CTI OS ActiveX Controls

The CTI OS Developer Toolkit includes a set of ActiveX controls to enable rapid application development. ActiveX controls are typically UI components (there are also ActiveX controls that are invisible at run time) that enable easy drag-and-drop creation of custom CTI applications in a variety of container applications. Container applications include: Microsoft Visual Basic 6.0, Microsoft Internet Explorer, Microsoft Visual C++ 7.1(1), Borland Delphi, Sybase Powerbuilder and other applications supporting the OC96 ActiveX standard.

The CTI OS Agent Desktop and CTI OS Supervisor Desktop for Unified CCE applications are both Visual Basic applications based on the CTI OS ActiveX controls.

For more information, see *CTI OS Agent Desktop User Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted* and the *CTI OS Supervisor Desktop User Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*.

The following table lists the ActiveX controls included with CTI OS. As seen in the table, you can group CTI OS Controls into Agent Related Controls, Call Related Buttons, Statistics Controls, and Supervisor Controls.

Table 1: CTI OS ActiveX Controls

Control	Description
Agent Related Controls	
AgentGreetingCtl	Provides a UI for turning the Agent Greeting feature on or off.
AgentStateCtl	Provides UI to for login, logout, ready, not ready and wrapup requests, also enables the use to specify reason codes for logout and Not_Ready (if supported and configured).
ChatCtl	Provides UI to send text messages to a supervisor or (if allowed) to other agents.
EmergencyAssistCtl	Provides UI to place Emergency and Supervisor Assist calls. These calls allow agents to conveniently contact a supervisor if they need help. You must configure the corresponding Unified ICM scripts for this control to work.

Control	Description
Record Greeting Ctl	Provides a UI for recording and managing agent greeting messages by placing a Make Call request to a hard coded “RecordAgentGreeting” Dialed Number and setting the Placement type (CPT_RECORD_AGENT_GREETING) to 7.
Call Related Controls	
AlternateCtl	Provides UI for alternate requests. If an agent has Call A active and Call B on hold, alternate puts call A on hold and makes Call B active. Useful during consult calls.
AnswerCtl	Provides UI to answer a call. Only a call with state “LCS_Alerting” (Ringing) can be answered.
BadLineCtl	Provides a UI to report a Bad Line, such as bad voice quality or equipment problems.
CallAppearanceCtl	A grid control displaying call information, including call status and context data.
ConferenceCtl	Provides UI to place a conference call in single step or consultative mode.
HoldCtl	Provides UI to put calls on hold and retrieve them.
MakeCallCtl	Provides UI to enter a telephone number and place a make call request.
ReconnectCtl	Provides a UI for reconnect requests. If an agent has Call A active and Call B on hold, reconnect hangs up call A and makes Call B active. Useful during consult calls to return to the original call.
StatusBarCtl	Visually displays information about the logged on agent (id, instrument, extension, current state).
RecordCtl	Provides UI for Call Recording requests (start/stop recording), the requests are forwarded to CTI Server, so they care handled by a configured call recording service.
TransferCtl	Provides UI to transfer a call in single step or consultative mode.
Statistics Controls	
AgentStatisticsCtl	A grid control displaying real-time agent statistics. Columns are configurable at CTI OS server (for more information, see *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*).

Control	Description
SkillgroupStatisticsCtl	A grid control displaying real time skill group statistics. Columns are configurable at CTI OS server (for more information, see *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*).
Supervisor Controls	
AgentSelectCtl	Supervisor specific; displays all agent team members of a supervisor (configured by Unified ICM) , including agent name, agentid, agentstate, timeinstate and skillgroups.
SupervisorOnlyCtl	Provides UI for Supervisor functions including Barge-In, Intercept, logout monitored agent and make monitored agent ready.
SilentMonitorCtl.dll	Standalone control that can create a monitoring application that connects to CTI OS, but does not need to login as a supervisor.

- Property Pages, page 3
- Button Controls and Grid Controls, page 3
- CTI OS ActiveX Control Descriptions, page 8
- The Silent Monitor StandAlone ActiveX Control, page 47

Property Pages

While most settings in CTI OS are downloaded from CTI OS server to the client, ActiveX controls additionally offer property pages. The activation of the property pages is container dependent (for example, in Visual Basic, you can “right-click” on an ActiveX control and select Properties from the pop-up menu). In CTI OS the most common properties selectable via property pages are ButtonType (for example, The Holdctl can be a hold or retrieve button), fonts and colors.

Button Controls and Grid Controls

Most of the CTI OS ActiveX controls are either Button Type Controls or Grid Type Control, with the following exceptions:

- Statusbarcontrol
- ChatCtl
- Utility controls (such as CtiCommonDlgs and SubClassFormCtl)

Note The Utility controls, such as CtiCommonDlgs and SubClassFormCtl (used by the CTI OS Agent and Supervisor desktops), are for *Internal Use Only*.

As such they share common principles.

The following table describes button enablement scenarios only for call control, agent state control and supervisor assist in Unified CCE.

Note The *video control button* (under the tools group) is not included in standard Unified CCE desktops. This button is related to controls exercised on the supervisor desktop.

Table 2: Button Enablement in a Standard CTI OS Desktop for Unified CCE

Scenarios	Buttons enabled and Description
Agent is not logged in to a desktop	Only the **Log-in** button is enabled.
Agent in the Not ready state	The agent is in the **Not Ready** state. The following buttons are enabled: Ready, Supervisor Assist, Emergency Assist, Statistics and Chat Control, and Make Call Control. **Note** The Make Call Control button allows you to dial out only in the Not Ready (NR) state.
Agent in the Ready state	The agent is in the **Ready** state. The following buttons are enabled: Not Ready, Supervisor Assist, Emergency Assist, Statistics and Chat Control, and Make call Control. **Note** The Make Call Control button allows you to dial out only in the Not Ready state.
Agent gets an incoming call that is alerted on the agent desktop	The agent is in the **Reserved** state. The following buttons are enabled: Statistics, Chat Control, and Bad Line Ctrl.
Agent answers the call	The agents is in the **Talking** state. The following buttons are enabled: Hold, Release, Supervisor Assist, Emergency Assist, Conference, transfer, Statistics, Chat ctl, and Badline control. **Note** The Ready, Not Ready, and wrap-up buttons are enabled based on the agent desktop settings configured in the Configuration Manager.
Agent puts an answered call on hold	The agent is in the **On hold** state. The following buttons are enabled: Retrieve, Release, Supervisor Assist, Emergency Assist, Statistics, Chat ctl, and Badline control.

Scenarios	Buttons enabled and Description
Agent retrieves the call on hold	The agents is in the **Talking** state. The following buttons are enabled: Hold, Release, Supervisor Assist, Emergency Assist, Conference, transfer, Statistics, Chat ctl and Badline control.
Agent releases the call	The agent continues to be in the same state he was in before talking the call. **Note** The agent is in the Wrap-up state, provided Wrap-up is configured. If Wrap-up was not configured, then the **Ready** and **Not Ready** buttons are enabled.
Agent initiates a conference	The agent is in the **Talking** state. The following buttons are enabled: Statistics, Chat ctl, Bad Line Ctrl, and Reconnect.
Consult conference call is answered	*Conference Initiator Desktop* The agent is in the **Talking** state. The following buttons are enabled: Statistics, Chat ctl, BadLineCtrl, Alternate, Reconnect, and Conference complete. **Note** The Ready, NR and Wrap-up buttons are enabled based on the agent desktop setting. *Conference Receiver Desktop* The agent is in the **Talking** state. The following buttons are enabled: Hold, Release, Supervisor Assist, Emergency Assist, Conference, transfer, Statistics, Chat ctl, and Badline control.
Complete conference is done for the conference initiator	The agent continues to be in the same state he was in, before talking the call. **Note** The agent is in the Wrap-up state, provided Wrap-up is configured. If Wrap-up was not configured, then the **Ready** and **Not Ready** buttons are enabled.
Agent initiates a consult transfer	The agent is in the **Talking** state. The following buttons enabled: Statistics, Chat ctl, BadLine Ctrl, and Reconnect.

Scenarios	Buttons enabled and Description
Consult transfer call is answered	*Consult transfer Initiator Desktop* The agent is in the **Talking** state. The following buttons are enabled: Statistics, Chat ctl, Bad Line Ctrl, Alternate, Reconnect, Conference complete. **Note** The Ready, NR, and Wrap-up buttons are enabled based on the agent desktop setting. *Consult transfer Receiver Desktop* The agents is in the **Talking** state. The following buttons are enabled: Hold, Release, Supervisor Assist, Emergency Assist, Conference, transfer, Statistics, Chat ctl, and Badline control. **Note** The Ready, Not Ready, and wrap-up buttons are enabled based on the agent desktop settings configured in the Configuration Manager.
Complete conference is done for the Transfer Initiator	The agent continues to be in the same state he was in, before talking the call. **Note** The agent is in the Wrap-up state, provided Wrap-up is configured. If Wrap-up was not configured, then the **Ready** and **Not Ready** buttons are enabled.
Agent does a Single Step Transfer	The agent continues to be in the same state he was in, before talking the call. **Note** The agent is in the Wrap-up state, provided Wrap-up is configured. If Wrap-up was not configured, then the **Ready** and **Not Ready** buttons are enabled. After Initiating a Consult Transfer or a Consult Conference, there are two calls on the agent desktop of the initiator of Transfer and Conference until the Transfer/Conference is complete.
The held call is selected for Consult Conference when the call rings on the conference agent desktop	The following buttons are enabled: Statists, Chat ctl, BadLine Ctrl, and Reconnect. **Note** The Ready, NR, and Wrap-up buttons are enabled based on the agent desktop setting.

