VaxVoIP SIP RECORDING SDK

TECHNICAL DOCUMENTATION

VERSION 8.0.2.6

CONTENTS

INTRODUCTION AND QUICK START

The VaxVoIP SIP CALL RECORDING COM (Component Object Model) component, specifically the VaxTeleServerREC.dll, serves as a robust framework comprising various functions and events tailored for the seamless development of SIP REC (Session Initiation Protocol Recording) Protocol-based Call Recording Servers.

This dynamic VaxVoIP SIP REC COM component, encapsulated within the VaxTeleServerREC.dll, empowers developers by providing a comprehensive set of functions and events. This facilitates the swift and efficient creation of SIP REC-based Call Recording Servers, offering versatility for integration in diverse environments such as call centers, offices, and other SIP-based VoIP services.

By leveraging the capabilities of the VaxVoIP SIP REC COM component, developers gain a powerful toolset to enhance call recording functionalities, ensuring compatibility with SIP protocols and optimizing performance across various VoIP services.

EXPORTED FUNCTIONS

SetLicenseKey()

The trial version of the VaxVoIP SDK is subject to a time limitation of 30 days. To continue using the SDK beyond this trial period without encountering an evaluation message box, a valid license key is necessary. License keys are provided to customers upon placing an order.

The SetLicenseKey() method is crucial for converting the trial version of the VaxVoIP SDK into a fully registered version, removing any time limitations or trial restrictions. This method requires users to input a valid license key, enabling uninterrupted and unrestricted access to the SDK's complete functionality.

Syntax

```
SetLicenseKey(LicenseKey)
```

Parameters

LicenseKey (string)
The value of this parameter is license key provided by the VaxVoIP.

Return Value

No return value.

Example

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

See Also

Initialize(), GetVaxErrorCode()

GetVaxErrorCode()

The GetVaxErrorCode() method retrieves the error code associated with the most recent operation that failed to execute successfully. This function serves as a tool for developers to identify and diagnose issues by obtaining specific error codes related to the failed operation, facilitating effective troubleshooting and resolution.

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

Syntax

```
integer GetVaxErrorCode()
```

Parameters

No parameters.

Return Value

The GetVaxErrorCode() returns the error code.

Example

```
SetLicenseKey("LicenseKey")

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

See Also

Initialize(), SetLicenseKey()

Initialize()

The Initialize() function is tasked with configuring the VaxVoIP SIP REC server COM component. This process includes allocating internal memory resources and preparing the component for use by the application.

By executing this function, the component becomes ready to expose its exported methods, allowing the application to interact with and utilize its functionalities.

Syntax

```
boolean Initialize(DomainRealm)
```

Parameters

DomainRealm (string)

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

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

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

But if value of DomainRealm is "" empty string and assigned IP value is 10.3.5.66 then VaxTele creates SIP URI sip:username@10.3.5.66

Note: It is not necessary that an IP-address should be assigned to DomainRealm.

Return Value

Upon successful execution, this function returns a non-zero value. If the execution encounters an issue, it returns 0, and developers can retrieve a specific error code by calling the GetVaxErrorCode() method.

Example

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

See Also

UnInitialize(), GetVaxErrorCode(), SetListenPortRangeRTP()

OpenNetworkUDP()

The OpenNetworkUDP() function is designed to initiate the opening of a UDP (User Datagram Protocol) socket/network within the context of the VaxVoIP SIP server COM component. This process involves allocating a UDP listen port and commencing the reception of incoming UDP-based SIP requests.

Syntax

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

Parameters

ListenIP (string)

This parameter's value determines the IP address on which VaxVoIP will listen for incoming UDP-based SIP requests. It can either be an empty value or an IP address assigned to the computer where the VaxVoIP COM integrated SIP-REC server is running.

ListenPort (integer)

This parameter's value designates the port number for the SIP server to receive SIP requests. The standard SIP listen port is 5060.

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

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

See Also

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

OpenNetworkTCP()

The OpenNetworkTCP() function is responsible for setting up a TCP socket/network in the VaxVoIP SIP server COM component. It involves allocating a TCP listen port, initiating the reception of incoming TCP-based SIP requests, and triggering relevant COM events.

Syntax

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

Parameters

ListenIP (string)

This parameter's value determines the IP address on which VaxVoIP will listen for incoming TCP-based SIP requests. 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's value designates the port number for the SIP server to receive SIP requests. The standard SIP listen port is 5060.

Return Value

If the function executes successfully, it returns a non-zero value. In case of failure, it returns 0, and you can retrieve a specific error code by calling the GetVaxErrorCode() method.

Example

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

See Also

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

OpenNetworkTLS()

The OpenNetworkTCP() function establishes a TLS communication channel/network. It allocates a TLS listen port, begins listening for incoming TLS-based SIP requests, and triggers the associated COM (Component Object Model) events accordingly.

Syntax

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

Parameters

ListenIP (string)
: This parameter's value determines the IP address on which VaxVoIP will listen for incoming TLS-based SIP requests. 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's value designates the port number for the SIP server to receive SIP requests. The standard SIP listen port is 5061.

CertPEM (string)
: This parameter's value designates the SSL certificate (.PEM) file name or its data in string/text format. It can be an empty value, in which case VaxVoIP COM will utilize the default VaxVoIP SSL certificate.

Return Value

If the function executes successfully, it returns a non-zero value. In case of failure, it returns 0, and you can retrieve a specific error code by calling the GetVaxErrorCode() method.

Example

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

See Also

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

CloseNetworkUDP()

The CloseNetworkUDP() function is responsible for shutting down the UDP socket/network.

Syntax

