A MITEL
PRODUCT
GUIDE

MiContact Center Enterprise

Agent Service Open Interface - Description

Release 9.8
Document Version 1.0

July 2025

Mitel
Powering connections

Notices

The information contained in this document is believed to be accurate in all respects but is not warranted by **Mitel Networks™ Corporation (MITEL®).**

The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes. No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

Trademarks

The trademarks, service marks, logos and graphics (collectively “Trademarks”) appearing on Mitel's Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC) or its subsidiaries (collectively "Mitel") or others. Use of the Trademarks is prohibited without the express consent from Mitel. Please contact our legal department at legal@mitel.com for additional information. For a list of the worldwide Mitel Networks Corporation registered trademarks, please refer to the website: http://www.mitel.com/trademarks.

®,™ Trademark of Mitel Networks Corporation

INTRODUCTION

The MiContact Center Agent Service provides an open interface that is implemented in a COM (Component Object Model) object, allowing clients to connect to the Agent Service and receive events about agent activity. The purpose of this interface is to allow integration with MiCC Enterprise on the server side, rather than at each individual desktop client.

COM OBJECT

The COM interface is implemented in a DLL named CCASComClient.dll. The object implements an API (Application Programming Interface) for client applications to connect to the Agent Service and query initial configuration data. It also provides events to inform all connected clients about agent activity.

The COM object is installed, registered, and ready to use on the MiCC Enterprise server. To use the COM object on another machine, do the following:

1. Create a new folder on the target machine.
2. Copy the following files from the MiCC Enterprise server to the new folder on the target machine

   C:\Program Files (x86)\Common Files\EricssonShare\NextCCShare\CCASComClient.dll
   C:\Program Files (x86)\Common Files\EricssonShare\Socketmanager.dll
   C:\Program Files (x86)\Common Files\EricssonShare\sectracelog.dll

3. Open a command window on the machine, change to the folder where the files were copied, and enter the following command:

   regsvr32.exe CCASComClient.dll

4. If your application will not be running as a Windows service, please add the following Registry value:
   HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mitel\SeC\Common\Parameters\RunningAsService
5. Set the value to 0
6. If the MiCC Enterprise server is configured to use Native Security (SSL) then add the following Registry value:
   HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mitel\SeC\Common\Parameters\UseSSLClient
7. Set the value to 1

INTERFACES

The interfaces are the public COM methods. Applications call the various methods to request an action to be executed. Table 1 provides a detailed description for each interface.

Table 1

INTERFACE	DESCRIPTION
Initialize()	Initializes the connection to the COM object. This is the first method that should be called. **Return Values**: Always returns S_OK
Connect([in] BSTR bstrCCASMac hineName, [in] LONG lCCASPort)	Tells the COM object to attempt to connect to the Agent Service running on the machine name provided, at the port number indicated. The COM object will attempt to make the connection. If no response is received within 30 seconds from the Agent Service, a failure will be returned. Note that this is not supported for tenanted systems **Return Values:** S_OK – Indicates connected to the Agent Service E_FAIL – Indicates failure to connect to the Agent Service
ConnectWithI PAddress([in] BSTR bstrCCASMachineName, [in] LONG lCCASPort, [in] BSTR bstrNICIPAddress)	Same as Connect, except it allows the client to specify the IP address of the Network Interface Card to be used to connect to the Agent Service. Note that this is not supported for tenanted systems. **Return Values:** S_OK – Indicates connected to the Agent Service E_FAIL – Indicates failure to connect to the Agent Service
ConnectWithDirectConnect([in] BSTR bstrCCASMachineName, [in] LONG lCCASPort, [in] BSTR bstrNICIPAddr ess)	Connects to the Agent Service running on the machine name provided, at the port number indicated. This interface must be used if requests will be sent to the MiContact Center Agent Service using the Open Interface API. If a specific IP address is to be used to connect to the Agent Service, it can be provided in the bstrNICIPAddress field; otherwise, an empty string should be provided. Using this interface causes a direct TCP/IP connection to be established and maintained with the Agent Service. The multicast address will not be connected to. Note that only a limited number of clients (up to 10) may connect to the Agent Service via this interface. Note that this API must be used for tenanted systems. **Return Values:** S_OK – Indicates connected to the Agent Service E_FAIL – Indicates failure to connect to the Agent Service
Uninitialize()	Tells the COM object to disconnect from the Agent Service. Once uninitialized, no further events will be received. Uninitialize() should always be called when the client is disconnecting. **Return Values:** Always returns S_OK

INTERFACE	DESCRIPTION
GetLastErrorInfo([out] LONG *lError, [out] BSTR *bstrErrorString)	Returns the error value and a descriptive string of the last error occurring. **Return Values:** Always returns S_OK
SetLogonCredentials([in] BSTR bstrUserId, [in] BSTR bstrPassword, [in] LONG lTenantID) This method is only required when tenanting is configured for the MiCC Enterprise system.	For a tenanted installation, allows the client to specify the user ID, password and tenant identification. This method should be called prior to ConnectWithDirectConnect() for tenanted installations. The user Id and password fields provided can be any of the following: - Host Administrator user Id and password, Tenant Id = -1 Events for all tenants will be received - Tenant Administrator user Id and password, Tenant Id = tenant's assigned Id Events for the specified tenant will be received No other logon credentials other than Host Administrator or Tenant Administrator will be accepted. The ConnectWithDirectConnect option must be specified; events will not be generated on the multicast interface for tenanted systems. **Return Values**: Always returns S_OK
GetAgentRecIDFromLogonID([in] BSTR bstrAgentLogonID, [out] LONG* lAgentRecID)	Retrieves the database record ID for the provided agent logon ID. **Return Values:** S_OK – Agent record ID found E_FAIL – Agent unknown or not connected to Agent Service
MakeCall([in] LONG lAgentRecID, [in] BSTR bstrDialString)	Makes a call from the agent indicated by the provided record ID to the destination entered. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
AnswerCall([in] LONG lAgentRecID, [in] LONG lCallID)	Answers the indicated call ringing at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
HangupCall([in] LONG lAgentRecID, [in] LONG lCallID)	Hangs up the indicated call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service

INTERFACE	DESCRIPTION
HangupSession([in] LONG lAgentRecID, [in] LONG lSessionID)	Disconnects the indicated session at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. The session disconnected can be an Open Media, E-mail, or SMS session. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
HoldCall([in] LONG lAgentRecID, [in] LONG lCallID)	Holds the indicated call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
RetrieveCall([i n] LONG lAgentRecID, [in] LONG lCallID)	Retrieves the indicated held call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
ConferenceCall([in] LONG lAgentRecID)	Creates a conference with existing calls at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
TransferCall([in] LONG lAgentRecID, [in] LONG lHeldCallID)	Transfers the indicated held call to the active call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
DeflectCall([in] LONG lAgentRecID, [in] BSTR bstrDest, [in] LONG lCallID)	Deflects the indicated call at the agent indicated by the provided record ID to the destination provided. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service

INTERFACE	DESCRIPTION
RejectService Call([in] LONG lAgentRecID, [in] LONG lCallID)	Rejects the indicated incoming service group call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
CancelClerical ([in] LONG lAgentRecID, [in] LONG lCallID)	Cancels clerical state for the indicated call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
CancelSessionClerical ([in] LONG lAgentRecID, [in] LONG lSessionID)	Cancels clerical state for the indicated session at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. The session can be an Open Media, E-mail, or SMS session. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
ExtendClerical ([in] LONG lAgentRecID, [in] LONG lCallID, [in] LONG lNumSeconds)	Extends clerical time on the indicated agent by the number of seconds provided. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
SetVoiceRead y([in] LONG lAgentRecID)	Sets the indicated agent to Voice Ready state. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
SetVoiceNotReady([in] LONG lAgentRecID, [in] LONG lReason)	Sets the indicated agent to Voice Not Ready state. If the record ID of a defined reason is provided, the reason will be recorded with the Not Ready status. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service

INTERFACE	DESCRIPTION
GetVoiceReadyStatus([in] LONG lAgentRecID, [out] LONG* lStatus)	Retrieves the current Voice Ready state for the indicated agent. The status is provided as 0 (Not Ready) or 1 (Ready). The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
SetEmailReady([in] LONG lAgentRecID)	Sets the indicated agent to Email Ready state. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
SetEmailNotReady([in] LONG lAgentRecID)	Sets the indicated agent to Email Not Ready state. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service
GetEmailReadyStatus([in] LONG lAgentRecID, [out] LONG* lStatus)	Retrieves the current Email Ready state for the indicated agent. The status is provided as 0 (Not Ready) or 1 (Ready). The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service