Scenarios	Buttons enabled and Description
The held call is selected for Consult Transfer when the call rings on the conference agent desktop	The following buttons are enabled: Statistics, Chat ctl, BadLine Ctrl, and Reconnect. **Note** The Ready, NR, and Wrap-up buttons are enabled based on the agent desktop setting. The button enablement on desktops based on the agent desk settings depends on: • Work Mode incoming • Work Mode outgoing The button enablement described are common, the only difference being they are enabled on an incoming call or an outgoing call based on the settings.
Agent desk setting configuration is set to **Not Allowed**	The agent is not allowed to go to the Wrap-up state. No state transition buttons are enabled as long as the call is on the agent desktop. Buttons Enabled - Default behavior.
Agent desk setting configuration is set to **Optional**	The Ready, NR, and Wrap-up buttons are enabled after a call is answered and the agent is in the **Talking** state.
Agent desk setting configuration is set to **Required/Required with data**	No state change buttons are enabled when the agent is talking on a call. After the call, the agent state changes to **Wrap-up**. In the Wrap-up dialog box, if you select “Apply”, the Ready and NR buttons are enabled and the agent state changes based on the selection (button click) done by the agent. **Note** Supervisor assist and Emergency assist are not present on a supervisor desktop. The rest of the button enablement are only applicable for different scenarios described above.

These are basic call scenarios and only to be used for reference. Customized desktops can have different enablement behaviors.

Button Controls

Button Controls include the AgentStateCtl, AlternateCtl, AnswerCtl, BadLineCtl, ConferenceCtl, EmergencyAssistCtl, HoldCtl, MakeCallCtl, ReconnectCtl, SupervisorOnlyCtl, RecordCtl, and TransferCtl. They provide an UI to perform a certain CTI task (like logging in or answering a call). All of the Button Controls are based on the CTI OS ButtonCtl and share the same characteristics. All CTI OS buttons enable and disable themselves automatically based on the current state of the system. For example, if an agent is not logged in, the only button available to click is the Login Button (see AgentStateCtl), or if a call was not answered and is selected in the CallAppearanceCtl, the Answer Button is enabled (see AnswerCtl and CallAppearanceCtl). All buttons can be configured via their property pages to show custom text captions, custom icons and custom tooltip captions.

Grid Controls

Grid controls include the AgentSelectCtl, CallAppearanceCtl, AgentStatisticsCtl and SkillGroupStatisticsCtl. The Grid Controls display data, select calls (see CallAppearanceCtl) or Agents (AgentSelectCtl), and in some cases allow you to enter data (for example, Callvariables in the CallAppearanceCtl). You can use CTI OS to configure the following grid properties. See the *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*.

- Columns to display
- Column header
- Column width
- Column alignment

Supervisor Status Bar

The Supervisor Softphone has a status bar that appears at the bottom of the window. The supervisor status bar information is configurable at design time using the property pages. You can also set it programmatically at run time.

CTI OS ActiveX Control Descriptions

This section describes the CTI OS ActiveX softphone controls listed in Table 1: CTI OS ActiveX Controls, on page 1.

AgentGreetingCtl

The Agent Greeting control enables the Agent Greeting feature to be turned on or off by toggling the Agent Greeting button to the on or off state. Agent Greeting is automatically in the on state at login and the icon appears like this

Click the button to turn Agent Greeting off and the icon changes to

Click the button again to turn Agent Greeting on.

RecordGreetingCtl

The Record Agent Greeting button is used to record and manage Agent Greeting messages.

Recording an Agent Greeting is very similar to recording a personal message for your voice mail. To record a greeting, you must be logged in to your desktop software and in the Not Ready state. After you click the Agent Greeting Record button you hear a brief ring tone, after which you receive voice instructions for recording a greeting. Options include selecting a greeting type (if your contact center uses more than one greeting per agent), recording, playing back, and confirming whether to use the new greeting. There is also an option for listening to your existing greetings.

The RecordGreeting control is basically just a MakeCall request to the appropriate DN. It places a Make Call request to a hard coded "RecordAgentGreeting" Dialed Number and sets the Placement type (CPT_RECORD_AGENT_GREETING) to 7.

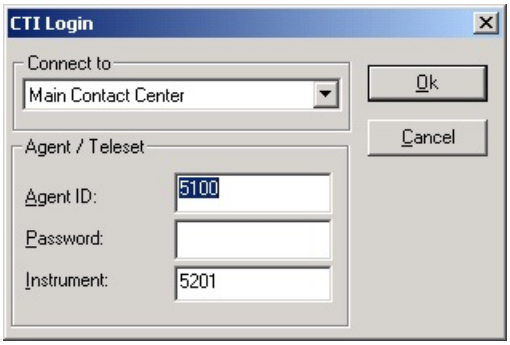
AgentStateCtl

The agentstate control is based on the CTI OS button control and can be one of several button types. To select the button type, bring up the property page (container dependent, for example right click in VB) and select the desired agentstate functionality from the following:

- **Login Button**. Click the login button to allow the agent to select a connection profile (see the *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*), agent ID and instrument or other switch specific fields.

Figure 1: Login Button

Figure 2: Login Dialog



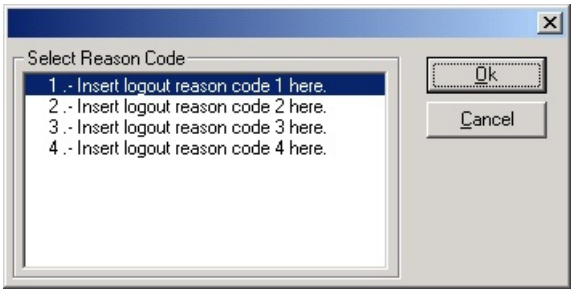
You can configure the fields displayed. The dialog box shows a login dialog box configured for Unified CCE . An agent logging in can select a connection profile for the **Connect To:** drop down box, enter an agent ID, password and instrument and click **OK** to send a Login request.

- **Logout Button**. Click the logout button to log out the currently logged in agent. For some switches, including Unified CCE, the agent must be in the not ready state to enable this button. If Reason Codes

are supported on the switch and configured on Unified ICM , a reason code dialog box pops up as shown below.

Figure 3: Logout Button

Figure 4: Reason Code Dialog for Logout

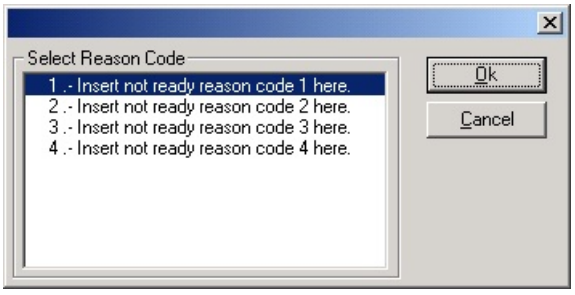


Use this dialog box to select a reason code to send with the logout request. You can configure reason codes at CTI OS server

- **Ready Button**. Click the ready button to put the agent in ready state (ready to accept calls).

- **Not Ready Button**. Click the not ready button to put the agent in not ready state (Unified ICM does not route calls to an agent in the not ready state). If Reason Codes are supported on the switch and configured on Unified ICM , a reason code dialog box pops up as shown below.

Figure 5: Reason Code Dialog for Not Ready



Use this dialog box to select a reason code to send with the not_ready request. You can configure reason codes at the CTI OS Server.

- **Work Ready Button**. Click this button to put the agent in the work ready or wrapup state. The behavior of this button depends on the wrapup mode support of the switch. On Unified CCE, the behavior is controlled by Unified ICM AgentDeskSettings (for more information, see *Administration Guide for Cisco Unified Contact Center Enterprise & Hosted*).

- **Work Not Ready Button**. Click this button to put the agent in the work not ready or wrapup state. The behavior of this button depends on the wrapup mode support of the switch. On Unified CCE, the behavior is controlled by Unified ICM AgentDeskSettings (for more information, see the *Administration Guide for Cisco Unified Contact Center Enterprise & Hosted*).

Related Methods

The following methods may be of interest to users of the agent state ActiveX control.

ReasonCodeState

This version of the ReasonCodeState method returns the enumReasonCodeState value.

Syntax

COM

```
HRESULT ReasonCodeState([out, retval] enumReasonCodeState *pVal)
```

VB

```
ReasonCodeState()As AgentStateCtlLib.enumReasonCodeState
```

.NET

```
AgentStateCtlLib.enumReasonCodeState ReasonCodeState()
```

Parameters

None

Return Value

Return value is enumReasonCodeState (this returns an Integer type).

ReasonCodeState

This version of the ReasonCodeState method sets the enumReasonCodeState value.

Syntax

COM

```
HRESULT ReasonCodeState([in] enumReasonCodeState newVal)
```

VB

```
ReasonCodeState = AgentStateCtlLib.enumReasonCodeState
```

.NET

```
ReasonCodeState = AgentStateCtlLib.enumReasonCodeState
```

Parameters

None

Return Value

None.

Following are the enumerated values for ReasonCodeState:

```
typedef enum {
                  eNotUsed,
                  eRequested,
                  eRequired,
    } enumReasonCodeState;
```

Related Events

The agent state control handles the following events.

OnAgentStateChanged

The OnAgentStateChanged message is generated when an agent state change event is fired.

Syntax

COM

```
HRESULT OnAgentStateChanged([in] LPDISPATCH vEventParam)
```

Parameters

vEventParam

Event fired to change the Agent state.