```
CloseNetworkUDP()
```

Parameters

No parameters.

Return Value

No return value.

Example

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

See Also

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

CloseNetworkTCP()

The CloseNetworkTCP() function is utilized to close the TCP socket/network.

Syntax

```
CloseNetworkTCP()
```

Parameters

No parameters.

Return Value

No return value.

Example

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

See Also

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

CloseNetworkTLS()

The CloseNetworkTLS() function is used to close the TLS communication channel/network..

Syntax

```
CloseNetworkTLS()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
OpenNetworkTLS("", 5061, "")
CloseNetworkTLS()
```

See Also

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

SetListenPortRangeRTP()

The SetListenPortRangeRTP() function sets the specified range, allowing VaxREC to allocate/open the listen RTP port within that defined range.

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

Syntax

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

Parameters

ListenStartPort (integer)
: This parameter's value designates the starting port of the specified range.

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

Return Value

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

Example

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

SetListenPortRangeRTP(8088, 32406)
```

See Also

Initialize(), GetVaxErrorCode()

AddNetworkRouteSIP(), AddNetworkRouteRTP()

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

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

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

Syntax

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

Parameters

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

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

Return Value

This function returns a non-zero value upon successful execution. In case of failure, it returns 0, and you can obtain a specific error code by invoking the GetVaxErrorCode() method.

Example

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

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

SetListenPortRangeRTP(8000, 32406)
```

Run VaxVoIP based SIP SERVER behind Firewall/router, from the router settings;

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

See Also

Initialize(), SetListenPortRangeRTP(), GetVaxErrorCode()

UnInitialize()

The UnInitialize() function straightforwardly releases all resources allocated by the Initialize() function.

Syntax

```
UnInitialize()
```

Parameters

No parameters.

Return Value

No return value.

Example

```
UnInitialize()
```

See Also

Initialize()

AddUser()

The AddUser() function is used to add users to the SIP server developed by the VaxVoIP COM component. The VaxTele COM component, specifically the VaxTeleServerREC.dll, internally maintains a list of users to handle SIP registration and call requests.

By utilizing the AddUser() function, you can add a user with a specific login and password to the VaxVoIP-based SIP REC SERVER. This login and password combination can then be used in any SIP REC-based softphone, SIP Server, VaxPhoneSDK, VaxServerSDK or any other SIP REC client. This enables users to connect and register with the VaxTele-based SIP server using the provided credentials.

It's great to know that the VaxPhoneSDK and VaxServerSDK support the SIP REC protocol, seamlessly integrating with the VaxVoIP SIP REC SDK. This compatibility enables the implementation of call recording features within a VoIP network.

Syntax

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

Parameters

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

Password (string)
: This parameter value specifies the password of user’s login.

CodecList (string)
: This parameter value specifies the list of audio codecs to be added in the call request.

  Supported codecs:

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

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

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

Return Value

Upon successful execution, this function returns a non-zero value. If the execution encounters an issue, it returns 0, and developers can retrieve a specific error code by calling the GetVaxErrorCode() method.

Example

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

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

See Also

RemoveUser(), GetVaxErrorCode()

RemoveUser()

The RemoveUser() function is designed to remove a user that was previously added using the AddUser() function. This allows for the management and modification of the list of users within the VaxVoIP based SIP REC SERVER.

Syntax