INTERFACE	DESCRIPTION
GetNotReadyReasons([in] long lTenantRecID, [out] BSTR* bstrReasons)	Returns the list of defined Not Ready reasons for the tenant provided. Pass -1 if the system is non-tenanted. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
LogonPhoneAgent([in] BSTR bstrPIN, [in] BSTR bstrExtension, [in] LONG lTenantID, [in] LONG lOASID)	Allows the user to logon a Phone Agent with the indicated PIN and extension. Provide the tenant ID of the tenant that the agent is defined on, or -1 for a non-tenanted system, and the ID of the OAS or TAS Server that the extension will be monitored on. If the Phone Agent is already logged on to a different extension, no action will be taken. **Return Values:** S_OK – Request sent to Router Service E_FAIL – Not connected to Agent Service
LogoffPhoneAgent([in] BSTR bstrPIN, [in] LONG lTenantID)	Allows the user to logoff a Phone Agent with the indicated PIN. Provide the tenant ID of the tenant that the agent is defined on, or -1 for a non-tenanted system. **Return Values:** S_OK – Request sent to Router Service E_FAIL – Not connected to Agent Service
SetPhoneAgentReady([in] BSTR bstrPIN, [in] LONG lTenantID)	Allows the user to change the status of the Phone Agent logged on with the indicated PIN to Voice Ready. Provide the tenant ID of the tenant that the agent is defined on, or -1 for a non-tenanted system. **Return Values:** S_OK – Request sent to Router Service E_FAIL – Not connected to Agent Service
SetPhoneAgentNotReady([in] BSTR bstrPIN, [in] LONG lTenantID, [in] LONG lReasonID)	Allows the user to change the status of the Phone Agent logged on with the indicated PIN to Voice Not Ready. Provide the tenant ID of the tenant that the agent is defined on, or -1 for a non-tenanted system. A Not Ready Reason can also be provided by passing the record ID of the reason in the lReasonID field. Otherwise, pass 0. **Return Values:** S_OK – Request sent to Router Service E_FAIL – Not connected to Agent Service
GetPhoneAgentVoiceReadyStatus([in] LONG lAgentRecID, [out] VARIANT* val)	Returns the voice ready status of a Phone Agent. If the request succeeds, the result is returned in the long member (lVal) of the Variant structure. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on

GetPhoneAgentRecIDFromExtension([in] BSTR bstrExtension, [out] VARIANT* val)	Returns the record ID of the Phone Agent when provided with the extension that the Phone Agent is currently logged onto. If the request succeeds, the result is returned in the long member (lVal) of the Variant structure. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on
GetPhoneAgentServiceGroupID([in] BSTR bstrExtension, [out] VARIANT* val)	Returns the record ID of the Service Group for the active call of a Phone Agent when provided with the extension that the Phone Agent is currently logged onto. If the request succeeds, the result is returned in the long member (lVal) of the Variant structure. If the agent is not currently handling a service group call, the value returned is 0. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on
GetAgentServiceGroups([in] LONG lAgentRecID, [out] BSTR* bstrGroups)	Returns a list of the service groups that the MiContact Center Agent is currently skilled to serve. The groups are returned in the format of Record ID | Group Name. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on
GetPhoneAgentstatus([in] LONG lAgentRecID, [out] VARIANT BOOL* loggedOn, [out] VARIANT BOOL* ready, [out] LONG* notReadyReasonID, [out]LONG* callState, [out] LONG* serviceGroupID, [out] LONG* callID)	Returns information about the Phone Agent's current status, including logon status, voice ready status, not ready reason, call state, and current service group and call ID if a service group call is in progress. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on

SetAgentCQCodes([in] long lAgentRecID, [in] long lCallID, [in] long lServiceGroupID, [in] BSTR bstrCQCodes)	Sets the call qualification codes for a current agent call. The call must be in progress or in clerical state. bstrCQCodes contains a list of code/name pairs delimited by | which may be codes already defined or custom codes. There is a maximum of 100 codes that can be specified, but there may be no more than 50 already defined codes or 50 custom codes. The format of the code string is: Code1|Name1|Code2|Name2 The name may be blank. For already defined codes, the name is ignored. If the name is blank for a custom code, the code itself is used as the name. The placeholder for the name must still exist in the code string. Examples: Code1|Name1|Code2|Name2 Code1||Code2|Name2 Code1|Name1|Code2| **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on
SetPrivateData([in] long lAgentRecID, [in] long lCallID, [in] BSTR bstrPrivateData)	Sets private data for a current agent call. The call must be in active state. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service or agent not logged on
DeflectPhoneAgentCall([in] long lAgentRecID, [in] long lTenantID, [in] long lCallID, [in] BSTR bstrDest)	Deflects a call from the phone agent indicated by the provided record ID to the destination provided. Note that if the destination provided corresponds to the defined Customer Authentication number, the phone agent will remain in Talking state while the call is handled at the destination. For further details on the Customer Authentication feature, refer to the *Advanced Configurations* document (3_1543-LXA119154). **Return Values:** S_OK – Request sent to MiCC-E Router Service to be processed E_FAIL – Not connected to Agent Service
SetAgentIVRData([in] long lAgentRecID, [in] long lTenantID, [in] long lCallID, [in] BSTR IVRLabel1, [in] BSTR IVRData1, [in] BSTR IVRLabel2, [in] BSTR IVRData2, [in] BSTR IVRLabel3, [in] BSTR IVRData3)	Sets IVR data for the voice call indicated by the lCallID parameter. The voice call must exist at the agent indicated by the lAgentRecID parameter. Up to 3 IVR data fields can be set. If an IVR data field already exists with the provided IVR Label, it will be updated with the IVR Data provided. If the field doesn't exist, it will be added. IVR data can be set for Agents or Phone Agents. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service

GetAgentCallManagerID([in] long lAgentRecID, [out] VARIANT* val)	Returns the record ID of the Call Manager that the agent is currently logged on to. This request can be used for regular agents or phone agents. If the request succeeds, the result is returned in the long member (lVal) of the Variant structure. If the agent is not currently logged on, the value returned is 0. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SetAgentIVRDataWithCDR([in] LONG lAgentRecID, [in] long lTenantID, [in] long lCallID, [in] long lMediaType, [in] BSTR IVRLabel1, [in] BSTR IVRData1, [in] BSTR IVRLabel2, [in] BSTR IVRData2, [in] BSTR IVRLabel3, [in] BSTR IVRData3)	Sets IVR data for the session indicated by the lCallID parameter and produces the necessary CDR events. The session must exist at the agent indicated by the lAgentRecID parameter. Up to 3 IVR data fields can be set. If an IVR data field already exists with the provided IVR Label, it will be updated with the IVR Data provided. If the field doesn't exist, it will be added. IVR data can be set for Agents or Phone Agents. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SetChatReady([in] long lAgentRecID)	Sets the agent ready for chat sessions. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SetChatNotReady([in] long lAgentRecID)	Sets the agent not ready for chat sessions. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SetOpenMediaReady([in] long lAgentRecID)	Sets the agent ready for open media sessions. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service

SetOpenMediaNotReady([in] long lAgentRecID)	Set the agent not ready for open media sessions. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
StopApp([in] long lAgentRecID)	Shuts down the Agent application. Application is forcibly closed with no warning prompts. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
MakeCampaignCall([in] long lAgentRecID, [in] long lCampaignID, [in] long lCustomerID, [in] long lCampaignCallID, [in] BSTR bstrDialString)	Initiates the dialing process for a pending campaign call indicated by the lCampaignCallID parameter. The call must exist at the agent indicated by the lAgentRecID parameter. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
RejectCampaignCall([in] long lAgentRecID, [in] long lCampaignID, [in] long lCustomerID, [in] long lReason)	Rejects the pending campaign call residing at the agent indicated by the lAgentRecID parameter. Return Values: S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SetCampaignCallStatus([in] long lAgentRecID, [in] long lCampaignCallID, [in] long lStatus, [in] DATE dNextCallAttempt, [in] VARIANT_BOOL bCloseForm, [in] VARIANT_BOOL bUpdateComment, [in] BSTR bstrComment)	Set the status of an active campaign call and closes the status form if necessary. The dialing process must have already been initiated for the campaign call. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service

GetAgentMediaSettings([in] LONG lAgentRecID, [out] LONG* lNumberOfConcurrentOM, [out] LONG* lReceiveChatWhileHandlingOM, [out] LONG* lReceiveEmailWhileHandlingOM, [out] LONG* lReceiveVoiceWhileHandlingOM,[out] LONG* lReceiveChatWhileHandlingVoice, [out] LONG* lReceiveEmailWhileHandlingVoice, [out] LONG* lReceiveOMWhileHandlingVoice, [out] LONG* lNumberOfConcurrentChat, [out] LONG* lReceiveVoiceWhileHandlingChat, [out] LONG* lReceiveEmailWhileHandlingChat, [out] LONG* lReceiveOMWhileHandlingChat, [out] LONG* lNumberOfConcurrentEmail, [out] LONG* lReceiveVoiceWhileHandlingEmail, [out] LONG* lReceiveChatWhileHandlingEmail, [out] LONG* lReceiveOMWhileHandlingEmail, [out] LONG* lChatEnabled, [out] LONG* lEmailEnabled, [out] LONG* lSMSEnabled, [out] LONG* lOMEnabled);)	Retrieves the current media settings including the max number of concurrent open media, chat and email sessions, as well as media interaction settings for the current agent and whether chat, email, SMS and open media are enabled for the agent to receive interactions of that type. Note that this information is only available for logged on MiCC-E Agents. Otherwise, the information returned will all have a value of 0. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
MakeCallbackCall([in] long lAgentRecID, [in] long lCallbackID, [in] long lCallbackCallID, [in] BSTR bstrDialString)	Initiates the dialing process for a pending callback call indicated by the lCallbackCallID parameter. The call must exist at the agent indicated by the lAgentRecID parameter. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
RejectCallbackCall([in] long lAgentRecID, [in] long lCallbackID, [in] long lReason)	Rejects the pending callback call residing at the agent indicated by the lAgentRecID parameter. Return Values: S_OK – Request succeeded E_FAIL – Not connected to Agent Service

SetCallbackCallStatus([in] long lAgentRecID, [in] long lCallbackCallID, [in] long lStatus, [in] DATE dNextCallAttempt, [in] VARIANT_BOOL bCloseForm)	Set the status of an active callback call and closes the status form if necessary. The dialing process must have already been initiated for the callback call. **Return Values:** S_OK – Request succeeded E_FAIL – Not connected to Agent Service
SendDTMF([in] long lAgentRecID, [in] long lCallID, [in] BSTR bstrDigits)	Sends the provided DTMF digits through the indicated call at the agent indicated by the provided record ID. The agent must be logged on to MiContact Center Agent to complete this request. **Return Values:** S_OK – Request sent to agent application E_FAIL – Not connected to Agent Service

EVENTS

The object communicates with the Agent Service through a COM connection point mechanism. Event handlers in the client application are called in response to events generated by the object.