Return Value

None.

OnCtlEnabledChanged

The OnCtlEnabledChanged message is generated when control enabled is changed.

Syntax

COM

```
HRESULT OnCtlEnabledChanged(BOOL enabled)
```

Parameters

enabled

This is a control enabled changed value and returns a Boolean value.

OnEnableControlReceived

The OnEnableControlReceived message is generated when button enablement is changed.

Syntax

COM

```
void OnEnableControlReceived(BOOL enabled)
```

Parameters

enabled

This is a control enabled changed value and returns a Boolean value.

Following are the Button enablement masks return by OnEnableControlReceived method.

AgentSelectCtl

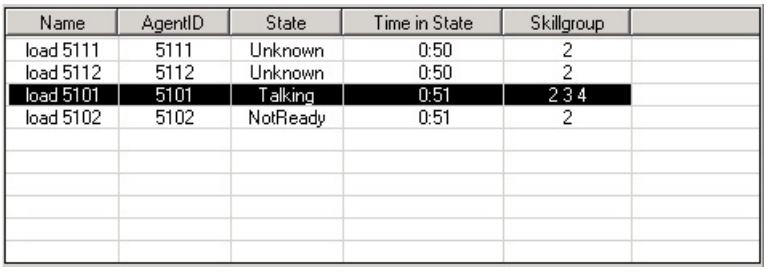
Table 3: OnEnableControlReceived Button Enablement Masks

m_eButtonType	nButtonMask
eLoginAgentBtn	ENABLE_LOGIN
eLoginSupervisorBtn	ENABLE_LOGIN
eLogoutBtn	ENABLE_LOGOUT or ENABLE_LOGOUT_WITH_REASON
eReadyBtn	ENABLE_READY

m_eButtonType	nButtonMask
eNotReadyBtn	ENABLE_NOTREADY or ENABLE_NOTREADY_WITH_REASON
eWorkReadyBtn	ENABLE_WORKREADY
eWorkNotReadyBtn	ENABLE_WORKNOTREADY

The agent select control is used for supervising agents and becomes active if the currently logged in agent is a supervisor. When a supervisor logs in, this grid based control displays all agent team members of a supervisor (configured by Unified ICM), including agent name, AgentID, AgentState, TimeInState, and SkillGroups. The TimeInState column resets in real-time as the agents change state. If an agent remains in a state for more than 10 minutes, the TimeInState figure displays in red.

Figure 6: Agent Select Grid Populated with Sample Data



The agent select control handles the following events:

- **OnNewTeamMember**. Informs the supervisor of a new team member or a team member change. This updates a row in the agentselect grid (add/remove agent).
- **OnMonitoredAgentStateChange**. Informs the supervisor of an agent state change. The new agentstate displays in the State column and the TimeInState Column is set to zero.
- **OnAgentInfo Event**.

A supervisor can select a “currently monitored agent” by clicking on an agent displayed in the grid. This causes a set monitored agent method call on the Agent object. Any supervisory action (for example logout monitored agent—see SupervisorOnlyCtl) is performed on the “currently monitored” agent.

Methods

Table 4: Available Methods for AgentSelectCtl

Method	Description
get_UserDefinedCell	If the column type is user defined, gets the text from the requested cell.
GetCellText	Gets the text from the requested cell in requested row.

Method	Description
GetColumnInfo	Gets the information about the requested column.
GetSelectedRow	Gets the selected row index.
SelectRow	Sets the requested row as selected.
set_ColumnHeader	Sets the column header of requested column with given text.
set_ColumnType	Sets the column type of requested column with given value.
set_ColumnWidth	Sets the column width of requested column with given value.
set_ UserDefinedCell	Sets the given text into the requested cell.
SetColumnInfo	Sets the given information for the requested column.

get_UserDefinedCell

If the column type is user defined, gets the text from the requested cell.

Syntax

COM

```
HRESULT UserDefinedCell(short nIndex, [out, retval] BSTR *pVal)
```

VB

```
get_UserDefinedCell(nIndex As Short) As String
```

.NET

```
System.String get_UserDefinedCell(System.Int16 nIndex)
```

Parameters

nIndex

This is a cell index number and an input parameter as type Short.

Return Value

Return type is String.

If the requested cell is not user defined type, it throws an Invalid Argument error.

GetCellText

Gets the text from the requested cell in requested row.

Syntax

COM

```
HRESULT GetCellText([in] int nRow, [in] int nCol, [out,retval] BSTR* bstrContent)
```

VB

```
GetCellText(nRow As Integer, nCol As Integer) As String
```

.NET

```
System.String GetCellText(System.Int16 nRow, System.Int16 nCol)
```

Parameters

nRow

This is a row index number and an input parameter as type Integer.

nCol

This is a column index number and an input parameter as type Integer.

Return Values

Return type is String.

GetColumnInfo

Gets the information about the requested column.

Syntax

COM

```
HRESULT GetColumnInfo([in] short nCol, [out] long *plColType, [out] int *iColWidth,
[out] int *iColTextAlign, [out] BSTR *bstrColTitle)
```

VB

```
GetColumnInfo(nCol As Short, ByRef plcoltype As Integer, ByRef icolwidth As Integer,
ByRef bstrcoltitle As String)
```

.NET

```
GetColumnInfo(System.Int16 nCol, System.Int32 plcoltype, System.Int32 icolwidth,
System.String bstrcoltitle)
```

Parameters

nCol

This is a column index number and an input parameter as type Short.

plcoltype

This is a column type value and an output parameter as type Integer.

icolwidth

This is a column width value and an output parameter as Integer.

bstrcoltitle

This is a column title text and an output parameter as type String.

Return Values

None.

GetSelectedRow

Gets the selected row index.

Syntax

COM

```
HRESULT GetSelectedRow([out,retval] int *nRow)
```

VB

```
GetSelectedRow() As Integer
```

.NET

```
System.Int32 GetSelectedRow()
```

Parameters

None

Return Values

Return type is Integer.

SelectRow

Sets the requested row as selected.

Syntax

COM

```
HRESULT SelectRow([in] int nRow, [out,retval] VARIANT_BOOL * bStatus)
```

VB

```
SelectRow(nRow As Integer) As Boolean
```

.NET

```
System.Boolean SelectRow(System.Int32 nRow)
```

Parameters

nRow

This is a row index number and an input parameter as type Integer.

Return Values

Return type is Boolean.

set_ColumnHeader

Sets the column header of requested column with given text.

Syntax

COM

```
HRESULT ColumnHeader(short nCol, [in] BSTR newVal)
```

VB

```
set_ColumnHeader(nCol As Short, newVal As String)
```

.NET

```
set_ColumnHeader(System.Int16 nCol, System.String newVal)
```

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing header text and an input parameter as type String.

Return Values

None.

set_ColumnType

Sets the column type of requested column with given value.

Syntax

COM

```
HRESULT ColumnType(short nCol, [in] short newVal)
```

VB

```
set_ColumnType(nCol As Short, newVal As Short)
```

.NET

```
set_ColumnType(System.Int16 nCol, System.Int16 newVal)
```

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing column type value and an input parameter as type Short.

Return Values

None.

set_ColumnWidth

Sets the column width of requested column with given value.

Syntax

COM

```
HRESULT ColumnWidth(short nCol, [in] short newVal)
```

VB

```
set_ColumnWidth(nCol As Short, newVal As Short)
```

.NET

```
set_ColumnWidth(System.Int16 nCol, System.Int16 newVal)
```

Parameters

nCol

This is a column index value and an input parameter as type Short.

newVal

This is a user passing column width value and an input parameter as type Short.

Return Values

None.

set_ UserDefinedCell

Sets the given text into the requested cell.

Syntax

COM

```
HRESULT UserDefinedCell(short nIndex, [in] BSTR newVal);
```

VB

```
set_UserDefinedCell(nindex As Short, newVal As String)
```

.NET

```
set_UserDefinedCell(System.Int16 nindex, System.String newVal)
```

Parameters

nindex

This is a cell index number and an input parameter as type Short.

newVal

This is a user passing text and an input parameter as type String.

Return Values

None.

SetColumnInfo

Sets the given information for the requested column.

Syntax

COM

```
HRESULT SetColumnInfo([in] short nCol, [in] long lColType, [in] int iColWidth, [in]
int iColTextAlign, [in] BSTR bstrColTitle)
```

VB

```
SetColumnInfo(nCol As Short, iColType As Integer,iColWidth As Integer, iColTextAlign
As Integer, bstrColTitle As String)
```

.NET

```
SetColumnInfo(System.Int16 nCol, System.Int32 iColType, System.Int32 iColWidth,
System.Int32 iColTextAlign, System.String bstrColTitle)
```

Parameters

nCol

This is a column index number and an input parameter as type Short.

iColType

This is a column type value and an input parameter as type Integer.

iColWidth

This is a column width value and an input parameter as type Integer.

iColTextAlign

This is a column text align value and an input parameter as type Integer.

bstrColTitle

This is a column title text and an input parameter as type String.

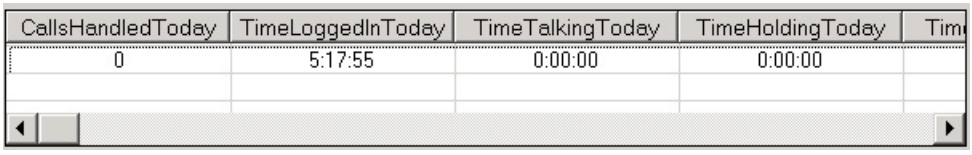
Return Values

None.