```
RemoveUser(UserName)
```

Parameters

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

Return Value

No return value.

Example

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

See Also

AddUser()

RegisterUserExpiry()

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

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

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

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

Syntax

```
boolean RegisterUserExpiry(Expiry)
```

Parameters

Expiry (integer)
The Expire parameter specifies the time interval in seconds.

Return Value

Upon successful execution, the function returns a non-zero value. In the event of failure, it returns 0, and developers can obtain a specific error code by invoking the GetVaxErrorCode() method..

Example

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

See Also

AddUser(), OnRegisterUser()

AcceptRegister()

The AcceptRegister() f unction is used to accept a registration request received from a SIP client.

During the SIP registration process, SIP REC clients, which can include softphones, Session Border Controllers (SBCs), SIP servers, etc., send registration requests. SIP REC SERVERS, such as the VaxVoIP SIP REC SERVER, authenticate the provided login and password, and upon successful authorization, accept the registration requests from the SIP REC clients. This secure authentication process ensures the proper registration of SIP REC clients in the VoIP network.

When a registration request is received, the VaxREC triggers the OnRegisterUser() event. If the AcceptRegister() function is called within this event, it accepts the registration request without performing an authorization check for the login and password. This allows for a streamlined process where registration requests are accepted without additional login and password validation during the OnRegisterUser() event.

Exactly, the AcceptRegister() function skips the login authorization process and directly accepts the registration request. This implies that, when invoked, it approves the registration without going through the usual login and password validation steps.

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

Syntax

```
boolean AcceptRegister(RegId)
```

Parameters

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

Return Value

If the function executes successfully, it returns a non-zero value. Conversely, in case of failure, it returns 0. To retrieve a specific error code related to the failure, you can use the GetVaxErrorCode() method.

Example

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

See Also

AuthRegister(), RejectRegister(), OnRegisterUser()

RejectRegister()

The RejectRegister() function in VaxREC is used to decline a registration request sent to it by a SIP client.

When the RejectRegister() function is called within the OnRegisterUser() event, it straightforwardly rejects the registration request, providing a mechanism to control and manage incoming registrations.

Please see **LIST OF SIP RESPONSES** for more details.

Syntax

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

Parameters

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

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

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

Return Value

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

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

See Also

AuthRegister(), AcceptRegister(), OnRegisterUser()

AuthRegister()

The AuthRegister() function initiates the authentication process for a specified user before accepting a registration request. This allows for additional security measures, such as validating login credentials, before approving the registration.

When SIP REC clients send SIP register requests to connect and register with SIP server(s), VaxREC receives these register requests and triggers the OnRegisterUser() event. To commence the authentication process, VaxREC calls the AuthRegister() function. This function is responsible for initiating the necessary steps to authenticate the user before accepting the registration request.

Use AuthRegister() function and then VaxREC;

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

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

Syntax

```
boolean AuthRegister(RegId)
```

Parameters

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

Return Value

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

Example

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

See Also

AcceptRegister(), RejectRegister(), OnRegisterUser()

AddLine()

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

VaxREC is an SDK for SIP-REC (Session Initiation Protocol Recording) servers. Software created with the VaxREC SDK can connect to other SIP servers and devices using the SIP protocol, facilitating the exchange of SIP-REC call requests with these servers and devices.

To integrate VaxREC with another SIP server, create a user login on the target SIP server. Then, configure VaxREC by adding the user account settings using the AddLine() function. This establishes a connection between VaxREC and the SIP server, allowing seamless communication.

Syntax

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

Parameters

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

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

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

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

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

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

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

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

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

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

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

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

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

Return Value

This function returns a non-zero value upon successful execution. In case of failure, it returns 0, and you can obtain a specific error code by invoking the GetVaxErrorCode() method.

Example

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

See Also

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

RemoveLine()

The RemoveLine() function is designed to remove a specific line that was previously added using the AddLine() function. This allows for dynamic management and modification of the lines within the VaxVoIP REC server COM component.

Syntax

```
RemoveLine(LineName)
```

Parameters

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

Return Value

No return value.

Example

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

See Also

AddLine()

RegisterLine()

The RegisterLine() function is utilized to register a line with an external third-party SIP-REC supported SIP server. During the line registration process, VaxREC triggers various registration-related events such as OnLineRegisterTrying(), OnLineRegisterSuccess(), etc.

These events provide feedback and notifications at different stages of the line registration process, allowing for effective handling and monitoring.

Prior to registration, it is necessary to add the line using the AddLine() function before invoking the RegisterLine() function.

Syntax

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

Parameters

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

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

Return Value

Upon successful execution, the function returns a non-zero value. In the event of failure, it returns 0, and developers can obtain a specific error code by invoking the GetVaxErrorCode() method.

Example

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

See Also

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

UnRegisterLine()

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

Syntax