There are only two types of events provided by the object: OnEvent, and OnDisconnect. OnDisconnect is generated when the connection to the Agent Service is lost. To reconnect, it is necessary to call Connect() on the object.

The OnEvent generates a generic event object that contains different information depending on the type of event that it contains. To retrieve information about the event, the client application can use the interfaces described in Table 2.

Table 2 Events

INTERFACE	DESCRIPTION
GetType([out] LONG* lEventType)	Returns the type of event that is provided. For a list of event types, see below. **Return Values:** Always returns S_OK
GetStringValue([in] BSTR bstrName, [out] BSTR bstrValue)	For a particular event type, returns the value of the event parameter passed in bstrName. For example, to retrieve the agent's name from an event of type BROADCAST_LOGON, pass the value "Name" in the bstrName parameter, and bstrValue will return the agent's defined user name. **Return Values:** If the value exists, it is returned in bstrValue and the return value is S_OK. Otherwise, the return value is E_FAIL.
GetLongValue([in] BSTR bstrName, [out] LONG* lValue)	For a particular event type, returns the value of the event parameter passed in bstrName. For example, to retrieve the agent's ID from an event of type BROADCAST_LOGON, pass the value "RecID" in the bstrName parameter, and lValue will return the agent's defined record ID. **Return Values:** If the value exists, it is returned in lValue and the return value is S_OK. Otherwise, the return value is E_FAIL.

Immediately after successfully connecting to the Agent Service, initial configuration events will automatically be sent to the application, with information about currently logged on MiContact Center Agents and configured service groups. The event types that will be generated are BROADCAST_LOGON and BROADCAST_SERVICEGROUP_ADDED.

If the MiCC Enterprise system is configured to generate events for Phone Agents, BROADCAST_LOGON events will be generated for all logged on Phone Agents as well.

Once all configuration data has been sent, the event BROADCAST_DATA_COMPLETE will be sent, indicating that all initial data has been provided.

EVENT INTERFACES

Table 3 Event Interfaces provides a detailed description for each event type that can be generated by OnEvent, and the parameters it contains.

Table 3 Event Interfaces

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_LOGON (1)	Returns information about the MiContact Center Agent logging on. This event type is also sent with the initial configuration data.	Name	String	Agent's name
		DN	String	Agent's extension
		RecID	Long	Agent's record ID
		LogonID	String	Agent's logon ID
		VoiceReady	Long	Flag indicating whether the agent is ready to receive voice calls.
		EmailReady	Long	Flag indicating whether the agent is ready to receive e-mail sessions
		OpenMediaReady	Long	Flag indicating whether the agent is ready to receive Open Media sessions
		MediaReady	Long	Flag indicating whether the agent is ready to receive chat sessions
		MachineName	String	Name of the machine the agent is logging onto.
		NumberOfServiceGroups	Long	The number of service groups this agent is able to serve

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		ServiceGroupList	String	List of service group record IDs that this agent is able to serve. Service groups are separated by the character |. Ex: 5|4|23
		OASID	Long	The OAS/TAS Server ID that this agent is connected to.
		PhoneAgent	Long	Indicates whether the user is a phone agent (1) or not (0). Note that this flag is only sent in the initial events sent after connecting to the CCAS interface, not in the event generated when an agent logs on while connected to the CCAS interface. In that case the BROADCAST_LOGON_PHONEAGENT event can be used to indicate that a phone agent logged on.
		TenantID	Long	The TenantID that this agent is assigned to.
BROADCAST_LOGOFF (2)	Indicates a MiContact Center Agent has logged off.	RecID	Long	Agent's record ID

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_READY (3)	Indicates a MiContact Center Agent is Ready to receive voice service group calls.	RecID	Long	Agent's record ID
BROADCAST_NOTREADY (4)	Indicates a MiContact Center Agent is Not Ready to receive voice service group calls.	RecID	Long	Agent's record ID
		ReasonID	Long	Record ID of the reason for not ready; -1 if no reason provided
		ReasonString	String	Not Ready reason description; empty if no reason provided
BROADCAST_EMAILREADY (5)	Indicates a MiContact Center Agent is Ready to receive e-mail sessions.	RecID	Long	Agent's record ID
BROADCAST_EMAILNOTREADY (6)	Indicates a MiContact Center Agent is Not Ready to receive e-mail sessions.	RecID	Long	Agent's record ID
		ReasonID	Long	Record ID of the reason for not ready; -1 if no reason provided
		ReasonString	String	Not Ready reason description; empty if no reason provided
BROADCAST_ORIGINATED (7)	Indicates the agent has initiated a call.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call
		ServiceGroupID	Long	Record ID of the service group, if this call is associated with a service group
		CallType	Long	Type of the call, defined as: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_DELIVERED (8)	Indicates that an incoming call has arrived at the agent, or an outbound call from the agent is ringing the opposite party.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call
		OppositeParty Number	String	Identity of the calling/called party
		TrunkID	String	Identifier of the trunk, if provided
		CalledNumber	String	Number originally dialed. Note that this value is not provided for phone agents.
		PrivateData	String	Private Data associated with the call, if provided
		Cause	Long	Indicates the cause associated with the call, if any.
BROADCAST_ESTABLISHED (9)	Indicates that the agent is connected to the opposite party.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call
		PrivateData	String	Private Data associated with the call, if provided
		TrunkID	String	Identifier of the trunk, if provided
		CallSubstate	Long	Indicates the substate of the call, as follows: Talking=0 Assisting=1 Assisted=2 Monitoring=3 Monitored=4

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		OppositePartyNumber	String	Identity of the calling/called party
		Cause	Long	Indicates the cause associated with the call, if any.
		MediaServerSessionUri	String	Indicates the URI of the media server session for integration with external recorders.
BROADCAST_HELD (10)	Indicates that the agent has placed a call on hold, or has been placed on hold.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call
		OnHold	Long	Flag indicating if the agent has been placed on hold
BROADCAST_RETRIEVED (11)	Indicates that the agent has retrieved a previously held call, or has been retrieved.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call
		MediaServerSessionUri	String	Indicates the URI of the media server session for integration with external recorders
BROADCAST_TRANSFERRED (12)	Indicates the agent has transferred a held call to an active call or	RecID	Long	Agent's record ID

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
	has been transferred.	HeldCallID	Long	Call ID of the previously held call.
		ActiveCallID	Long	Call ID of the previously active call.
		NewCallID	Long	Call ID of the new call, after the transfer.
		ClericalFlag	Long	Flag indicating whether the agent is now entering clerical state.
		TransferringAgent	Long	Flag indicating whether this agent performed the transfer.
		OppositePartyNumber	String	Identity of the opposite party
		PrivateData	String	Private Data associated with the call, if provided
		MediaServerSessionUri	String	Indicates the URI of the media server session for integration with external recorders. Not provided for the agent performing the transfer.
BROADCAST_CONFERENCED (13)	Indicates a conference has been initiated.	RecID	Long	Agent's record ID
		HeldCallID	Long	Call ID of the previously held call
		ActiveCallID	Long	Call ID of the previously active call
		NewCallID	Long	Call ID of the new call, after the conference
		OppositePartyNumber	String	Comma separated list of the other parties in the conference

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		MediaServerSessionUri	String	Indicates the URI of the media server session for integration with external recorders.
BROADCAST_CONNECTIONCLEARED (14)	Indicates the agent has cleared from a call.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call.
		ClericalFlag	Long	Flag indicating whether the agent is now entering clerical state.
		DivertedFlag	Long	Flag indication whether the agent diverted the call to another destination
		DivertDestination	String	If the call is diverted, indicates the destination to which the call was diverted
BROADCAST_CLERICALENDED (15)	Indicates the agent has exited Clerical state and is ready to receiveservice calls.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call.
BROADCAST_CALLREJECTED (16)	Indicates the agent has rejected a service group or callback call.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the call.