AgentStatisticsCtl

The AgentStatistics control is a grid based control displaying Unified ICM agent real time statistics. You can configure the displayed columns at CTI OS server (for more information, see the *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*). Also, you can adjust the update interval, which defaults to 10 seconds.

Figure 7: Agent Statistics Grid



Methods

Table 5: Available methods for AgentStatisticsCtl

Method	Description
get_UserDefinedCell	If the column type is user defined, gets the text from the requested cell.
GetCellText	Gets the text from the requested cell in requested row.
GetColumnInfo	Gets the information about the requested column.
set_ColumnHeader	Sets the column header of requested column with given text.
set_ColumnType	Sets the column type of requested column with given value.
set_ColumnWidth	Sets the column width of requested column with given value.
set_ UserDefinedCell	Sets the given text into the requested cell.
SetColumnInfo	Sets the given information for the requested column.

get_UserDefinedCell

If the column type is user defined, gets the text from the requested cell.

Syntax

COM: HRESULT UserDefinedCell(short nIndex, [out, retval] BSTR *pVal)

VB: get_UserDefinedCell(nIndex As Short) As String

.NET: System.String get_UserDefinedCell(System.Int16 nIndex)

Parameters

nIndex

This is a cell index number and an input parameter as type Short.

Return Value

Return type is String.

If the requested cell is not user defined type, it throws an Invalid Argument error.

GetCellText

Gets the text from the requested cell in requested row.

Syntax

COM: HRESULT GetCellText([in] int nRow, [in] int nCol, [out,retval] BSTR* bstrContent)

VB: GetCellText(nRow As Integer, nCol As Integer) As String

.NET: System.String GetCellText(System.Int16 nRow, System.Int16 nCol)

Parameters

nRow

This is a row index number and an input parameter as type Integer.

nCol

This is a column index number and an input parameter as type Integer.

Return Value

Return type is String.

GetColumnInfo

Gets the information about the requested column.

Syntax

COM: HRESULT GetColumnInfo([in] short nCol, [out] long *plColType, [out] int *iColWidth, [out] int *iColTextAlign, [out] BSTR *bstrColTitle)

VB: GetColumnInfo(nCol As Short, ByRef plcoltype As Integer, ByRef icolwidth As Integer, ByRef bstrcoltitle As String)

.NET: GetColumnInfo(System.Int16 nCol, System.Int32 plcoltype, System.Int32 icolwidth, System.String bstrcoltitle)

Parameters

nCol

This is a column index number and an input parameter as type Short.

plcoltype

This is a column type value and an output parameter as type Integer.

icolwidth

This is a column width value and an output parameter as Integer.

bstrcoltitle

This is a column title text and an output parameter as type String.

Return Value

None.

set_ColumnHeader

Sets the column header of requested column with given text.

Syntax

COM: HRESULT ColumnHeader(short nCol, [in] BSTR newVal)

VB: set_ColumnHeader(nCol As Short, newVal As String)

.NET: set_ColumnHeader(System.Int16 nCol, System.String newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing header text and an input parameter as type String.

Return Value

None.

set_ColumnType

Sets the column type of requested column with given value.

Syntax

COM: HRESULT ColumnType(short nCol, [in] short newVal)

VB: set_ColumnType(nCol As Short, newVal As Short)

.NET: set_ColumnType(System.Int16 nCol, System.Int16 newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing column type value and an input parameter as type Short.

Return Value

None.

set_ColumnWidth

Sets the column width of requested column with given value.

Syntax

COM: HRESULT ColumnWidth(short nCol, [in] short newVal)

VB: set_ColumnWidth(nCol As Short, newVal As Short)

.NET: set_ColumnWidth(System.Int16 nCol, System.Int16 newVal)

Parameters

nCol

This is a column index value and an input parameter as type Short.

newVal

This is a user passing column width value and an input parameter as type Short.

Return Value

None.

set_ UserDefinedCell

Sets the given text into the requested cell.

Syntax

COM: HRESULT UserDefinedCell(short nIndex, [in] BSTR newVal);

VB: set_UserDefinedCell(nindex As Short, newVal As String)

.NET: set_UserDefinedCell(System.Int16 nindex, System.String newVal)

Parameters

nindex

This is a cell index number and an input parameter as type Short.

newVal

This is a user passing text and an input parameter as type String.

Return Value

None.

SetColumnInfo

Sets the given information for the requested column.

Syntax

COM: HRESULT SetColumnInfo([in] short nCol, [in] long lColType, [in] int iColWidth, [in] int iColTextAlign, [in] BSTR bstrColTitle)

VB: SetColumnInfo(nCol As Short, iColType As Integer,iColWidth As Integer,iColTextAlign As Integer, bstrColTitle As String)

.NET: SetColumnInfo(System.Int16 nCol, System.Int32 iColType, System.Int32 iColWidth, System.Int32 iColTextAlign, System.String bstrColTitle)

Parameters

nCol

This is a column index number and an input parameter as type Short.

iColType

This is a column type value and an input parameter as type Integer.

iColWidth

This is a column width value and an input parameter as type Integer.

iColTextAlign

This is a column text align value and an input parameter as type Integer.

bstrColTitle

This is a column title text and an input parameter as type String.

Return Value

None.

AlternateCtl

Figure 8: AlternateCtl

The AlternateCtl is a button type control allowing the agent to send an alternate call request. Alternate is a compound action of placing an active call on hold and then retrieving a previously held call or answering an alerting (ringing) call on the same device. Alternate is a useful feature during a consult call.

AnswerCtl

The Answer Control is a button that provides UI for sending answer and release call requests. You can set the behavior (answer or release) via the ButtonType set from the property page as explained under AgentState controls.

Figure 9: Answer Icon:

Figure 10: Release Icon:

BadLineCtl

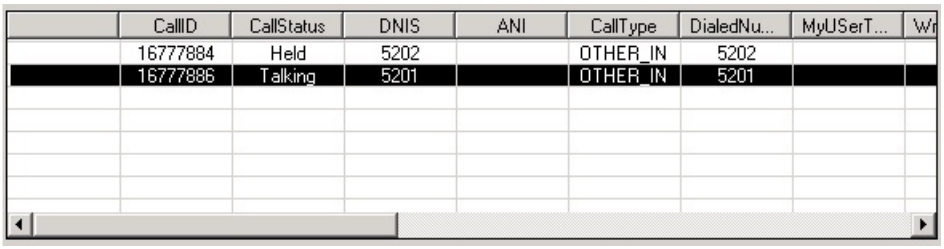
Figure 11: BadineCtl

The Bad Line Control is a button that provides a UI for reporting a Bad Line. This request generates a database entry in Unified ICM and is an indicator for voice/equipment problems.

CallAppearanceCtl

The CallAppearance Control is a grid based control displaying call information, including call status and call context data (for example, CallVariable1 through CallVariable10 and ECC variables).

Figure 12: CallAppearance Control Displaying Two Calls



Each incoming or outgoing call appears in one row in the grid. When a call first arrives, it usually shows a status of “Ringing” until it is answered. You can answer a call by a double click in the grid, similar to a click on the Answer Button. You can edit some columns in the CallAppearance grid if so configured (for example, the Columns displaying Callvariables) by selecting the cell you want to edit.

The grid can display multiple calls (see above). If the grid is displaying multiple calls, a user can click and select a call anywhere on the row where the call appears. This highlights the whole row displaying this call (for example, in the above figure the call with ID 16777886 is currently selected). Any button controls (for example, Answer, Release, Hold) enable or disable themselves based on the state the newly selected call is in.

The CallAppearance grid handles most call related events. It displays a call as soon as it receives an eCallBeginEvent. It updates the CallStatus and CallContext (CallVariables and ECC variables) on eCallDataUpdate and other call events (eServiceInitiated, eCallEstablished,). It erases the call from the grid when it receives an eCallEnd event.

The CallAppearance grid can be in one of two modes. In “normal” mode it shows any calls for the agent/supervisor logged in; in “monitored” mode (only for supervisor), the CallAppearance grid displays all calls for a currently monitored agent (see Agent Select grid). A supervisor can click and then select a “monitored call” on a row in the grid to perform supervisory functions like barge-in or intercept (see SupervisorOnly control).

Related Methods

The following methods may be of interest to users of the call appearance control.

Answer

For more information, see Call Object

GetValueInt

For more information, see CtiOs Object

GetValueString

For more information, see CtiOs Object

Related Events

The call appearance control handles the following events.

OnSetCurrentCallAppearance

The OnSetCurrentCallAppearance event is generated when the current call appearance object is changed.

Syntax

```
void OnSetCurrentCallAppearance([in] IDispatch * pCall);
```

Parameters

pCall

A Pointer to ICall COM Call object (pCall is a pointer to ICall).

Return Value

None.

Methods

Table 6: Available methods for CallAppearanceCtl

Method	Description
GetCellText	Gets the text from the requested cell in requested row.
GetSelectedRow	Gets the selected row index.
SelectRow	Sets the requested row as selected.
set_ColumnECCName	Sets the column ECC name of requested column with given text.
set_ColumnECCOffset	Sets the column Offset value of requested column with given value.
set_ColumnHeader	Sets the column header of requested column with given text.
set_ColumnWidth	Sets the column width of requested column with given value.
SetCellText	Sets the given text to the requested cell in requested row.

GetCellText

Gets the text from the requested cell in requested row.

Syntax

COM: HRESULT GetCellText([in] int nRow, [in] int nCol, [out,retval] BSTR* bstrContent)

VB: GetCellText(nRow As Integer, nCol As Integer) As String

.NET: System.String GetCellText(System.Int16 nRow, System.Int16 nCol)

Parameters

nRow

This is a row index number and an input parameter as type Integer.

nCol

This is a column index number and an input parameter as type Integer.