```
boolean UnRegisterLine(LineName)
```

Parameters

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

Return Value

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

Example

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

See Also

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

AcceptCallSession()

The AcceptCallSession() function is used to accept an incoming SIP-REC call request.

The OnIncomingCall() event triggers when the VaxREC-based SIP REC Server receives an incoming call request. This event, in turn, calls the AcceptCallSession() method to process the incoming call request accordingly, facilitating the acceptance and handling of the call session.

Syntax

```
boolean AcceptCallSession(
                              SessionId,
                              Timeout
                              )
```

Parameters

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

Timeout (integer)
: This parameter specifies the time interval, in seconds, during which VaxREC waits for the Call-Session to be connected or established. If the Call-Session is not connected within the specified time interval, a timeout occurs, triggering the OnCallSessionTimeout() event. This event provides notification when a timeout occurs during the establishment of a Call-Session.

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

```
OnIncomingCall(SessionId, CallerName, CallerId, CalleeName, CalleeId,
                    DialNo, FromPeerType, FromPeerName, UserAgentName,
                    RecXML, FromIP, FromPort)
{
        AcceptCallSession(SessionId, 20)
}
```

See Also

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

RejectCallSession()

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

Syntax

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

Parameters

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

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

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

Return Value

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

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, CalleeName, CalleeId,
                DialNo, FromPeerType, FromPeerName, UserAgentName,
                RecXML, FromIP, FromPort)
{
        RejectCallSession(SessionId, 404, "Not found")
}
```

See Also

AcceptCallSession(), CloseCallSession(), OnIncomingCall()

CloseCallSession()

The CloseCallSession() function closes the session, disconnecting all associated calls.

Syntax

```
boolean CloseCallSession(SessionId)
```

Parameters

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

Return Value

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

Example

```
CloseCallSession(SessionId)
```

See Also

RejectCallSession(), AcceptCallSession(), OnCallSessionClosed()

RecordWaveStartToCallSession()

The RecordWaveStartToCallSession() function starts recording the voice stream of a particular call in a Call-Session to a Memory or Wave File (.wav).

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

Syntax

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

Parameters

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

FileName (string)
: The value of this parameter specifies the file name. If the value is an empty string (" "), then VaxREC starts recording in memory. When the recording stops or the call gets disconnected, it triggers the OnCallSessionRecordedWaveREC() event.

TypeREC (integer)
: The parameter's value designates the recording type.

  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

Upon successful execution, the function returns a non-zero value. In the event of failure, it returns 0, and developers can obtain a specific error code by invoking the GetVaxErrorCode() method.

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId, "Record.wav", 1)
}

OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId, "", 1)
}

OnCallSessionRecordedWaveREC(SessionId, DataREC, SizeREC)
{
        //DataREC is refrence to Memory Array with WaveFile Data
        //Open a binary file and write data
}
```

See Also

RecordWaveStopToCallSession(), RecordWavePauseToCallSession(), GetVaxErrorCode()

RecordWaveStopToCallSession()

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

Syntax

```
boolean RecordWaveStopToCallSession(SessionId)
```

Parameters

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

Return Value

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

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId, "Record.wav", 1)
}

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

See Also

RecordWaveStartToCallSession(), RecordWavePauseToCallSession(), GetVaxErrorCode()

RecordWavePauseToCallSession()

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

Syntax

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

Parameters

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

Enable (Boolean)
: This parameter value can be 0 or 1. Assign value 1 to pause the recording process or 0 to un-pause the recording process.

Return Value

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

Example

```
OnCallSessionConnected(SessionId)
{
        RecordWaveStartToCallSession(SessionId, "Record.wav", 2)
}

RecordWavePauseToCallSession(SessionId, 1)
RecordWavePauseToCallSession(SessionId, 0)
```

See Also

RecordWaveStartToCallSession(), RecordWaveStopToCallSession(), GetVaxErrorCode()

AudioSessionLost()

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

Syntax

```
boolean AudioSessionLost(Enable, Timeout)
```

Parameters

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

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

Return Value

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

Example

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

See Also

OnCallSessionLost(), GetVaxErrorCode()

DiagnosticLogSIP()

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

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

Syntax

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

Parameters

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

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

Return Value

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

Example

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

See Also

OnOutgoingDiagnosticLOg(), OnIncomingDiagnosticLog(), GetVaxErrorCode()

VaxServerStartTick()

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

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

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

Syntax