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Callback Ring Timeout Callback Rejected By Agent Agent Logged Off Callback Error Campaign Error Unknown Cause
BROADCAST_CALLBACKACCEPT (17)	Indicates the agent has accepted a callback call and will initiate the callback.	RecID	Long	Agent's record ID
		OrigCallID	Long	Call ID of the original call
		CallID	Long	Call ID of the initiated callback call
		CallType	Long	Type of the callback, defined as: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
		ServiceGroupID	Long	Record ID of the associated service group
BROADCAST_CALLBACKREJECT (18)	Indicates the agent has rejected a callback call and will not initiate the callback.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the original call

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		CallType	Long	Type of the callback, defined as: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Callback Ring Timeout Callback Rejected By Agent Agent Logged Off Callback Error Unknown Cause
BROADCAST_CALLBACKSTATUS (19)	Indicates the status of a complete callback call.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the callback call
		CallType	Long	Type of the callback, defined as: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
		CallbackStatus	Long	Flag indicating whether the callback succeeded (Value = 1) or failed (Value = 0).

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		Cause	String	Status of the callback, defined as: Succeeded No Answer Busy Not Available Wrong Number
BROADCAST_EMAILREJECT (20)	Indicates the agent has rejected an allocated e-mail.	RecID	Long	Agent's record ID
		MessageID	Long	ID of the e-mail message
		ServiceGroupID	Long	Record ID of the e-mail service group
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Agent Logged Off Unknown Cause
BROADCAST_EMAILDELETE (21)	Indicates the agent has deleted an allocated e-mail.	RecID	Long	Agent's record ID
		MessageID	Long	ID of the e-mail message
		ServiceGroupID		Record ID of the e-mail service group
		ClericalFlag	Long	Indicates whether the agent will enter clerical state.
BROADCAST_EMAILREPLY (22)	Indicates the agent has replied to an allocated e-mail.	RecID	Long	Agent's record ID

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		MessageID	Long	ID of the e-mail message
		ServiceGroupID	Long	Record ID of the e-mail service group
		Subject	String	Subject of the e-mail
		ForwardFlag	Long	Indicates whether the e-mail was forwarded; if so, the value is set to 1, otherwise, it is 0.
		RecipientList	String	List of recipients for the e-mail or SMS. For e-mails, this is only the **To** addresses and is limited to a maximum of 10 addresses.
		EmailText	String	Text of the SMS sent. Only applicable for SMS messages.
		ClericalFlag	Long	Indicates whether the agent will enter clerical state.
BROADCAST_EMAILFORWARD (23)	Indicates an e-mail or SMS message has been forwarded to another service group	RecID	Long	Agent's record ID
		MessageID	Long	ID of the e-mail or SMS message
		ServiceGroupID	Long	Record ID of the service group to which the e-mail or SMS is forwarded
BROADCAST_CALLINFORMATION (24)	Indicates a service group call has been allocated to the agent.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the service group call
		ServiceGroupID	Long	Record ID of the service group

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		CallingPartyNumber	String	Number of the calling party
		CalledNumber	String	Originally called number
		IVRLabel1	String	Label for Data Field 1 from the IVR
		IVRData1	String	Data for Data Field 1 from the IVR
		IVRLabel2	String	Label for Data Field 2 from the IVR
		IVRData2	String	Data for Data Field 2 from the IVR
		IVRLabel3	String	Label for Data Field 3 from the IVR
		IVRData3	String	Data for Data Field 3 from the IVR
		IVRLabel4	String	Label for Data Field 4 from the IVR
		IVRData4	String	Data for Data Field 4 from the IVR
		IVRLabel5	String	Label for Data Field 5 from the IVR
		IVRData5	String	Data for Data Field 5 from the IVR
		IVRLabel6	String	Label for Data Field 6 from the IVR
		IVRData6	String	Data for Data Field 6 from the IVR

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRLabel7	String	Label for Data Field 7 from the IVR
		IVRData7	String	Data for Data Field 7 from the IVR
		IVRLabel8	String	Label for Data Field 8 from the IVR
		IVRData8	String	Data for Data Field 8 from the IVR
		IVRLabel9	String	Label for Data Field 9 from the IVR
		IVRData9	String	Data for Data Field 9 from the IVR
		IVRLabel10	String	Label for Data Field 10 from the IVR
		IVRData10	String	Data for Data Field 10 from the IVR
		TimeInQueue	Long	Time (in seconds) this call waited in queue
		CallType	Long	Type of call: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
BROADCAST_CALLBACK (25)	Indicates a callback call has been allocated to this agent.	RecID	Long	Agent's record ID
		ServiceGroupID	Long	Record ID of the service group
		CallingPartyNumber	String	Number of the calling party
		CalledNumber	String	Originally called number

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRLabel1	String	Label for Data Field 1 from the IVR
		IVRData1	String	Data for Data Field 1 from the IVR
		IVRLabel2	String	Label for Data Field 2 from the IVR
		IVRData2	String	Data for Data Field 2 from the IVR
		IVRLabel3	String	Label for Data Field 3 from the IVR
		IVRData3	String	Data for Data Field 3 from the IVR
		CallType	Long	Type of callback: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
		OrigCallID	Long	Original call ID for the callback
BROADCAST_CAMPAIGNACCEPT (29)	Indicates the agent has accepted a campaign call and will initiate the customer call.	RecID	Long	Agent's record ID
		CampaignID	Long	Record ID of the campaign
		CustomerID	Long	Record ID of the customer to be called
		CallID	Long	Call ID of the initiated call
BROADCAST_CAMPAIGNREJECT (30)	Indicates the agent has rejected a campaign call and will not initiate the customer call.	RecID	Long	Agent's record ID
		CampaignID	Long	Record ID of the campaign

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		CustomerID	Long	Record ID of the customer
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Agent Logged Off Campaign Error Unknown Cause
BROADCAST_CAMPAIGNSTATUS (31)	Indicates the status of a campaign call, after it is completed.	RecID	Long	Agent's record ID
		CampaignID	Long	Record ID of the campaign
		CustomerID	Long	Record ID of the customer
		Cause	String	Status: Not Yet Called Busy No Answer Callback Later Completed Successfully Wrong Number
BROADCAST_CAMPAIGNINFO (32)	Indicates a campaign call has been allocated to an agent.	RecID	Long	Agent's record ID
		CampaignID	Long	Record ID of the campaign
		CustomerID	Long	Record ID of the customer
		CustomerName	String	Name of the customer

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		CustomerNumber	String	Number of the customer to be called
		Comment	String	Comments added by a prior agent, or empty if none.
		IVRLabel1	String	Label for first Campaign Customer Data field
		IVRData1	String	Data for first Campaign Customer Data field
		IVRLabel2	String	Label for second Campaign Customer Data field
		IVRData2	String	Data for second Campaign Customer Data field
		IVRLabel3	String	Label for third Campaign Customer Data field
		IVRData3	String	Data for third Campaign Customer Data field
		IVRLabel4	String	Label for fourth Campaign Customer Data field
		IVRData4	String	Data for fourth Campaign Customer Data field
		IVRLabel5	String	Label for fifth Campaign Customer Data field
		IVRData5	String	Data for fifth Campaign Customer Data field
		IVRLabel6	String	Label for sixth Campaign Customer Data field
		IVRData6	String	Data for sixth Campaign Customer Data field

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRLabel7	String	Label for seventh Campaign Customer Data field
		IVRData7	String	Data for seventh Campaign Customer Data field
		IVRLabel8	String	Label for eighth Campaign Customer Data field
		IVRData8	String	Data for eighth Campaign Customer Data field
		IVRLabel9	String	Label for ninth Campaign Customer Data field
		IVRData9	String	Data for ninth Campaign Customer Data field
		IVRLabel10	String	Label for tenth Campaign Customer Data field
		IVRData10	String	Data for tenth Campaign Customer Data field
BROADCAST_SERVICEGROUP_ADDED (33)	Configuration event indicating a service group has been added to the MiCC Enterprise system.	RecID	Long	Service Group record ID
		ServiceGroupPurpose	Long	Defined purpose of the group: 0 = Voice 1 = Campaign 2 = E-mail 3 = SMS 4 = Chat 5 = Voice Dispatch 6 = Common Hold 7 = E-mail Dispatch 8 = SMS Dispatch
		Name	String	Service Group name
BROADCAST_AGENT_SERVICEG	Indicates that the set of service groups this	RecID	Long	Record ID of the agent

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
ROUPS_CHANGED (34)	agent can serve has changed.	ServiceGroupList	String	List of service group record Ids that this agent is able to serve. Service groups are separated by the character |. Ex: 5|4|23
BROADCAST_DATA_COMPLETE (35)	Indicates the initial configuration data has been sent.	None		
BROADCAST_DISCONNECT (36)	Indicates the Agent Service has disconnected the client.	None		
BROADCAST_XFERCALLINFORMATION (37)	Information sent when an agent makes a consultation call to another agent with a service group call on hold. This information is sent to the consulted agent.	RecID	Long	Record ID of the agent
		HeldCallID	Long	Call ID of the held customer
		ActiveCallID	Long	Call ID of the consultation call
		ServiceGroupID	Long	Record ID of the service group this customer call is associated with
		CallingPartyNumber	String	Identity of the held customer
		CalledNumber	String	Originally dialed number
		IVRLabel1	String	Label for Data Field 1 from the IVR
		IVRData1	String	Data for Data Field 1 from the IVR
		IVRLabel2	String	Label for Data Field 2 from the IVR