Return Value

Return type is String.

GetSelectedRow

Gets the selected row index.

Syntax

COM: HRESULT GetSelectedRow([out,retval] int *nRow)

VB: GetSelectedRow() As Integer

.NET: System.Int32 GetSelectedRow()

Parameters

None.

Return Value

Return type is Integer.

SelectRow

Sets the requested row as selected.

Syntax

COM: HRESULT SelectRow([in] int nRow, [out,retval] VARIANT_BOOL * bStatus)

VB: SelectRow(nRow As Integer) As Boolean

.NET: System.Boolean SelectRow(System.Int32 nRow)

Parameters

nRow

This is a row index number and an input parameter as type Integer.

Return Value

Return type is Boolean.

set_ColumnECCName

Sets the column ECC name of requested column with given text.

Syntax

COM: HRESULT ColumnECCName(short nCol, [in] BSTR newVal)

VB: set_ColumnECCName(nCol As Short, newVal As String)

.NET: set_ ColumnECCName (System.Int16 nCol, System.String newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing ECC Name text and an input parameter as type String.

Return Value

None.

set_ColumnECCOffset

Sets the column Offset value of requested column with given value.

Syntax

COM: HRESULT ColumnECCOffset(short nCol, [in] short nNewValue)

VB: set_ColumnECCOffset(nCol As Short, nNewValue As Short)

.NET: set_ColumnWidth(System.Int16 nCol, System.Int16 nNewValue)

Parameters

nCol

This is a column index number and an input parameter as type Short.

nNewVal

This is a user passing column width value and an input parameter as type Short.

Return Value

None.

set_ColumnHeader

Sets the column header of requested column with given text.

Syntax

COM: HRESULT ColumnHeader(short nCol, [in] BSTR newVal)

VB: set_ColumnHeader(nCol As Short, newVal As String)

.NET: set_ColumnHeader(System.Int16 nCol, System.String newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

nNewVal

This is a user passing header text and an input parameter as type String.

Return Value

None.

set_ColumnWidth

Sets the column width of requested column with given value.

Syntax

COM: HRESULT ColumnWidth(short nCol, [in] short newVal)

VB: set_ColumnWidth(nCol As Short, newVal As Short)

.NET: set_ColumnWidth(System.Int16 nCol, System.Int16 newVal)

Parameters

nCol

This is a column index value and an input parameter as type Short.

newVal

This is a user passing column width value and an input parameter as type Short.

Return Value

None.

SetCellText

Sets the given text to the requested cell in requested row.

Syntax

COM: HRESULT SetCellText([in] int nRow, [in] int nCol, [in] BSTR bstrContent, [out,retval] VARIANT_BOOL * bStatus)

VB: SetCellText(nRow As Integer, nCol As Integer, bstrContent As String) As Boolean

.NET: System. Boolean SetCellText(System.Int16 nRow, System.Int16 nCol, System.String bstrContent)

Parameters

nRow

This is a row index number and an input parameter as type Integer.

nCol

This is a column index number and an input parameter as type Integer.

bstrContent

This is a user passing cell text and an input parameter as type String.

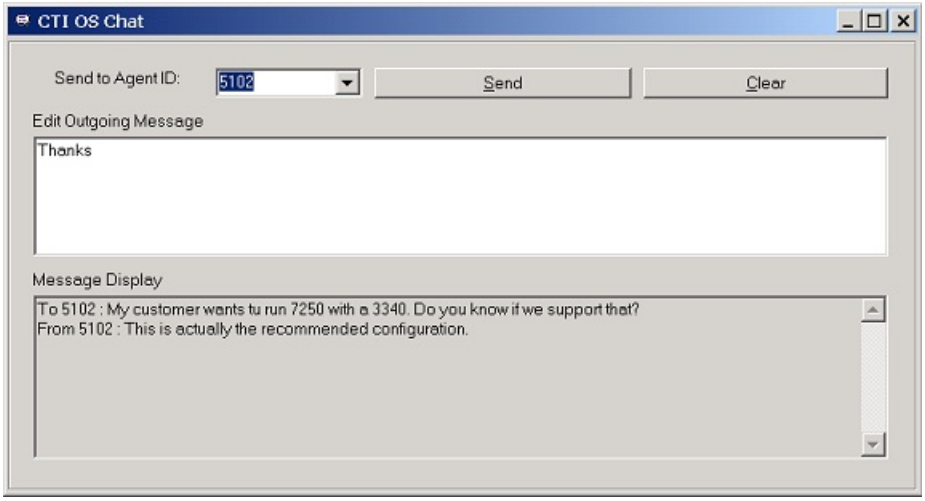
Return Value

Return type is Boolean.

ChatCtl

The Chat Control provides a UI to formulate and send text messages to a supervisor or (if allowed) other agents. The chat privileges are configurable at CTI OS server (for more information, see *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*).

Figure 13: Chat Control



You can specify an AgentID in the **Send to AgentID** field and then enter a message in the **Edit Outgoing Message** box. Click the **Send** Button to send the message. Incoming messages appear in the **Message Display**. Click the **Clear** button to clear the display.

The ChatCtl does not implement a button directly, but you can link it to a button through Visual Basic so that a click on the button pops up the ChatCtl.

Methods

Table 7: Available methods for ChatCtl

Method	Description
GetAddressee	Gets the current Addressee from the Send to Agent ID Combo box.
GetAllChatMessages	Gets the all chat messages from the Message Display Text Area.
GetChatMessageText	Gets the chat message from the Edit Outgoing Message Text Area.
OnMsgReceived	When message received from an Agent, appends the received message to the Message Display Text Area.
SendChatMessage	Sends the chat message to current Addressee in the Send to Agent ID Combo box.

Method	Description
SetAddressee	Sets the current Addressee to the Send to Agent ID Combo box.
SetChatMessageText	Sets the chat message to the Edit Outgoing Message Text Area.

GetAddressee

Gets the current Addressee from the Send to Agent ID Combo box.

Syntax

COM: HRESULT GetAddressee ([out,retval] BSTR* addressee)

VB: GetAddressee()As String

.NET: System.String GetAddressee()

Parameters

None.

Return Value

Return type is String.

GetAllChatMessages

Gets the all chat messages from the Message Display Text Area.

Syntax

COM: HRESULT GetAllChatMessages ([out, retval] BSTR* Messages)

VB: GetAllChatMessages() As String

.NET: System.String GetAllChatMessages()

Parameters

None.

Return Value

Return type is String.

GetChatMessageText

Gets the chat message from the Edit Outgoing Message Text Area.

Syntax

COM: HRESULT GetChatMessageText ([out, retval] BSTR* MessageText)

VB: GetChatMessageText() As String

.NET: System.String GetChatMessageText()

Parameters

None.

Return Value

Return type is String.

OnMsgReceived

When message is received from an Agent, appends the received message to the Message Display Text Area.

Syntax

COM: HRESULT OnMsgReceived ([in]BSTR from,[in]BSTR msg)

VB: OnMsgReceived(from As String, msg As String)

.NET: OnMsgReceived(System.String from, System.String msg)

Parameters

from

This is an Agent ID, who sends the message and is an input parameter as type String.

msg

This is a message text received form an Agent and is an input parameter as type String.

Return Value

None.

SendChatMessage

Sends the chat message to current Addressee in the Send to Agent ID Combo box.

Syntax

COM: HRESULT SendChatMessage([in] BSTR addressee, [in] BSTR msg)

VB: SendChatMessage(addressee As String, msg As String)

.NET: SendChatMessage (System.String addressee, System.String msg)

Parameters

addressee

This is as Agent ID, who receives the message and is an input parameter as type String.

msg

This is a message text sent to an Agent and is an input parameter as type String.

Return Value

None.

SetAddressee

Sets the current Addressee to the Send to Agent ID Combo box.

Syntax

COM: HRESULT SetAddressee ([in] BSTR addressee)

VB: SetAddressee(addressee As String)

.NET: SetAddressee(System.String addressee)

Parameters

addressee

This is as Agent ID, who receives the message and is an input parameter as type String.

Return Value

None.

SetChatMessageText

Sets the chat message to the Edit Outgoing Message Text Area.

Syntax

COM: HRESULT SetChatMessageText ([in] BSTR MessageText)

VB: SetChatMessageText(messageText As String)

.NET: SetChatMessageText (System.String messageText)

Parameters

messageText

This is an out going message text and is an input parameter as type String.

Return Value

None.

ConferenceCtl

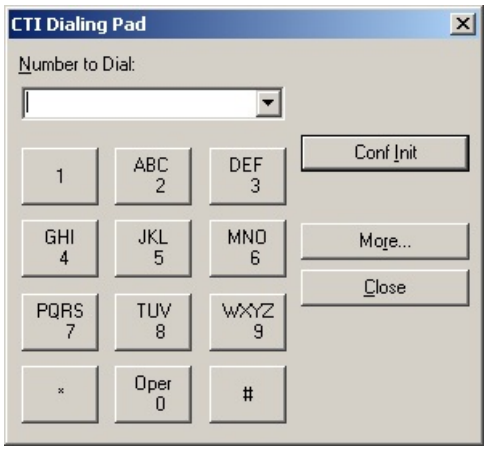
You can use the conference control to create a conference call. You can do this in either single step or consultative mode.