```
boolean VaxServerStartTick(TickId, Elapse)
```

Parameters

TickId (integer)
: This parameter specifies the unique tick identification.

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

Return Value

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

Example

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

VaxServerStartTick(2001, 8000)
```

See Also

VaxServerStopTick(), OnVaxServerTick(), GetVaxErrorCode()

VaxServerStopTick()

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

Syntax

```
boolean VaxServerStopTick(TickId)
```

Parameters

TickId (integer)
This parameter specifies the unique tick identification

Return Value

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

Example

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

VaxServerStartTick(2001, 8000)

VaxServerStopTick(2001)
```

See Also

VaxServerStartTick(), OnVaxServerTick(), GetVaxErrorCode()

SetUserAgentName()

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

Syntax

```
boolean SetUserAgentName(Name)
```

Parameters

Name (string)
This parameter value specifies the User agent name.

Return Value

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

Example

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

See Also

GetUserAgentName(), GetVaxErrorCode()

GetUserAgentName()

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

Syntax

```
string GetUserAgentName()
```

Parameters

No parameters.

Return Value

The function returns the user agent name otherwise empty string.

Example

```
GetUserAgentName()
```

See Also

SetUserAgentName()

SetSessionNameSDP()

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

Syntax

```
boolean SetSessionNameSDP(Name)
```

Parameters

Name (string)
: This parameter specifies the value that is to be set as session name of SIP packet.

Return Value

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

Example

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

See Also

GetSessionNameSDP(), GetVaxErrorCode()

GetSessionNameSDP()

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

Syntax

```
string GetSessionNameSDP()
```

Parameters

No parameters.

Return Value

The function returns the session name otherwise empty string.

Example

```
GetSessionNameSDP()
```

See Also

SetSessionNameSDP()

GetCallSessionTxCodec()

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

Syntax

```
integer GetCallSessionTxCodec(SessionId, ChannelId)
```

Parameters

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

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

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

Return Value

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

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

Audio Codec Numbers:

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

Example

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

See Also

GetCallSessionRxCodec(), GetVaxErrorCode()

GetCallSessionRxCodec()

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

Syntax

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

Parameters

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

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

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

Return Value

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

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

Audio Codec Numbers:

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

Example

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

See Also

GetCallSessionTxCodec(), GetVaxErrorCode()

CallSessionMuteVoice()

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

Syntax

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

Parameters

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

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

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

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

Speak (boolean)
This parameter specifies to mutes the speaking.

Return Value

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

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, CalleeName, CalleeId,
                DialNo, FromPeerType, FromPeerName, UserAgentName,
                RecXML, FromIP, FromPort)
{
    AcceptCallSession(SessionId, 20)
    CallSessionMuteVoice(SessionId, 1, 0)
}
```

See Also

OnIncomingCall(), GetVaxErrorCode()

AddCustomHeader()

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

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

Syntax

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

Parameters

ReqId (integer)
: This parameter specifies a unique identification of a SIP request. Supported ReqId values are;

  0 = INVITE
  1 = REFER
  2 = CANCEL
  3 = BYE

Name (string)
: This parameter specifies the name of custom header field.

Value (string)
: This parameter specifies the value of custom header field.

Return Value

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

Example

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

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

See Also

RemoveCustomHeader(), RemoveCustomHeaderAll(),

RemoveCustomHeader()

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

Syntax

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

Parameters

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

0 = INVITE
1 = REFER
2 = CANCEL
3 = BYE

Name (string)
This parameter specifies the custom header field.

Return Value

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

Example

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

See Also

AddCustomHeader(), RemoveCustomHeaderAll()

RemoveCustomHeaderAll()

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

Syntax

```
boolean RemoveCustomHeaderAll(ReqId)
```

Parameters

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

0 = INVITE
1 = REFER
2 = CANCEL
3 = BYE

Return Value

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

Example

```
RemoveCustomHeaderAll(0)
```

See Also

AddCustomHeader(), RemoveCustomHeader()

GetCallSessionHeaderCallId()

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

Syntax

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

Parameters

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

ChannelId (integer)

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

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

Return Value

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

Example

```
OnIncomingCall(SessionId, CallerName, CallerId, CalleeName, CalleeId,
                DialNo, FromPeerType, FromPeerName, UserAgentName,
                RecXML, FromIP, FromPort)
{
    HeaderId = GetCallSessionHeaderCallId(SessionId)
}
```

See Also

DialCallSession(), OnCallSessionCreated()

EXPORTED EVENTS

OnVaxErrorLog()

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

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

Syntax

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

Parameters

FuncName (string)
This parameter value specifies name of the function.