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRData2	String	Data for Data Field 2 from the IVR
		IVRLabel3	String	Label for Data Field 3 from the IVR
		IVRData3	String	Data for Data Field 3 from the IVR
BROADCAST_IP AGENTINFO (38)	Port information sent when a MiContact Center Agent using an IP telephone or SIP softphone registers	RecID	Long	Record ID of the agent
		IPAddress	String	IP Address of the IP phone or agent machine
		RASPort	Long	Port number used for RAS
		CSPort	Long	Port number used for call signaling
BROADCAST_CA LLINFORMATION _UPDATE (39)	Updated call information sent when there is a change to the data initiated by the MiContact Center Agent or by the SetAgentIVRData request. This can also be sent for Campaign Customer Data fields if updated by MiContact Center Agent.	RecID	Long	Record ID of the agent
		CallID	Long	Call ID of the service group call
		CallType	Long	Type of call: 2 = Web Callback 3 = Regular voice call 4 = E-mail 5 = Campaign call
		IVRLabel1	String	Updated Label for Data Field 1
		IVRData1	String	Updated Data field 1
		IVRLabel2	String	Updated Label for Data Field 2 (if updated)
		IVRData2	String	Updated Data for Data Field 2 (if updated)

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRLabel3	String	Updated Label for Data Field 3 (if updated)
		IVRData3	String	Updated Data for Data Field 3 (if updated)
BROADCAST_ASSOCIATEDATA_SET (40)	Updated Associated Data (Private Data) sent when there is a change to the data initiated by the MiContact Center Agent.	RecID	Long	Record ID of the agent
		CallID	Long	Call ID of the service group call
		CallType	Long	Type of call: 2 = Web Callback 3 = Regular voice call 4 = E-mail
		PrivateData	String	Updated Private Data Field
BROADCAST_CQCODES (41)	Sent when call qualification codes are entered by the MiContact Center Agent.	RecID	Long	Record ID of the agent
		CallID	Long	Call ID of the service group call
		CallType	Long	Type of call: 2 = Web Callback 3 = Regular Voice Call 4 = E-mail 5 =Campaign Call
		CQCodeList	String	List of Call Qualification codes entered by the agent, separated by the | character.
BROADCAST_FAILED (42)	Sent when an outgoing call fails to be completed.	RecID	Long	Record ID of the agent
		CallID	Long	Call ID of the failed call

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		Cause	Long	Cause for failure, including the following: - Busy = 3 - Call Cancelled = 5 - Destination Not Obtainable = 13 Other causes may be sent if received from the switching system.
		CalledNumber	String	Number originally dialed
BROADCAST_RECORDINGSTARTED (43)	Sent when recording is initiated by an outside recording system, by a supervisor, or directly by the agent.	RecID	Long	Record ID of the agent
		DN	String	Extension that is being recorded
		IPAddress	String	IP Address of the agent being recorded, if logged on to an IP extension
		CallID	Long	Call ID of the call being recorded
		ServiceGroupID	Long	Service Group ID of the group that the call is associated with, if it is a service group call
		OppositePartyNumber	String	Number of the opposite party on the call
BROADCAST_RECORDINGSTOPPED (44)	Sent when recording is stopped.	RecID	Long	Record ID of the agent
		CallID	Long	Call ID of the call previously being recorded.
BROADCAST_OPENMEDIAREJECT (49)	Sent when an Open Media session is rejected.	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Open Media session

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Agent Logged Off Unknown Cause
BROADCAST_OPENMEDIAHANDLING (50)	Sent when an Open Media session has been opened by the agent and is being handled.	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Open Media session
BROADCAST_OPENMEDIACOMPLETE (51)	Sent when an agent has completed handling an Open Media session.	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Open Media session
		ClericalFlag	Long	Indicates whether the agent will enter clerical state.
BROADCAST_OPENMEDIANOTREADY (52)	Sent when an agent changes Open Media status to Not Ready	RecID	Long	Record ID of the agent
		ReasonID	Long	Record ID of the reason for not ready; -1 if no reason provided
		ReasonString	String	Not Ready reason description; empty if no reason provided
BROADCAST_OPENMEDIAREADY (53)	Sent when an agent changes Open Media status to Ready	RecID	Long	Record ID of the agent

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_CHATREJECT (55)	Sent when an agent rejects a chat session	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Chat session
		Cause	String	Reason for rejection, defined as follows: Ring Timeout Rejected By Agent Agent Logged Off Unknown Cause
BROADCAST_CHATHANDLING (56)	Sent when an agent is handling a chat session	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Chat session
BROADCAST_CHATCOMPLETE (57)	Sent when an agent completes handling a chat session	RecID	Long	Record ID of the agent
		SessionID	Long	Session ID of the Chat session

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		ClericalFlag	Long	Indicates whether the agent has entered clerical state for the chat session
BROADCAST_CHATREADY (58)	Sent when an agent changes Chat status to Ready	RecID	Long	Record ID of the agent
BROADCAST_CHATNOTREADY (59)	Sent when an agent changes Chat status to Not Ready	RecID	Long	Record ID of the agent
		ReasonID	Long	Record ID of the reason for not ready; -1 if no reason provided
		ReasonString	String	Not Ready reason description; empty if no reason provided
BROADCAST_EMAILREDIRECT (60)	Sent when an agent redirects an e-mail to another agent.	RecID	Long	Record ID of the agent previously handling the e-mail.

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		ToRecID	Long	Record ID of the agent receiving the e-mail.
		MessageID	Long	ID of the e-mail session
		ServiceGroupID	Long	Record ID of the service group associated with the e-mail.
		EmailText	String	Text of the e-mail
BROADCAST_CHATREDIRECT (61)	Sent when an agent redirects a chat session to another agent.	RecID	Long	Record ID of the agent previously handling the chat.
		ToRecID	Long	Record ID of the agent receiving the chat.

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		SessionID	Long	ID of the chat session
		ServiceGroupID	Long	Record ID of the service group associated with the chat session.
BROADCAST_EMAILHANDLING (62)	Sent when an agent begins handling an e-mail.	RecID	Long	Record ID of the agent
		MessageID	Long	ID of the e-mail message
BROADCAST_SMSHANDLING (63)	Sent when an agent begins handling an SMS session	RecID	Long	Record ID of the agent
		MessageID	Long	ID of the SMS message

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_OPENMEDIAREDIRECT (64)	Sent when an Open Media session is redirected to another agent	RecID	Long	Record ID of the agent previously handling the Open Media session
		ToRecID	Long	Record ID of the agent receiving the Open Media session
		SessionID	Long	ID of the Open Media session
		ServiceGroupID	Long	Record ID of the service group associated with the Open Media session.
BROADCAST_CHATFORWARD (65)	Sent when an agent forwards a chat session to another service group.	RecID	Long	Record ID of the agent previously handling the chat.
		SessionID	Long	ID of the chat session

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		ServiceGroupID	Long	Record ID of the service group that the chat session was forwarded to.
BROADCAST_OPENMEDIAFORWARD(66)	Sent when an agent forwards an open media session to another service group.	RecID	Long	Record ID of the agent previously handling the Open Media session
		SessionID	Long	ID of the Open Media session
		ServiceGroupID	Long	Record ID of the service group that the Open Media session was forwarded to.

EVENTS FOR PHONE AGENTS

It is possible to provide events via the Agent Service Open Interface for Phone Agents. The option can be configured in Configuration Manager. The following events will be generated for Phone Agents:

Phone Agent Logon/Logoff

Agent Ready/Not Ready for voice

Phone Agent Status – Idle, Busy, Clerical

Call Information for service group call allocated to Phone Agent

In addition, the following events from Table 3 will also be generated for Phone Agents. Note that duplicate events will be generated for Phone Agent logon, logoff, and status change. In the future, the events in Table 4 will be removed and only the events below will be generated for Phone Agents:

BROADCAST_LOGON

BROADCAST_LOGOFF

BROADCAST_READY

BROADCAST_NOTREADY

BROADCAST_IPAGENTINFO

BROADCAST_ORIGINATED

BROADCAST_DELIVERED

BROADCAST_ESTABLISHED

BROADCAST_HELD

BROADCAST_RETRIEVED

BROADCAST_TRANSFERRED

BROADCAST_CONFERENCED

BROADCAST_CONNECTIONCLEARED

BROADCAST_CLERICALENDED

BROADCAST_CALLINFORMATION

BROADCAST_CALLINFORMATION_UPDATE

BROADCAST_CQCODES

BROADCAST_FAILED

Table 4 Phone Agent events

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
BROADCAST_LOGON_PHONEAGENT (45)	Sent when a phone agent logs on to MiCC Enterprise.	RecID	Long	Record ID of the phone agent logging on.
		DN	String	Extension used by the phone agent.
BROADCAST_LOGOFFPHONEAGENT (46)	Sent when a phone agent logs out of MiCC Enterprise.	RecID	Long	Record ID of the phone agent logging off.
BROADCAST_PHONEAGENTSTATUS (47)	Sent when the call status of a phone agent changes.	RecID	Long	Record ID of the phone agent.
		CallSubstate	Long	Indicates the substate of the phone agent as follows: 1 = Idle 2 = Busy 3 = Clerical
BROADCAST_PHONEAGENTCALLINFORMATION (48)	Sent when a service group call is allocated to a Phone Agent.	RecID	Long	Agent's record ID
		CallID	Long	Call ID of the service group call
		ServiceGroupID	Long	Record ID of the service group
		CallingPartyNumber	String	Number of the calling party
		CalledNumber	String	Originally called number
		IVRLabel1	String	Label for the first data field of the IVR information
		IVRData1	String	Data for the first data field of the IVR information
		IVRLabel2	String	Label for the second data field of the IVR
		IVRData2	String	Data for the second data field of the IVR