Figure 14: Icon for ConferenceInit:

Figure 15: Icon for Conference Complete:

Depending on the call status, selecting the Conference button once brings up the dialog box shown in the figure below (see also MakeCall dialog):

Figure 16: Conference Init Dialog



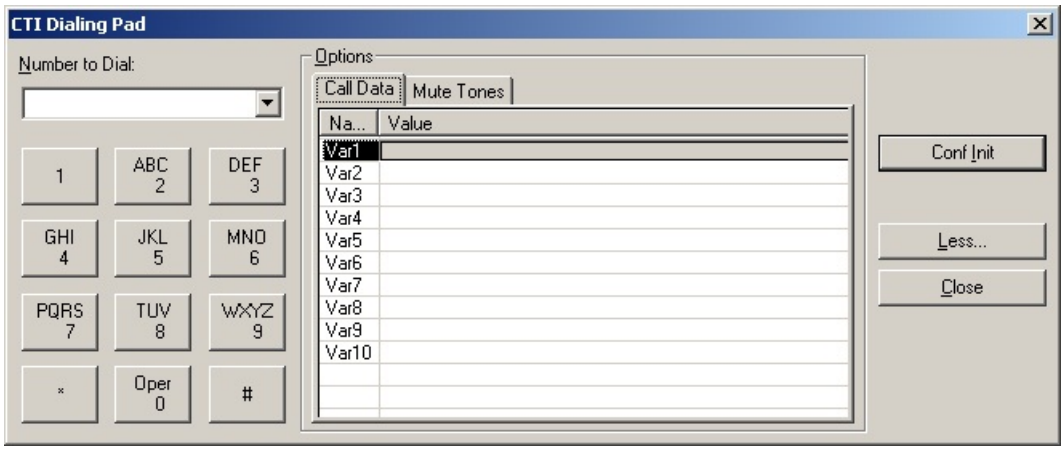
This dialog box is similar to the Make Call dialog box; you can initiate a consultative Conference (Conf Init) or place a Single Step Conference call.

Enter the number you wish to dial by either typing it into the **Number to Dial** text box or by clicks on the displayed keypad. After you enter the number you can click **Conf Init** to place a consultative conference call or **Single Step** to initiate a single step conference. This closes this dialog. If you choose to place a consultative call, the conference button changes to **Conference Complete**. You must click this button to complete the conference after talking to the consult agent.

The conference dialog box also has a **Mute Tones** section that you can use to suppress audio output of selected or all tones.

The **More** button brings up an additional section of the dialog displaying all CallVariables with any values set in the original call. The agent can double click the appropriate line in the Value column to change or add values to send with the consult call (see the figure below).

Figure 17: Expanded Dialog



EmergencyAssistCtl

The EmergencyAssistCtl is a button that provides a UI to place emergency or supervisor assist calls to a supervisor. On the Unified ICM side this functionality is implemented with a script (for more information, see *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*). The main difference between the emergency call and supervisor assist request is the script to be run. An agent can click this control whether they have a call or not. If the agent has an active customer call, clicking this button places a consult call to the supervisor. The "Conference Complete" as well as the "Transfer Complete" is enabled to allow the agent to either conference the supervisor into the call or to transfer the call to the supervisor. If configured, clicking this button can also cause a single step conference. You can set the behavior (emergency call or supervisor assist) via the ButtonType property set from the Property Page, as described under AgentState controls.

Figure 18: Emergency icon:

Figure 19: Supervisor Assist Icon:

HoldCtl

The HoldCtl is a button that provides a UI for sending hold and retrieve call requests. You can set the behavior (hold or retrieve) via the ButtonType property set from the Property Page, as described under AgentState controls.

Figure 20: Hold Icon

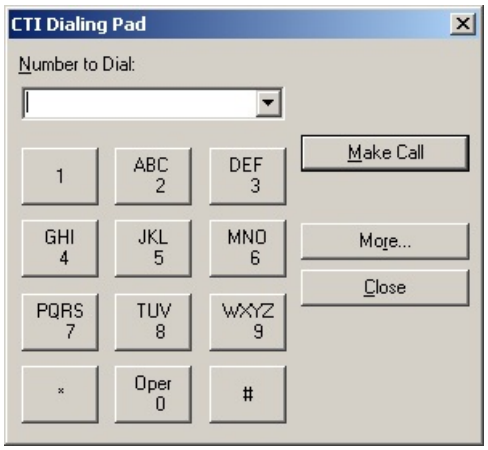
Figure 21: Retrieve Icon

MakeCallCtl

Figure 22: Make Call Icon

You use the MakeCallCtl to place calls and to generate DTMF tones. When you click this button it brings up the dialing pad dialog box to enter data and place a makecall request.

Figure 23: Dial Dialog

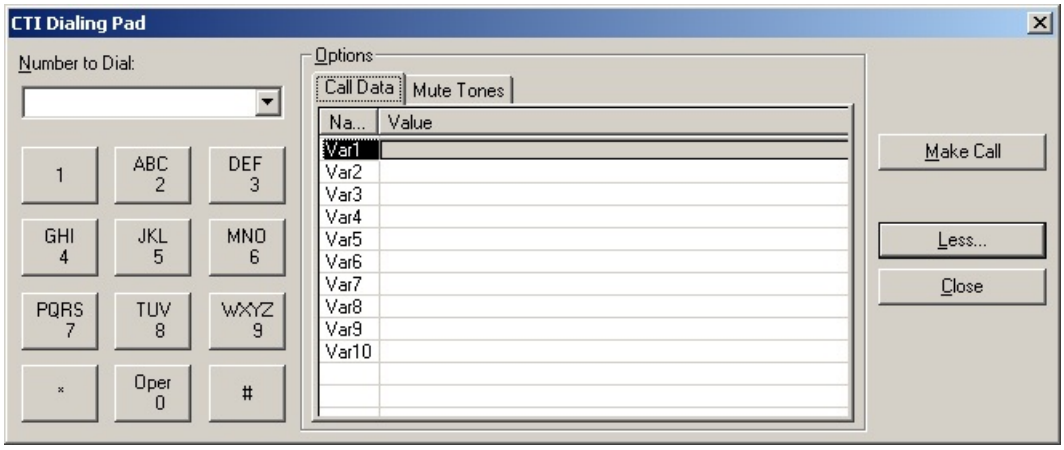


Enter the number you wish to dial by either typing it into the **Number to Dial** text box or click the numbers on the displayed keypad. After you enter the number you can click **Make Call** to send the MakeCall request.

This dialog box also has a **Mute Tones** section that allows you to suppress audio output of selected or all tones.

You can enter values for CallVariable1 through CallVariable10 and ECC Call Variables via the Dial Dialog. Click the **More** button on the dialog extends to display a grid listing all possible Call Variables. You can enter a value for each of these variables by double clicking the appropriate line in the Value column.

Figure 24: Expanded Dialog



If the agent is on a call while selecting the **Make Call** button, the dialpad appears without the MakeCall feature. The agent can then use the dialpad to play DTMF tones.

ReconnectCtl

The ReconnectCtl is a button control allowing the agent to send a Reconnect Call request. Reconnect is a useful feature during a consult call. If an agent has Call A held and Call B active, reconnect hangs up Call B and makes Call A active. In a consult call scenario, reconnect hangs up the consult call and returns to the original call.

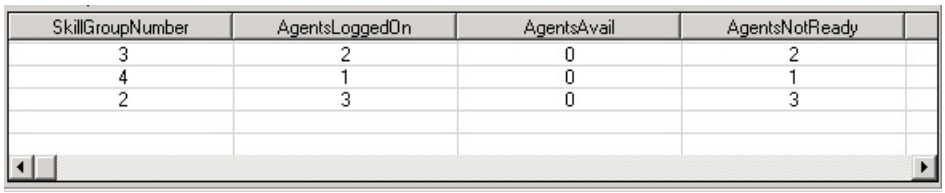
SkillgroupStatisticsCtl

The SkillGroupStatistics control is a grid based control displaying Unified ICM real time SkillGroup statistics.

You can configure the displayed columns at CTI OS server (for more information see *CTI OS System Manager Guide for Cisco Unified ICM/Contact Center Enterprise & Hosted*). You can configure the update interval, which defaults to 10 seconds.

If an agent belongs to multiple SkillGroups, each row displays statistics for one SkillGroup. For a supervisor this control displays all skillgroups in the team.

Figure 25: SkillgroupStatisticsCtl Displaying Sample Data for Three SkillGroups



Methods

Table 8: Available methods for SkillgroupStatisticsCtl

Method	Description
get_UserDefinedCell	If the column type is user defined, gets the text from the requested cell.
GetCellText	Gets the text from the requested cell in requested row.
GetColumnInfo	Gets the information about the requested column.
set_ColumnHeader	Sets the column header of requested column with given text.
set_ColumnType	Sets the column type of requested column with given value.
set_ColumnWidth	Sets the column width of requested column with given value.

Method	Description
set_ UserDefinedCell	Sets the given text into the requested cell.
SetColumnInfo	Sets the given information for the requested column.

get_UserDefinedCell

If the column type is user defined, gets the text from the requested cell.

Syntax

COM: HRESULT UserDefinedCell(short nIndex, [out, retval] BSTR *pVal)

VB: get_UserDefinedCell(nIndex As Short) As String

.NET: System.String get_UserDefinedCell(System.Int16 nIndex)

Parameters

nIndex

This is a cell index number and an input parameter as type Short.

Return Value

Return type is String.

If the requested cell is not user defined type, it throws an Invalid Argument error.

GetCellText

Gets the text from the requested cell in requested row.