ErrorCode (integer)
This parameter value specifies error code.

ErrorMsg (string)
This parameter value specifies error text message.

Example

```
Result = Initialize("sipsdk.com")

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

OnVaxErrorLog(FuncName, ErrorCode, ErrorMsg)
{

}
```

See Also

GetVaxErrorCode()

OnLineRegisterTrying()

VaxREC triggers the OnLineRegisterTrying() event when it receives the SIP response "100, Trying" from another SIP server during the line registration process.

To connect and operate with external SIP-REC supported SIP servers, VaxREC uses the AddLine() and RegisterLine() functions. These functions facilitate the addition of a line and the subsequent registration process with the external SIP server.

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()

VaxREC triggers the OnLineRegisterFailed() event when the registration of a LINE (SIP account settings) to an external SIP-REC supported SIP server fails.

This event serves as a notification that the registration attempt was unsuccessful, allowing for appropriate handling and error management in the application.

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()

VaxREC triggers the OnLineRegisterSuccess() event when the registration request for a line (SIP account settings) to an external SIP-REC supported SIP server is successfully completed.

This event signals that the line registration process has been successful, allowing for further actions or notifications to be taken based on this success.

Syntax

```
OnLineRegisterSuccess(LineName)
```

Parameters

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

Example

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

See Also

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

OnLineUnRegisterTrying()

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

To unregister or disconnect VaxREC from an external third-party SIP-REC supported SIP Server, the UnRegisterLine() function is used. This function initiates the process of unregistering or disconnecting the line previously registered with the external SIP Server.

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()

VaxREC triggers OnLineUnRegisterFailed() event if a provided LINE fails to un-register from external SIP-REC supported SIP server.

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()

VaxREC triggers OnLineUnRegisterSuccess() event when line unregisters successfully from external SIP-REC supported SIP server.

Syntax

```
OnLineUnRegisterSuccess(LineName)
```

Parameters

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

Example

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

See Also

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

OnUnRegisterUser()

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

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

Syntax

```
OnUnRegisterUser(UserLogin)
```

Parameters

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

Example

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

See Also

OnRegisterUser(), RemoveUser(), AddUser()

OnRegisterUser()

The OnRegisterUser() event is triggered when VaxREC receives a register request from any SIP-REC client. This event provides a notification that a SIP-REC client has initiated a registration request, allowing the application to handle and process the request accordingly.

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

Syntax

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

Parameters

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

Domain (string)
: This parameter specifies the domain and its value is used to configure and register the SIP clients to VaxTele and other SIP servers.

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

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

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

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

Example

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

See Also

OnUnRegisterUser(), AddUser(), RemoveUser()

OnRegisterUserSuccess()

The OnRegisterUserSuccess() event triggers when SIP-REC client successfully registers to VaxREC server.

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

Syntax

```
OnRegisterUserSuccess(UserLogin)
```

Parameters

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

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

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

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

Example

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

See Also

OnRegisterUser(), OnRegisterUserFailed()

OnRegisterUserFailed()

The OnRegisterUserFailed() event triggers when SIP-REC client fails to register with VaxREC server.

Syntax

```
OnRegisterUserFailed(UserLogin)
```

Parameters

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

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

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

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

Example

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

See Also

OnRegisterUser(), OnRegisterUserSuccess(), RemoveUser()

OnCallSessionCreated()

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

Syntax

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

Parameters

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

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

- INCOMING_CALL 1001
- OUTGOING_CALL 1002
- MERGED 1003
- TRANSFERED 1004

Example

```
OnCallSessionCreated(SessionId, ReasonCode)
{

}
```

See Also

OnCallSessionClosed()

OnCallSessionClosed()

The OnCallSessionClosed() event triggers when VaxREC internally closes a call-session. This event provides notification that a call session has been closed, allowing the application to respond or perform any necessary actions accordingly.