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		IVRLabel3	String	Label for the third data field of the IVR information
		IVRData3	String	Data for the third data field of the IVR information
		IVRLabel4	String	Label for Data Field 4 from the IVR
		IVRData4	String	Data for Data Field 4 from the IVR
		IVRLabel5	String	Label for Data Field 5 from the IVR
		IVRData5	String	Data for Data Field 5 from the IVR
		IVRLabel6	String	Label for Data Field 6 from the IVR
		IVRData6	String	Data for Data Field 6 from the IVR
		IVRLabel7	String	Label for Data Field 7 from the IVR
		IVRData7	String	Data for Data Field 7 from the IVR
		IVRLabel8	String	Label for Data Field 8 from the IVR
		IVRData8	String	Data for Data Field 8 from the IVR
		IVRLabel9	String	Label for Data Field 9 from the IVR
		IVRData9	String	Data for Data Field 9 from the IVR
		IVRLabel10	String	Label for Data Field 10 from the IVR
		IVRData10	String	Data for Data Field 10 from the IVR

EVENT TYPE	DESCRIPTION	PARAMETERS		
		NAME	TYPE	VALUE
		TimeInQueue	Long	Time in seconds that the call waited in queue

SAMPLE SOURCE CODE

The COM API is structured around standard COM technology and is therefore designed to be easy to use, and can be integrated into any programming language supporting COM.

The following sample source code is written using Visual Basic and illustrates basic handling of the Event interface.

The client is written in Visual Basic 6.0, and it's a Standard EXE project. Once the form has been created, select the References command from the Project menu and check CCASCom 1.0 Type Library. Add controls to the form for the user interface. Double-click on the form to get the code window, and add the following above Form_Load:

Private WithEvents CCASCom As CCASClient

The WithEvents keyword instructs the Visual Basic IDE to read the type library for the outgoing interfaces of the object. This will allow Visual Basic to determine what events the application should handle. Now if you select CCASCom from the (left-hand) Object list, you will find that the events of the object are listed in the (right-hand) Procedure list. Map the click event for all command buttons, and map all events of the object. The form source code is as follows.

```
Private    WithEvents    CCASCom    As
CCASClient

Attribute CCASCom.VB_VarHelpID =
-1

Private Sub

   bDisconnect_Click()

   CCASCom.Uninitialize

   lstEvents.Clear

   lstEvents.AddItem

   "Disconnected"

   bDisconnect.Enabled =

   False bInitialize.Enabled

   = True

   txtServerName.Enabled =

   True txtServerPort.Enabled

   = True

 End Sub
```

```
Private Sub
   bInitialize_Click()
   Dim szError As
   String
   Dim lError As Long

  CCASCom.Initialize

  If txtServerName =
     "" Then
     txtServerName.Set
     Focus
  ElseIf txtServerPort =
     "" Then
     txtServerPort.SetFo
     cus
  Else
     lstEvent
     s.Clear
     lstEvents.AddItem "Attempting to Connect to " +
txtServerName + " using port " + txtServerPort + "..."
     On Error GoTo Failed
     CCASCom.Connect txtServerName, txtServerPort
     lstEvents.AddItem "Connected to Server"
     GoTo FuncEnd
  End If
Failed:
     lstEvents.AddItem "Failed to Connect to
     Server"
     CCASCom.GetLastErrorInfo lError, szError
     lstEvents.AddItem "Error = " + szError

End Sub

Private Sub
  CCASCom_OnDisconnect()
  CCASCom.Uninitialize
  lstEvents.Clear
```

```
   lstEvents.AddItem
   "Disconnected"
   bDisconnect.Enabled =
   False bInitialize.Enabled
   = True
   txtServerName.Enabled =
   True txtServerPort.Enabled
   = True
End Sub

Private Sub CCASCom_OnEvent(ByVal pEvent As CCASCOMCLIEN-
TLib.ICCASEvent)

   Dim n As Long

   Dim nValue As Long

   Dim strData As String


    Dim strText As String

    Dim strType As String


    pEvent.GetType n

    ' Check for data end event

    If n = 35 Then
      bInitialize.Enabled = False
      bDisconnect.Enabled = True
      bDisconnect.SetFocus
      txtServerName.Enabled = False
      txtServerPort.Enabled = False
    End If

    lstEvents.AddItem "[OnEvent]"

    If n = 1 Then
      lstEvents.AddItem "Event Type:  " + Str(n) + " -
      BroadcastLogon"
      strText = "RECID"
      pEvent.GetLongValue strText, nValue lstEvents.AddItem
      "                    RecID = " + Str(nValue)
      strText = "DN"
      pEvent.GetStringValue strText, strData
      lstEvents.AddItem "   DN = " + strData
      strText = "NAME"
      pEvent.GetStringValue strText, strData
      lstEvents.AddItem "   Name = " + strData
```

```
    strText = "LogonID"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   LogonID = " + strData
    strText = "ServiceGroupList" pEvent.GetStringValue
    strText, strData
    lstEvents.AddItem "   SGList = " + strData
    strText = "VOICEREADY"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   VoiceReady = " + Str(nValue)
    strText = "MEDIAREADY"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   MediaReady = " + Str(nValue)
   strText = "EMAILREADY"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   EmailReady = " + Str(nValue)
   strText = "MACHINENAME"
   pEvent.GetStringValue strText, strData
   lstEvents.AddItem "   MachineName = " + strData
ElseIf n = 2 Then
   lstEvents.AddItem "Event Type:  " + Str(n) + " -
   BroadcastLogoff"
   strText = "RECID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   RecID = " +
   Str(nValue)
ElseIf n = 3 Then
   lstEvents.AddItem "Event Type:  " + Str(n) + " -
   BroadcastReady" strText = "RECID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   RecID = " +
   Str(nValue)
ElseIf n = 4 Then
   lstEvents.AddItem "Event Type:  " + Str(n) + " -
   BroadcastNotReady"
   strText = "RECID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   RecID = " + Str(nValue)
   strText = "REASONID"
```

```
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     ReasonID = " + Str(nValue)
    strText = "REASONSTRING"
    pEvent.GetStringValue strText, strValue
    lstEvents.AddItem "     ReasonString = " + strValue

  ElseIf n = 5 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastEmail- Ready"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue lstEvents.AddItem "
                        RecID = " + Str(nValue)

  ElseIf n = 6 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastEmail- NotReady"
    strText = "RECID" pEvent.GetLongValue strText,
    nValue
    lstEvents.AddItem "     RecID = " + Str(nValue)
    strText = "REASONID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     ReasonID = " + Str(nValue)
    strText = "REASONSTRING"
    pEvent.GetStringValue strText, strValue
    lstEvents.AddItem "     ReasonString = " + strValue

  ElseIf n = 7 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
    BroadcastOriginated"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue lstEvents.AddItem "
                        RecID = " + Str(nValue)
    strText = "CALLID"
    pEvent.GetLongValue strText, nValue lstEvents.AddItem "
                        CallID = " + Str(nValue)
    strText = "SERVICEGROUPID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   SG ID = " + Str(nValue)
```

```
    strText = "CALLTYPE"
    pEvent.GetLongValue strText, nValue
    GetCallTypeString nValue, strType
    lstEvents.AddItem "   CallType = " + Str(nValue) + " - "
    + strType
ElseIf n = 8 Then
  lstEvents.AddItem "Event Type:  " + Str(n) + " -
   BroadcastDelivered"
  strText = "RECID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     RecID = " + Str(nValue)
  strText = "CALLID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     CallID = " + Str(nValue)
  strText = "OPPOSITEPARTYNUMBER"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "     OppositeParty = " + strData
  strText = "PRIVATEDATA"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "     PrivateData = " + strData
  strText = "CAUSE"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "     Cause = " + strData
  strText = "CALLEDNUMBER"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "     CalledNumber = " + strData

ElseIf n = 9 Then
  lstEvents.AddItem "Event Type:  " + Str(n) + " -
   BroadcastEstablished"
  strText = "RECID" pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     RecID = " + Str(nValue)
  strText = "CALLID" pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "     CallID = " + Str(nValue)
  strText = "TRUNKID"
  pEvent.GetStringValue strText, strData
```

```
    lstEvents.AddItem "     TrunkID = " + strData
    strText = "CALLSUBSTATE"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     CallSubstate = " + strData
    strText = "OPPOSITEPARTYNUMBER"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     OppositeParty = " + strData
    strText = "CAUSE"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     Cause = " + strData
    strText = "MEDIASERVERSESSIONURI"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     MediaServerSessionUri = " +
     strData

ElseIf n = 10 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
    BroadcastHeld"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   RecID = " + Str(nValue)
    strText = "CALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   CallID = " + Str(nValue)
    strText = "ONHOLD"
    pEvent.GetLogValue strText, nValue
    lstEvents.AddItem "     OnHold = " + Str(nValue)

ElseIf n = 11 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
    BroadcastRetrieved"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   RecID = " + Str(nValue)
    strText = "CALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   CallID = " + Str(nValue)
```

```
  ElseIf n = 12 Then

    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastTransferred"

    strText = "RECID" pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   RecID = " + Str(nValue)

    strText = "HELDCALLID" pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   HeldCallID = " + Str(nValue)

    strText = "ACTIVECALLID" pEvent.GetLongValue strText,
    nValue

    lstEvents.AddItem "     ActiveCallID = " + Str(nValue)

    strText = "NEWCALLID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "     NewCallID = " + Str(nValue)
    strText = "CLERICALFLAG"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "     Clerical = " + Str(nValue)

    strText = "TRANSFERRINGAGENT"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "     Transferring Agent = " +
    Str(nValue)

    strText = "OPPOSITEPARTYNUMBER"

    pEvent.GetStringValue strText, strData

    lstEvents.AddItem "     OppositeParty = " + strData

    strText = "PRIVATEDATA"

    pEvent.GetStringValue strText, strData

    lstEvents.AddItem "     PrivateData = " + strData

  ElseIf n = 13 Then

    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastConferenced"

    strText = "RECID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   RecID = " + Str(nValue)

    strText = "HELDCALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   HeldCallID = " + Str(nValue)

    strText = "ACTIVECALLID" pEvent.GetLongValue strText,
```

```
        nValue
        lstEvents.AddItem "   ActiveCallID = " + Str(nValue)
        strText = "NEWCALLID"
        pEvent.GetLongValue strText, nValue lstEvents.AddItem "
                              NewCallID = " + Str(nValue)
       strText = "OPPOSITEPARTYNUMBER"
       pEvent.GetStringValue strText, strData
       lstEvents.AddItem "     OppositeParty = " + strData

     ElseIf n = 14 Then
       lstEvents.AddItem "Event Type:  " + Str(n) + " -
            BroadcastConnectionCleared"
       strText = "RECID"
       pEvent.GetLongValue strText, nValue
       lstEvents.AddItem "    RecID = " + Str(nValue)
       strText = "CALLID"
       pEvent.GetLongValue strText, nValue
       lstEvents.AddItem "    CallID = " + Str(nValue)
       strText = "CLERICALFLAG"
       pEvent.GetLongValue strText, nValue
       lstEvents.AddItem "    Clerical = " + Str(nValue)
       strText = "DIVERTEDFLAG"
       pEvent.GetLongValue strText, nValue
       lstEvents.AddItem "    Diverted = " + Str(nValue)
       strText = "DIVERTDESTINATION"
       pEvent.GetStringValue strText, strData
       lstEvents.AddItem "     DivertDest = " + strData

    ElseIf n = 15 Then
       lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastClericalEnded"
       strText = "RECID"
       pEvent.GetLongValue strText, nValue
       lstEvents.AddItem "   RecID = " + Str(nValue)
       strText = "CALLID"
       pEvent.GetLongValue strText, nValue
```

```
    lstEvents.AddItem "    CallID = " + Str(nValue)

  ElseIf n = 16 Then

    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastCallRejected"

    strText = "RECID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   RecID = " + Str(nValue)

    strText = "CALLID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   CallID = " + Str(nValue)

    strText = "CAUSE"

    pEvent.GetStringValue strText, strData

    lstEvents.AddItem "   Cause = " + strData


  ElseIf n = 17 Then

    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastCallbackAccept"

    strText = "RECID" pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   RecID = " + Str(nValue)

    strText = "ORIGCALLID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   OrigCallID = " + Str(nValue)

    strText = "CALLID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   CallID = " + Str(nValue)

    strText = "CALLTYPE"

    pEvent.GetLongValue strText, nValue

    GetCallTypeString nValue, strType

    lstEvents.AddItem "   CallType = " + Str(nValue) + " - " +
    strType


   ElseIf n = 18 Then

     lstEvents.AddItem "Event Type:  " + Str(n) + " -
 BroadcastCallbackReject"

     strText = "RECID"

     pEvent.GetLongValue strText, nValue

     lstEvents.AddItem "   RecID = " + Str(nValue)
```

```
    strText = "CALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   CallID = " + Str(nValue)
    strText = "CALLTYPE"
    pEvent.GetLongValue strText, nValue
    GetCallTypeString nValue, strType
    lstEvents.AddItem "   CallType = " + Str(nValue) + " - "
    + strType
    strText = "CAUSE"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   Cause = " + strData

  ElseIf n = 19 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastCallbackStatus"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   RecID = " + Str(nValue)
    strText = "CALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   CallID = " + Str(nValue)
    strText = "CALLTYPE"
    pEvent.GetLongValue strText, nValue
    GetCallTypeString nValue, strType
    lstEvents.AddItem "   CallType = " + Str(nValue) + " - "
    + strType
    strText = "CALLBACKSTATUS"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   Status = " + Str(nValue)
    strText = "CAUSE"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   Cause = " + strData

  ElseIf n = 20 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
    BroadcastEmailReject”
```

```
   strText = "RECID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem " RecID = " + Str(nValue)
   strText = "MESSAGEID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem " MsgID = " + Str(nValue)
   strText = "SERVICEGROUPID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem " SG ID = " + Str(nValue)
   strText = "CAUSE"
   pEvent.GetStringValue strText, strData
   lstEvents.AddItem " Cause = " + strData

ElseIf n = 21 Then
   lstEvents.AddItem "Event Type: " + Str(n) + " -
   BroadcastEmailDelete"
   strText = "RECID"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem " RecID = " + Str(nValue)
   strText = "MESSAGEID"
   lstEvents.AddItem " MsgID = " + Str(nValue)
