

The RecordWaveStartToConferenceRoom() method, when called with an empty string for the FileName parameter, initiates recording and stores the audio data directly in memory (RAM). To stop the recording, use the RecordWaveStopToConferenceRoom() method.

Once the recording is stopped, the OnConferenceRoomRecordedWaveREC() event is triggered, providing the recorded wave file data as an array. This array contains the audio data from the conference room session, allowing for further processing or analysis.

Syntax

```
OnConferenceRoomRecordedWaveREC(  
                                RoomName,  
                                DataREC,  
                                SizeREC,  
                                TypeREC  
                                )
```

Parameters

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

DataREC (array)
This parameter specifies the complete wave file.

SizeREC (integer)
This parameter specifies the size of wave file.

TypeREC (integer)
The parameter specifies the type of wave file.

REC TYPE MONO PCM	0
REC TYPE STEREO PCM	1
REC TYPE MONO G711U	2
REC TYPE STEREO G711U	3
REC TYPE MONO G711A	4
REC TYPE STEREO G711A	5

Example

```
OpenConferenceRoom("TechRoom")
RecordWaveStartToConferenceRoom("TechRoom", "", 2)

OnCallSessionConnected(SessionId)
{
    AddCallSessionToConferenceRoom("TechRoom", SessionId)
}

RecordWaveStopToConferenceRoom("TechRoom")

OnConferenceRoomRecordedWaveREC(RoomName, DataREC, SizeREC,
                                TypeREC)
{
    // DataREC is a WAV file containing the header & audio data.
    // Write the DataREC to a file with any name and a .wav extension.
}
```

See Also

RecordWaveStartToConferenceRoom()

OnAttackDetectedScanSIP()

The OnAttackDetectedScanSIP() event is triggered when the SIP server detects a potential attack, such as unauthorized or suspicious SIP traffic, based on certain criteria defined in the **AttackDetectScanSIP()** method.

Syntax

```
OnAttackDetectedScanSIP(  
    ReqMethod,  
    AddrIP,  
    AddrPort,  
    AddrType  
)
```

Parameters

ReqMethod (string)

Represents the request method in the incoming SIP packet, such as "INVITE," "REGISTER," or other SIP methods. This helps identify the type of SIP request that triggered the detection event.

AddrIP (string)

The IP address of the source that sent the suspicious SIP request. This allows identification of the source and can be used for blocking or investigating the IP further.

AddrPort (integer)

The port number of the source address from which the SIP request was sent. This helps pinpoint the exact source and enables blocking or restricting traffic from that port.

AddrType (integer)

Specifies the transport protocol used for the SIP request.

- 0: UDP (User Datagram Protocol)
- 1: TCP (Transmission Control Protocol)
- 2: TLS (Transport Layer Security)

This parameter indicates the transport protocol type, which is useful for differentiating between network communication types.

Example

```
AttackDetectScanSIP("demo.sipsdk.com")

OnAttackDetectedScanSIP(ReqMethod, AddrIP, AddrPort, AddrType)
{
    // Use Windows Defender Firewall APIs
    // Run command prompt-based Firewall commands
    // Block AddrIP and AddrPort.
}
```

See Also

AttackDetectScanSIP()

OnAttackDetectedFloodSIP()

The OnAttackDetectedFloodSIP() event is triggered when the SIP server detects a potential flooding attack, which occurs when the number of incoming SIP requests exceeds the threshold defined by the **AttackDetectFloodSIP()** method. This event provides details about the source of the flooding attempt, such as the IP address, port, and protocol type used by the attacker.

Syntax

```
OnAttackDetectedFloodSIP(  
    AddrIP,  
    AddrPort,  
    AddrType  
)
```

Parameters

AddrIP (string)

This parameter contains the IP address of the source that sent the excessive SIP requests. It identifies the origin of the flood and can be used for blocking or further investigation.

AddrPort (integer)

This parameter specifies the port number from which the flood of SIP requests was sent. It helps pinpoint the source of the attack at the network level, allowing the server to block or restrict traffic from that port.

AddrType (integer)

Indicates the transport protocol used for the SIP requests.

- 0: UDP (User Datagram Protocol)
- 1: TCP (Transmission Control Protocol)
- 2: TLS (Transport Layer Security)

This parameter helps identify the transport protocol used by the attacker, which can be useful for applying specific security measures based on the protocol type.

Example

```
AttackDetectFloodSIP(2000)

AttackDetectFloodSIP(AddrIP, AddrPort, AddrType)
{
    // Use Windows Defender Firewall APIs
    // Run command prompt-based Firewall commands
    // Block AddrIP and AddrPort.
}
```

See Also

AttackDetectFloodSIP()

OnAttackDetectedBruteForceSIP()

The OnAttackDetectedBruteForceSIP() event is triggered when the SIP server detects a potential brute-force attack, which occurs when there are multiple failed authentication attempts within a specified time period. This event provides detailed information about the failed attempts, including the request method, the number of failed attempts, and the source of the attack.