Syntax

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

Parameters

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

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

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

Example

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

}
```

See Also

OnCallSessionCreated()

OnIncomingCall()

The OnIncomingCall() event triggers when VaxREC receives a SIP-REC based call request.

Syntax

```
OnIncomingCall(SessionId, CallerName, CallerId, CalleeName, CalleeId,
               DialNo, FromPeerType, FromPeerName, UserAgentName,
               RecXML, FromIP, FromPort)
```

Parameters

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

CallerName (string)
This parameter specifies the caller name.

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

CalleeName (string)
This parameter specifies the callee name.

CalleeId (string)
This parameter specifies a unique identification of callee.

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.

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

RecXML (string)
This parameter specifies a XML part of SIP/SDP packet.

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

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

Example

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

}
```

See Also

AcceptCallSession(), OnCallSessionConnected, OnCallSessionFailed()

OnCallSessionConnected()

The OnCallSessionConnected() event triggers when VaxREC successfully established a Call-Session with SIP-REC client or remote SIP-REC SIP Server.

Syntax

```
OnCallSessionConnected(SessionId)
```

Parameters

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

Example

```
OnCallSessionConnected(SessionId)
{
	RecordWaveStartToCallSession(SessionId, "RecordCall.wav", 1)
}
```

Example

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

OnCallSessionRecordedWaveREC(SessionId, DataREC, SizeREC)
{
	//Create a file, and write data or process data
}
```

See Also

OnCallSessionRecordedWaveREC(), AcceptCallSession(), OnIncomingCall()

OnCallSessionLost()

The OnCallSessionLost() event triggers when VaxREC does not receive audio media packets for define interval of time.

Syntax

```
OnCallSessionLost(SessionId)
```

Parameters

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

Example

```
OnCallSessionLost(SessionId)
{
}
```

See Also

AudioSessionLost()

OnCallSessionHangup()

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

Syntax

```
OnCallSessionHangup(SessionId)
```

Parameters

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

Example

```
OnCallSessionHangup(SessionId)
{
}
```

See Also

AcceptCallSession(), OnCallSessionConnected(), OnIncomingCall()

OnCallSessionTimeout()

The OnCallSessionTimeout() event triggers when VaxREC fails to establish a Call-Session and does not receive a response from the SIP-REC client within the specified time period, as set in the AcceptCallSession() method. This event serves as a notification that the Call-Session establishment process has timed out, allowing for appropriate handling in the application.

Syntax

```
OnCallSessionTimeout(SessionId)
```

Parameters

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

Example

```
OnCallSessionTimeout(SessionId)
{
}
```

See Also

AcceptCallSession()

OnCallSessionCancelled()

The OnCallSessionCancelled() event triggers when the caller cancels the call before it is accepted by VaxREC. This event provides notification that a call session has been canceled by the caller, allowing the application to handle this event accordingly.

Syntax

```
OnCallSessionCancelled(SessionId)
```

Parameters

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

Example

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

See Also

OnOutgoingDiagnosticLog()

The OnOutgoingDiagnosticLog() event triggers when VaxREC 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 VaxREC 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()

OnVaxServerTick()

The OnVaxServerTick() event triggers after a specified time interval set by VaxServerStartTick() function.

Syntax

```
OnVaxServerTick(TickId)
```

Parameters

TickId (integer)
This parameter specifies the unique tick identification.

Example

```
OnVaxServerTick(TickId)
{
}
```

See Also

VaxServerStartTick(), VaxServerStopTick()

OnCallSessionRecordedWaveREC()

When recording on a specific call session starts using RecordWaveStartToCallSession() with an empty string as the second parameter, it will continue until the call disconnects or the recording is manually stopped using RecordWaveStopToCallSession().

This setup indicates that the recording should be associated with the ongoing call session until explicitly stopped or until the call is disconnected.

Syntax

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

Parameters

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

DataREC (array)
: The value of this parameter specifies the recorded wave (.wav) formated data.

SizeREC (integer)
: The value of this parameter specifies the size of recorded data.

Example