Syntax

COM: HRESULT GetCellText([in] int nRow, [in] int nCol, [out,retval] BSTR* bstrContent)

VB: GetCellText(nRow As Integer, nCol As Integer) As String

.NET: System.String GetCellText(System.Int16 nRow, System.Int16 nCol)

Parameters

nRow

This is a row index number and an input parameter as type Integer.

nCol

This is a column index number and an input parameter as type Integer.

Return Value

Return type is String.

GetColumnInfo

Gets the information about the requested column.

Syntax

COM: HRESULT GetColumnInfo([in] short nCol, [out] long *plColType, [out] int *iColWidth, [out] int *iColTextAlign, [out] BSTR *bstrColTitle)

VB: GetColumnInfo(nCol As Short, ByRef plcoltype As Integer, ByRef icolwidth As Integer, ByRef bstrcoltitle As String)

.NET: GetColumnInfo(System.Int16 nCol, System.Int32 plcoltype, System.Int32 icolwidth, System.String bstrcoltitle)

Parameters

nCol

This is a column index number and an input parameter as type Short.

plcoltype

This is a column type value and an output parameter as type Integer.

icolwidth

This is a column width value and an output parameter as Integer.

bstrcoltitle

This is a column title text and an output parameter as type String.

Return Value

None.

set_ColumnHeader

Sets the column header of requested column with given text.

Syntax

COM: HRESULT ColumnHeader(short nCol, [in] BSTR newVal)

VB: set_ColumnHeader(nCol As Short, newVal As String)

.NET: set_ColumnHeader(System.Int16 nCol, System.String newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing header text and an input parameter as type String.

Return Value

None.

set_ColumnType

Sets the column type of requested column with given value.

Syntax

COM: HRESULT ColumnType(short nCol, [in] short newVal)

VB: set_ColumnType(nCol As Short, newVal As Short)

.NET: set_ColumnType(System.Int16 nCol, System.Int16 newVal)

Parameters

nCol

This is a column index number and an input parameter as type Short.

newVal

This is a user passing column type value and an input parameter as type Short.

Return Value

None.

set_ColumnWidth

Sets the column width of requested column with given value.

Syntax

COM: HRESULT ColumnWidth(short nCol, [in] short newVal)

VB: set_ColumnWidth(nCol As Short, newVal As Short)

.NET: set_ColumnWidth(System.Int16 nCol, System.Int16 newVal)

Parameters

nCol

This is a column index value and an input parameter as type Short.

newVal

This is a user passing column width value and an input parameter as type Short.

Return Value

None.

set_ UserDefinedCell

Sets the given text into the requested cell.

Syntax

COM: HRESULT UserDefinedCell(short nIndex, [in] BSTR newVal);

VB: set_UserDefinedCell(nindex As Short, newVal As String)

.NET: set_UserDefinedCell(System.Int16 nindex, System.String newVal)

Parameters

nindex

This is a cell index number and an input parameter as type Short.

newVal

This is a user passing text and an input parameter as type String.

Return Value

None.

SetColumnInfo

Sets the given information for the requested column.

Syntax

COM: HRESULT SetColumnInfo([in] short nCol, [in] long lColType, [in] int iColWidth, [in] int iColTextAlign, [in] BSTR bstrColTitle)

VB: SetColumnInfo(nCol As Short, iColType As Integer,iColWidth As Integer,iColTextAlign As Integer, bstrColTitle As String)

.NET: SetColumnInfo(System.Int16 nCol, System.Int32 iColType, System.Int32 iColWidth, System.Int32 iColTextAlign, System.String bstrColTitle)

Parameters

nCol

This is a column index number and an input parameter as type Short.

iColType

This is a column type value and an input parameter as type Integer.

iColWidth

This is a column width value and an input parameter as type Integer.

iColTextAlign

This is a column text align value and an input parameter as type Integer.

bstrColTitle

This is a column title text and an input parameter as type String.

Return Value

None.

StatusBarCtl

The CTI OS statusbar control displays information about the logged in agent as well as CTI OS specific details (see the following figure).

Figure 26: StatusBar Control Displaying Sample Data

The statusbar is separated into several panes. The panes are defined as follows:

- Pane 1: displays current extension and instrument.
- Pane 2: displays Agent ID.
- Pane 3: Message Waiting Indicator. If media termination is used and Voicemail is active, this pane displays “Voicemail” to indicate a Voicemail was left.
- Pane 4: displays Agent State.
- Pane 5: displays the CTI OS server currently connected to.

- Pane 6: displays overall status (online, offline).

SupervisorOnlyCtl

The SupervisorOnly Control provides buttons for Supervisor functions including Barge-In, Intercept, Logout Monitored Agent and make Monitored Agent Ready. You can set the behavior of the button in the General tab of the Property Page.

Logout Monitored Agent: Logs out the currently monitored agent (set for example via the AgentselectCtl). If the currently monitored agent has a call active, the request is queued and the agent is logged out as soon as the call ends:

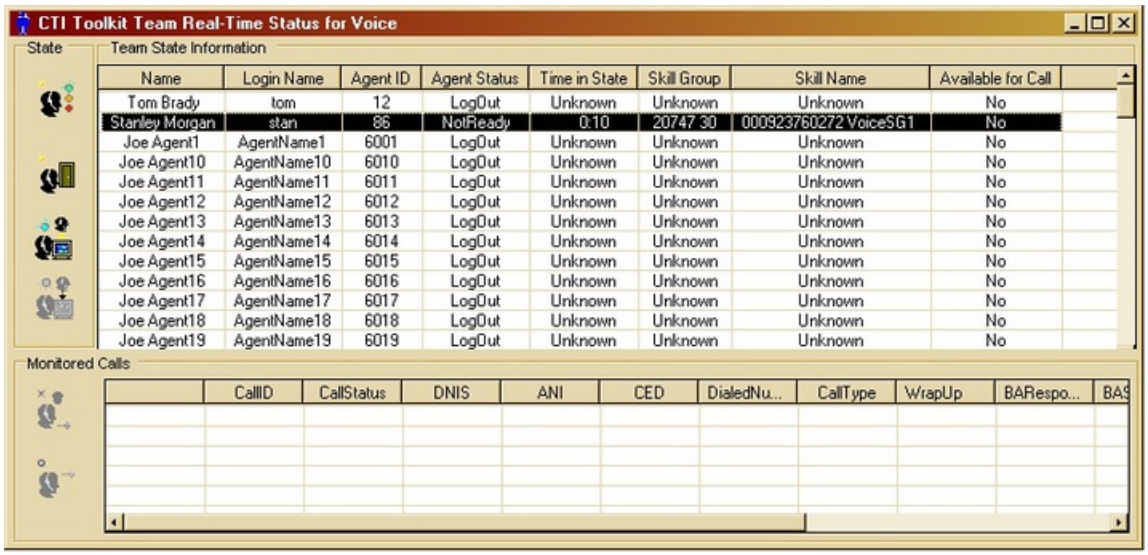
Set Monitored Agent Ready: Forces the currently monitored agent from the “not ready” state into the ready state:

Barge-In: Allows the supervisor to participate in the currently monitored call. The currently monitored call is selected via the CallAppearanceCtl (in monitor mode). Barge-in is really a conference on behalf of the monitored agent:

Intercept: You can only apply Intercept on a previously barged in call. The monitored agent is dropped out of the call and the supervisor is left with the customer in a call:

Together with the AgentSelectCtl and the CallAppearanceCtl (in monitor mode), you use the SupervisorOnlyCtl in the CTI OS Supervisor Desktop application to build the Agent Real Time Status window, as shown in the following figure.

Figure 27: Supervisor Softphone Agent-RealTime Status Window



This window shows the AgentSelectCtl and the CallappearanceCtl in monitor mode on the right side and four instances of the SupervisorOnlyCtl on the left side. From top to bottom they are: “Make Monitored Agent Ready” (disabled, since Agent 5101 is talking), “Logout monitored Agent”, Barge-in, and Intercept.

Start Silent Monitor: Initiates a silent monitor session with the currently monitored agent:

Stop Silent Monitor: Terminates the currently ongoing silent monitored session:

RecordCtl

The RecordCtl is a button that provides UI for Call Recording requests (start/stop recording). The requests are forwarded to CTI Server and are handled by a configured call recording service. To record a call you must select a current call (e.g. via the CallAppearanceCtl). After you click the record button it turns into record stop button.

Icon for Record Start:

Icon for Record Stop:

TransferCtl

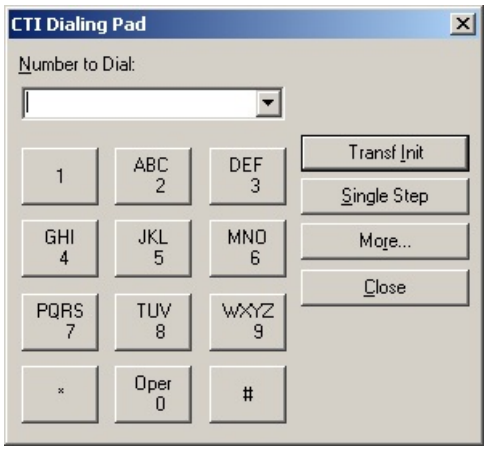
The TransferCtl is a button that provides UI to transfer a call in single step or consultative mode. The mechanism is the same as explained for the conference control.

Icon for TransferInit:

Icon for Transfer Complete:

Depending on call status, selecting the Transfer button once brings up the dialog box shown in the following figure (see also MakeCall dialog box):

Figure 28: Dial Dialog



This dialog box is similar to the Make Call dialog box. You can initiate a consultative Transfer (Transfer Init) or place a Single Step Transfer call.