   pEvent.GetLongValue strText, nValue
   strText = "CLERICALFLAG"
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem " Clerical = " + Str(nValue)

 ElseIf n = 22 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
       BroadcastEmailReply"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue lstEvents.AddItem "
       RecID = " + Str(nValue)
    strText = "MESSAGEID"
    pEvent.GetLongValue strText, nValue lstEvents.AddItem "
       MsgID = " + Str(nValue)
    strText = "SERVICEGROUPID"
```

```
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   SG ID = " + Str(nValue)

  strText = "SUBJECT"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem " Subject = " + strData
  strText = "FORWARDFLAG"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     Forwarded = " + Str(nValue)
  strText = "RECIPIENTLIST"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem " Recipients = " + strData
  strText = "EMAILTEXT"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem " Text = " + strData
  strText = "CLERICALFLAG"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem " Clerical = " + Str(nValue)

ElseIf n = 23 Then
  lstEvents.AddItem "Event Type:  " + Str(n) + " -
     BroadcastEmailForward"
  strText = "RECID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "   RecID = " + Str(nValue)
  strText = "MESSAGEID" pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "   MsgID = " + Str(nValue)
  strText = "SERVICEGROUPID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "   SG ID = " + Str(nValue)

ElseIf n = 24 Then
  lstEvents.AddItem "Event Type:  " + Str(n) + " -
      BroadcastCallInformation"

  strText = "RECID" pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     RecID = " + Str(nValue)
  strText = "CALLID"
  pEvent.GetLongValue strText, nValue
```

```
lstEvents.AddItem "   CallID = " + Str(nValue)
strText = "SERVICEGROUPID"
pEvent.GetLongValue strText, nValue
lstEvents.AddItem "   SG ID = " + Str(nValue)
strText = "CALLINGPARTYNUMBER"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   CallingParty = " + strData
strText = "CALLEDNUMBER" pEvent.GetStringValue strText,
strData lstEvents.AddItem "    Called Number = " + strData
strText = "IVRLABEL1"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 1= " + strData
strText = "IVRDATA1"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 1= " + strData
strText = "IVRLABEL2"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 2= " + strData
strText = "IVRDATA2"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 2= " + strData
strText = "IVRLABEL3"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 3= " + strData
strText = "IVRDATA3"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 3= " + strData
strText = "IVRLABEL4"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 4= " + strData
strText = "IVRDATA4"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 4= " + strData
strText = "IVRLABEL5"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 5= " + strData
strText = "IVRDATA5"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 5= " + strData
strText = "IVRLABEL6"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 6= " + strData
strText = "IVRDATA6"
```

```
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 6= " + strData
  strText = "IVRLABEL7"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 7= " + strData
  strText = "IVRDATA7"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 7= " + strData
  strText = "IVRLABEL8"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 8= " + strData
  strText = "IVRDATA8"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 8= " + strData
  strText = "IVRLABEL9"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 9= " + strData
  strText = "IVRDATA9"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 9= " + strData
  strText = "IVRLABEL10"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 10= " + strData
  strText = "IVRDATA10"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 10= " + strData
  strText = "TIMEINQUEUE" pEvent.GetLongValue strText,
     nValue
  lstEvents.AddItem "   Time in Queue = " + Str(nValue)
  strText = "CALLTYPE" pEvent.GetLongValue strText, nValue
  GetCallTypeString nValue, strType
  lstEvents.AddItem "   CallType = " + Str(nValue) + " - " +
     strType

ElseIf n = 25 Then
  lstEvents.AddItem "Event Type:  " + Str(n) + " -
     BroadcastCallback" strText = "RECID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "   RecID = " + Str(nValue)
  strText = "SERVICEGROUPID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "   SG ID = " + Str(nValue)
  strText = "CALLINGPARTYNUMBER"
  pEvent.GetStringValue strText, strData
```

```
    lstEvents.AddItem "   CallingParty = " + strData
    strText = "CALLEDNUMBER"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   Called Number = " + strData
    strText = "IVRLABEL1"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Label 1= " + strData
    strText = "IVRDATA1"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Data 1= " + strData
    strText = "IVRLABEL2"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Label 2= " + strData
    strText = "IVRDATA2"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Data 2= " + strData
    strText = "IVRLABEL3"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Label 3= " + strData
    strText = "IVRDATA3"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "   IVR Data 3= " + strData
    strText = "CALLTYPE"
    pEvent.GetLongValue strText, nValue

    GetCallTypeString nValue, strType
    lstEvents.AddItem "     CallType = " + Str(nValue) + " - "
        + strType


ElseIf n = 29 Then

   lstEvents.AddItem "Event Type:  " + Str(n) + " -
       BroadcastCampaignAccept"

    strText = "RECID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   RecID = " + Str(nValue)

    strText = "CAMPAIGNID"

    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "   CampaignID = " + Str(nValue)

    strText = "CUSTOMERID"

    pEvent.GetLongValue strText, nValue

    lstEvents.AddItem "   CustomerID = " + Str(nValue)

    strText = "CALLID"
```

```
   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   CallID = " + Str(nValue)


ElseIf n = 30 Then

  lstEvents.AddItem "Event Type:  " + Str(n) + " -
        BroadcastCampaignReject"

   strText = "RECID"

   pEvent.GetLongValue strText, nValue

   lstEvents.AddItem "   RecID = " + Str(nValue)

   strText = "CAMPAIGNID"

   pEvent.GetLongValue strText, nValue
   lstEvents.AddItem "   CampaignID = " + Str(nValue)

   strText = "CUSTOMERID"

   pEvent.GetLongValue strText, nValue

   lstEvents.AddItem "   CustomerID = " + Str(nValue)

   strText = "CAUSE"
   pEvent.GetStringValue strText, strData
   lstEvents.AddItem "   Cause = " + strData


ElseIf n = 31 Then

    lstEvents.AddItem "Event Type:  " + Str(n) + " -
         BroadcastCampaignStatus"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     RecID = " + Str(nValue)
    strText = "CAMPAIGNID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     CampaignID = " + Str(nValue)
    strText = "CUSTOMERID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     CustomerID = " + Str(nValue)
    strText = "CAUSE"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     Cause = " + strData

 ElseIf n = 32 Then

   lstEvents.AddItem "Event Type:  " + Str(n) + " -
         BroadcastCampaignInfo"

    strText = "RECID"
```

```
pEvent.GetLongValue strText, nValue lstEvents.AddItem "
   RecID = " + Str(nValue)
strText = "CAMPAIGNID"
pEvent.GetLongValue strText, nValue
lstEvents.AddItem "   CampaignID = " + Str(nValue)
strText = "CUSTOMERID"
pEvent.GetLongValue strText, nValue lstEvents.AddItem "
   CustomerID = " + Str(nValue)
strText = "CUSTOMERNAME"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "    Customer Name = " + strData
strText = "CUSTOMERNUMBER"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   Customer Number = " + strData
strText = "COMMENT"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   Comment = " + strData
strText = "IVRLABEL1"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 1= " + strData
strText = "IVRDATA1"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 1= " + strData
strText = "IVRLABEL2"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 2= " + strData
strText = "IVRDATA2"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 2= " + strData
strText = "IVRLABEL3"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 3= " + strData
strText = "IVRDATA3"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 3= " + strData
strText = "IVRLABEL4"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Label 4= " + strData
strText = "IVRDATA4"
pEvent.GetStringValue strText, strData
lstEvents.AddItem "   IVR Data 4= " + strData
```

```
  strText = "IVRLABEL5"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 5= " + strData
  strText = "IVRDATA5"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 5= " + strData
  strText = "IVRLABEL6"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 6= " + strData
  strText = "IVRDATA6"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 6= " + strData
  strText = "IVRLABEL7"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 7= " + strData
  strText = "IVRDATA7"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 7= " + strData
  strText = "IVRLABEL8"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 8= " + strData
  strText = "IVRDATA8"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 8= " + strData
  strText = "IVRLABEL9"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 9= " + strData
  strText = "IVRDATA9"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 9= " + strData
  strText = "IVRLABEL10"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Label 10= " + strData
  strText = "IVRDATA10"
  pEvent.GetStringValue strText, strData
  lstEvents.AddItem "   IVR Data 10= " + strData


ElseIf n = 33 Then

  lstEvents.AddItem "Event Type:  " + Str(n) + " -
      BroadcastServiceGroupAdded"
  strText = "RECID"
  pEvent.GetLongValue strText, nValue
  lstEvents.AddItem "     SG ID = " + Str(nValue)
```

```
     strText = "NAME"
     pEvent.GetStringValue strText, strData
     lstEvents.AddItem "   SG Name = " + strData
     strText = "SERVICEGROUPPURPOSE"
     pEvent.GetLongValue strText, nValue
     If nValue = 0 Then strData = " (General)"
     ElseIf nValue = 1 Then strData = " (Campaign)"
     ElseIf nValue = 2 Then strData = " (Email)"
     End If
     lstEvents.AddItem "   SG Type = " + Str(nValue) + strData

   ElseIf n = 34 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastAgentServiceGroupsChanged"
     strText = "RECID" pEvent.GetLongValue strText, nValue
     lstEvents.AddItem "   RecID = " + Str(nValue)
     strText = "SERVICEGROUPLIST"
     pEvent.GetStringValue strText, strData
     lstEvents.AddItem "   SG List = " + strData

   ElseIf n = 35 Then
    lstEvents.AddItem "Event Type:  " + Str(n) + " -
BroadcastDataComplete"

   ElseIf n = 36 Then
     lstEvents.AddItem "Event Type: " + Str(n) + " =
         BroadcastDisconnect"

   ElseIf n = 37 Then
    lstEvents.AddItem "Event Type: " + Str(n) + " =
         BroadcastXferCallInformation"
    strText = "RECID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     RecID = " +
         Str(nValue)
    strText = "HELDCALLID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     HeldCallID = " + Str(nValue)
    strText = "ACTIVECALLID"
    pEvent.GetLongValue strText, nValue
```

```
    lstEvents.AddItem "     ActiveCallID = " + Str(nValue)
    strText = "SERVICEGROUPID"
    pEvent.GetLongValue strText, nValue
    lstEvents.AddItem "     SG ID = " + Str(nValue)
    strText = "CALLINGPARTYNUMBER"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     CallingParty = " + strData
    strText = "CALLEDNUMBER"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "     Called Number = " + strData
    strText = "IVRLABEL1"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Label 1= " + strData
    strText = "IVRDATA1"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Data 1= " + strData
    strText = "IVRLABEL2"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Label 2= " + strData
    strText = "IVRDATA2"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Data 2= " + strData
    strText = "IVRLABEL3"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Label 3= " + strData
    strText = "IVRDATA3"
    pEvent.GetStringValue strText, strData
    lstEvents.AddItem "    IVR Data 3= " + strData

  End If
  If cmdScrollToBottom.Value = 1 Then lstEvents.ListIndex =
    lstEvents.ListCount - 1
  End If
End Sub
Private Sub cmdClearEvents_Click()
  lstEvents.Clear
End Sub
Private Sub Form_Load() On Error Resume Next
Set CCASCom = New CCASClient bDisconnect.Enabled = False
```

```
    bInitialize.SetFocus
End Sub

Private Sub GetCallTypeString(ByVal lCallType As Long, strCallType
As String)

    If lCallType = 0 Then strCallType = "WebTextChat"
    ElseIf lCallType = 1 Then strCallType = "WebEmail"
    ElseIf lCallType = 2 Then strCallType = "WebVoIP"
    ElseIf lCallType = 3 Then strCallType = "WebCallback"
    ElseIf lCallType = 4 Then strCallType = "Voice Call"
    ElseIf lCallType = 5 Then strCallType = "Email"
    ElseIf lCallType = 6 Then strCallType = "Campaign"
    ElseIf lCallType = 7 Then strCallType = "OtherPBX Call"
    ElseIf lCallType = 8 Then strCallType = "NetMeeting Call"
    Else strCallType = "Unknown"
    End If
End Sub.
```

EXCEPTION HANDLING

Exceptions generated in either the client application or in the COM object itself will be handled within the COM object. If the client application needs to handle exceptions generated in the client code, a Try/Catch block should be added to the OnEvent() and OnDisconnect() event handlers inside the client application.