```
OnCallSessionRecordedWaveREC(SessionId, DataREC, SizeREC)
{

}
```

See Also

RecordWaveStartToCallSession()

LIST OF ERROR CODES

200	Failed to initialize RTP Socket.
201	Failed to allocate RTP port or provided value of RTP listen IP is incorrect.
202	Failed to create RTP task manager.
203	Failed to start media thread.
204	Unable to create RTP communication manager.
205	Unable to run RTP communication manager.
206	Unable/failed to open file.
207	Unable to read file.
208	Incorrect wave file format, please use 8000Hz, 16bit, mono uncompressed wave file.
209	Provided wave id is not valid.
210	Unable to start recording manager.
211	Voice session is not connected or started yet.
212	Failed to initialize VaxVoIP Library.
213	Unable to construct SIP SDP body.
214	Unable to construct SIP INVITE request.
215	Error to initialize SIP communication layer.
216	Failed to open SIP listen port or provided SIP listen IP is incorrect.
217	Failed to create SIP task manager.
218	Unable to create SIP REGISTER request.
219	Unable to create SIP UN-REGISTER request.
220	Unable to create SIP BYTE request.
221	Unable to create SIP CANCEL request.
222	Provided SessionId does not exist.
223	Voice session does not exist.
224	Provided login does not exist.
225	Login length is incorrect.
226	Provided line does not exist.
227	Can't send SIP response.
228	Invalid SIP response code, please check SIP RFC 3261.
229	Error to create SIP response.
230	Provided line already exist.
231	Incorrect RegId or RegId does not exist.
232	Error to create SIP response.
233	User is not registered.
234	Invalid codec OR no codec found for voice streaming.
235	Invalid DTMF type.
236	Remote party does not support RFC2833 DTMF digits.
237	Error to create REFER request to transfer the call.
238	Error to create event handler.
239	License key is not valid or expired.
240	Invalid digit for DTMF.
241	Invalid proxy URI.
242	Line is not registered.
243	Invalid SIP To-URI.

244	Desired operation can only be performed on connected session.
245	Can't perform desired operation on connected session.
246	Direct communication is not supported.
247	One of the provided parameter(s) value is not correct.
248	Invalid chat message-Id.
249	Invalid subscribe-Id.
250	Failed to generate a unique-Id.
251	Provided unique-Id is not valid.
252	Provided unique-Id or value already exist.
253	Provided unique-Id or value does not exist.
254	Failed to create session.
255	Failed to enable DTMF detection.
256	Error to process transfer request.
257	Provided ListenIP is already binded.
258	Provided ListenIP does not exist in Server Key.
259	Provided SessionId has No Free Channel.
260	Both users (pickup and ringing) should be members of same pickup group.
261	Crypto audio or video media mismatched.
262	Unable to create socket dispatcher.
263	Unable to post message to socket dispatcher.
264	Provided addr is not valid.
265	Unable to bind socket addr or IP.
266	Failed to allocate socket port or provided value of listen IP or port is already in use.
267	General socket or network failure.
268	Failed to add wait-event to socket dispatcher.
269	Unable to create socket (timeout).
270	Line catch-all can't be used.
271	Route prefix conflicts with another route.
272	Unable to find provided SessionId.
273	Unable to find incoming call or inbound call does not exist.
274	Inbound call already connected.
275	Unable to connect socket. Provided address or port is not valid.
276	Unable to capture socket. Provided address or port is not valid.
277	User already attached.
278	Unable to create thread dispatcher.
279	Unable to post message to thread dispatcher.
280	Unable to process to thread dispatcher.
281	Terminating abnormally thread dispatcher.
282	Unable to create pipe dispatcher.
283	Unable to post message to pipe dispatcher.
284	Unable to process to pipe dispatcher.
285	Failed to add wait-event to pipe dispatcher.
286	General pipe communication failure.
287	Unable to create pipe (timeout).

LIST OF SIP RESPONSES (SIP RFC 3261)

Provisional responses 1xx			
100	Trying	180	Ringing
181	Call Is Being Forwarded	182	Queued
183	Session Progress		
Redirection 3xx			
300	Multiple Choices	301	Moved Permanently
302	Moved Temporarily	305	Use Proxy
380	Alternative Service		
Request Failure 4xx			
400	Bad Request	401	Unauthorized
402	Payment Required	403	Forbidden
404	Not Found	405	Method Not Allowed
406	Not Acceptable	407	Proxy Authentication Required
408	Request Timeout	410	Gone
413	Request Entity Too Large	414	Request-URI Too Long
415	Unsupported Media Type	416	Unsupported URI Scheme
420	Bad Extension	421	Extension Required
423	Interval Too Brief	480	Temporarily Unavailable
481	Call/Transaction Does Not Exist	482	Loop Detected
483	Too Many Hops	484	Address Incomplete
485	Ambiguous	486	Busy Here
487	Request Terminated	488	Not Acceptable Here
491	Request Pending	493	Undecipherable
Server Failure 5xx			
500	Server Internal Error	501	Not Implemented
502	Bad Gateway	503	Service Unavailable
504	Server Time-out	505	Version Not Supported
513	Message Too Large		
Global Failures 6xx			
600	Busy Everywhere	603	Decline
604	Does Not Exist Anywhere	606	Not Acceptable

SIP CLIENT REGISTRATION FLOW

Please see SIP CLIENT REGISTRATION FLOW document for further details.