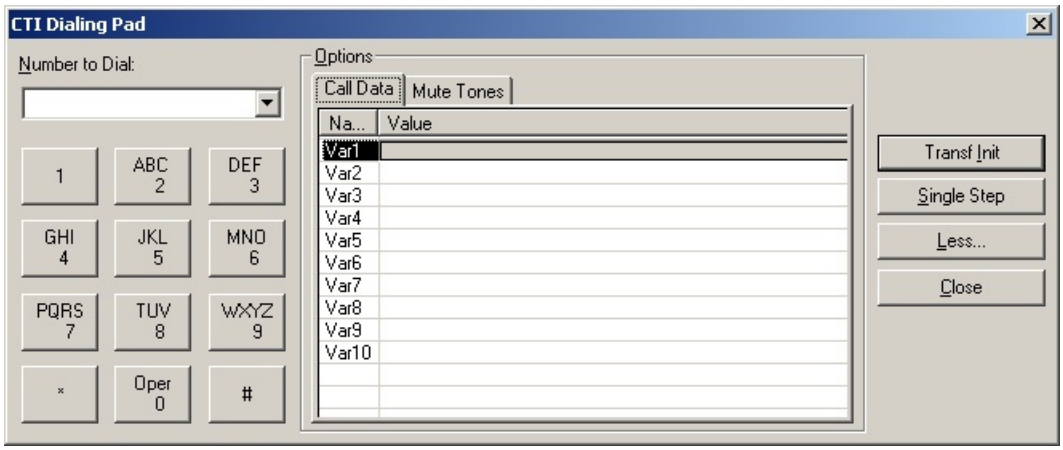
Enter the number you wish to dial by either typing it into the **Number to Dial** text box or click the numbers on the displayed keypad. After you enter the number you can click **Conf Init** to place a consultative transfer call or **Single Step** to initiate a single step transfer. This closes this dialog box. If you choose to place a

consultative call, the transfer button changes to **Transfer Complete**. You must click this button to complete the transfer after talking to the consult agent.

The transfer dialog box also has a **Mute Tones** section that allows you to suppress audio output of selected or all tones.

The **More** button brings up an additional section of the dialog box displaying all CallVariables and any values set in the original call. The agent can change or add values to send with the consult call by double clicking the appropriate line in the Value column (see the following figure).

Figure 29: Expanded Dialog



The Silent Monitor StandAlone ActiveX Control

The Silent Monitor StandAlone ActiveX Control provides an interface for easy integration with the CTI OS Silent Monitor functionality. You can use the ComObject in Visual Basic 6.0 as well as other host containers. This section demonstrates the use of this control in Visual Basic 6.0.

Note The Silent Monitor StandAlone ComObject is supported for use on Unified CCE only.

The Standalone ComObject wraps calls to the CTI OS Session as well as SilentMonitor manager. It provides the following methods (displayed in IDL format; IDL is the language used to define COM interfaces).

```
interface ISilentMonitor : IDispatch
{
    [id(1), helpstring("method Connect to CTIOS")] HRESULT Connect ([in] IArguments *
 args, [out] int* returnvalue);
    [id(2), helpstring("method Disconnect to CTIOS")] HRESULT Disconnect (/*[in]
 IArguments * args, [out] int* returnvalue*/);
    [id(3), helpstring("method StartMonitoring to CTIOS")] HRESULT StartMonitoring
([in] IArguments * args, [out] int* returnvalue);
    [id(4), helpstring("method StopMonitoring to CTIOS")] HRESULT StopMonitoring ([in]
 IArguments * args, [out] int* returnvalue);

};
```

Connect

The Connect method establishes a Monitor Mode Session with the specified CTI OS Server. The syntax and parameters are the same as the CTI OS session object Connect method (for more information, see Returns under AddEventListener method).

Disconnect

The Disconnect method disconnects an established session. This method has no required parameters. For more information about syntax and optional parameters, see CreateSilentMonitorManager in Session Object.

StartMonitoring

The StartMonitoring method starts a Silent Monitor Session.The StartMonitoring Arguments array contains the following parameters:

Table 9: StartMonitoring Arguments Array Parameters

Keyword	Value
AgentID	AgentID of the agent to be monitored.
Peripheralnumber	PeripheralID of the Peripheral to which the Agent is logged in.

Note If a pointer to the Agent object is available (for example, a m_MonitoredAgent), you can retrieve the PeripheralID via m_MonitoredAgent.GetValueInt ("PeripheralID").

StopMonitoring

The StopMonitoring method stops a Silent Monitor Session.The StopMonitoring Arguments array contains the same parameters as the StartMonitoring method (StartMonitoring, on page 48).

SilentMonitor Com Object Events

The ComObject fires the following events via a COM connection point event interface (again in IDL):

```
dispinterface _ISilentMonitorCtlEvents
{
         properties:
         methods:

      [id(1)] void OnConnection([in] IArguments *pIArguments);
```

```
      [id(2)] void OnConnectionFailure([in] IArguments *pIArguments);
     [id(5)] void OnMonitorModeEstablished([in] IArguments *pIArguments);

      [id(39)] void OnConnectionClosed([in] IArguments *pIArguments);
      [id(41)] void OnControlFailureConf([in] IArguments *pIArguments);
      [id(304)] void OnCtiOsFailure([in] IArguments *pIArguments);
     [id(502)] void OnCallRTPStartedEvent([in] IArguments *pIArguments);

     [id(503)] void OnCallRTPStoppedEvent([in] IArguments *pIArguments);

      [id(802)] void OnSilentMonitorStatusReportEvent([in] IArguments
*pIArguments);
    [id(803)] void OnStartSilentMonitorConf([in] IArguments *pIArguments);

    [id(804)] void OnStopSilentMonitorConf([in] IArguments *pIArguments);

     [id(805)] void OnSilentMonitorSessionDisconnected([in] IArguments
*pIArguments);
 //////////////////////////////////////////////////////////////////////////
};
```

Following is a brief description of each event. These events are described in detail in the Session Object and Silent Monitor Object sections of Event Interfaces and Events

Table 10: SilentMonitor Com Object Events

Event	Description
OnConnection	Indicates that the connect method was successful in establishing a connection.
OnConnectionFailure	Indicates that an active connection has failed. Can also indicate a bad parameter in the Connect method.
OnMonitorModeEstablished	Signals a successful call to SetMsgFilte. The call to Setmsgfilter is hidden by the Standalone control.
OnConnectionClosed	Disconnect was called and the connection is now closed.
OnControlFailureConf	A ControlFailureConf was received and can be handled.
OnCtiOsFailure	A CtiosFailure event was received. This could be Silent Monitor specific error code.
OnCallRTPStartedEvent, OnCallRTPStoppedEvent	RTP events have been received signaling the start and stop of the RTP streams.
OnSilentMonitorStatusReport Event	Reports status from a monitored client to the monitoring application.
OnStartSilentMonitorConf, OnStopSilentMonitorConf	Acknowledge that CTI OS handled the StartMonitoring and StopMonitoring request, respectively.

Event	Description
OnSilentMonitorSession Disconnected	Indicates that the Silent Monitor session has timed out on the monitoring side.

Deployment

The StandAlone Com Object is a COM dll that you must register on the client system via the Regsvr32 Silentmonitorctl.dll. You also require, ccnsmt.dll and the two standard CTI OS COM dlls (CTIOSClient.dll and Arguments.dll).

Sample Usage in Visual Basic 6.0

The following sample code assumes a VB 6.0 form with 4 buttons (Connect, Disconnect, StartMonitoring, and StopMonitoring). If the parameters are based on edit fields, the source code below is all that is needed to silent monitor via CTI OS. It is important to note that this control does not require supervisor privileges or even any login. The only event handler shown below (OnSessionDisconnected) is the one for a timed out Silent Monitor session.

```
Dim WithEvents SilentMonitorCtl As SILENTMONITORCTLLib.SilentMonitor
Dim m_Args As New Arguments
Const CIL_OK = 1

Private Sub btnConnect_Click()
    m_Args.clear
   m_Args.AddItem “CtiosA”, “localhost”
   m_Args.AddItem “portA”, “42028”
    Dim nRetVal As Long
    SilentMonitorCtl.Connect m_Args, nRetVal
    If nRetVal <> CIL_OK Then
       MsgBox “Connect returned error ” + Str(nRetVal)
    End If
End Sub

Private Sub btnDisconnect_Click()
    Dim nRetVal As Long
    SilentMonitorCtl.Disconnect
End Sub

Private Sub btnStartMonitoring_Click()
     m_Args.clear
     m_Args.AddItem “AgentId”, “1000”
     m_Args.AddItem “PeripheralID”, “5004”
    Dim nRetVal As Long
    SilentMonitorCtl.StartMonitoring m_Args, nRetVal
    If nRetVal <> CIL_OK Then
        MsgBox “StartMonitoring returned error ” + Str(nRetVal)
    End If
End Sub

Private Sub btnStopMonitoring_Click()
     m_Args.clear
     m_Args.AddItem “AgentId”, “1000”
     m_Args.AddItem “PeripheralID”, “5004”

    Dim nRetVal As Long
    SilentMonitorCtl.StopMonitoring m_Args, nRetVal
    If nRetVal <> CIL_OK Then
        MsgBox  “StopMonitoring returned error ” + Str(nRetVal)
```

```
    End If
End Sub

Private Sub SilentMonitorCtl_OnSessionDisconnected(ByVal pIArguments As
SILENTMONITORCTLLib.IArguments)
    MsgBox “SilentMonitorSession Disconnected Event”
End Sub
```