Syntax

```
OnAttackDetectedFloodSIP(ReqMethod, AuthFailureCount, AddrIP,  
                          AddrPort, AddrType)
```

Parameters

ReqMethod (string)

Represents the SIP request method (e.g., "INVITE," "REGISTER") associated with the authentication failures. This helps identify which SIP method the brute-force attempts are targeting.

AuthFailureCount (integer)

The number of authentication failure attempts that were detected within the time window specified by the **AttackDetectBruteForceSIP()** method. If this count exceeds the defined threshold, it triggers the event to alert about the potential brute-force attack.

AddrIP (string)

The IP address of the source from which the failed authentication attempts originated. This helps to identify the source of the attack and can be used for blocking or further investigation.

AddrPort (integer)

The port number from which the failed authentication attempts were made. This helps pinpoint the exact source port for further action or investigation.

AddrType (integer)

Indicates the transport protocol used for the SIP requests.

- 0: UDP (User Datagram Protocol)
- 1: TCP (Transmission Control Protocol)
- 2: TLS (Transport Layer Security)

This parameter helps identify the transport protocol used by the attacker, which can be useful for applying specific security measures based on the protocol type.

Example

```
AttackDetectBruteForceSIP(20, 5) // 5 seconds

OnAttackDetectedFloodSIP(ReqMethod, AuthFailureCount, AddrIP,
                          AddrPort, AddrType)
{
    // Use Windows Defender Firewall APIs
    // Run command prompt-based Firewall commands
    // Block AddrIP and AddrPort.
}
```

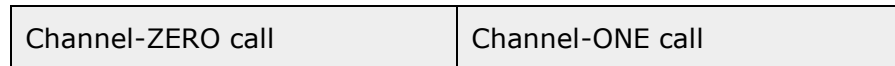
See Also

AttackDetectBruteForceSIP()

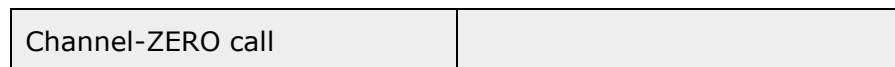
WHAT IS A CALL-SESSION

A Call Session represents a grouping of one or two calls that are managed together. Within a Call Session, the calls are identified as either a Channel-ZERO call or a Channel-ONE call. This structure allows for organized management and control of the calls, where Channel-ZERO typically refers to the primary call, and Channel-ONE may refer to a secondary or associated call.

Call-Session with two calls (Channel-ZERO call & Channel-ONE call)



Call-Session with one call (Channel-ZERO call)



Call-Session with one call (Channel-ONE call)



MANAGING A CALL SESSION WITH TWO CALLS (CHANNEL-ZERO & CHANNEL-ONE)

The VaxVoIP COM component receives SIP-based call requests. Upon receiving these requests, it internally creates and allocates a Call Session. The incoming call is assigned as the Channel-ZERO call within this Call Session. Following this, the `OnIncomingCall()` event is triggered to notify the application of the new incoming call.

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

The `AcceptCallSession()` function, when provided with appropriate values for `DialNo` and `ToPeerName`, initiates an outgoing call request and adds this outgoing call as the Channel-ONE call to the same Call Session that was created for the incoming call.

Additionally, `AcceptCallSession()` connects and accepts the Channel-ZERO call (the incoming call) within that Call Session, thereby establishing a connection between both calls.

MANAGING A CALL SESSION WITH ONE CALL (CHANNEL-ZERO)

The VaxVoIP COM component receives SIP-based call requests. Upon receipt, it internally creates and allocates a Call Session. The incoming call is designated as the Channel-ZERO call within this Call Session, and the OnIncomingCall() event is triggered to notify the application of the incoming call.

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

To accept and connect only the incoming call without initiating an outgoing call, the AcceptCallSession() function should be called with empty values for DialNo, ToPeerName, and RoutID. This configuration ensures that only the incoming call (Channel-ZERO) is connected within the Call Session.

MANAGING A CALL SESSION WITH ONE CALL (CHANNEL-ONE)

The VaxVoIP COM component exports the DialCallSession() function, which sends a SIP-based call request to initiate a call.

This function is particularly useful for developing predictive dialers and telemarketing software, enabling automated and efficient call handling.

```
OnVaxTeleTick(TickId)
{
    SessionId = DialCallSession(CallerName, CallerId, DialNo,
                               ToPeerName, 20)
}
```

The DialCallSession() function internally creates a new Call Session and adds the outbound call as the Channel-ONE call within this session. This process ensures that the newly initiated call is properly managed and tracked within the context of the created 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 BYE 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 to the destination address. Provided address or port is not valid.
276	Unable to capture or bind to the destination address. 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).
288	Call already attached.
289	Async read IO failed.
290	Async write IO failed.
291	Socket attach IO failed.
292	Socket read IO failed.
293	Socket write IO failed.

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 PROCESS

Please see [SIP CLIENT REGISTRATION PROCESS](#) document for further details.

SIP PHONE TO SIP PHONE CALL FLOW

Please see [SIP PHONE TO SIP PHONE CALL FLOW](#) document for further details.

ACCESS AUDIO DATA PCM PROCESS

Please see [ACCESS AUDIO DATA PCM](#) 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.