Call Object

Call Object

The Call object provides developers using the CTI OS Client Interface Library with an interface to Call behavior and control. The Call object enables you to perform all call behaviors, such as answering, hanging up, or transferring a call. The Call object represents one call connection of a call. For a call between two parties there are two call connections, and thus there are two distinct CIL Call objects.

The object stores specific call information as properties, including the ICMEnterpriseUniqueID, ANI, DNIS, Call variables, and ExpandedCallContext variables. The Call object is created in response to call events received at the CIL. The Call object properties and state are updated throughout the lifetime of the call connection.

For more information about accessing Call and ECC variables via the GetValue mechanism, see CIL Coding Conventions.

Current Call

The Client Interface Library uses the concept of a Current Call. The CTI OS Toolkit uses the Current Call concept as a way for the controls and the application to communicate with each other regarding which call is currently selected and is the one to act on. For example, if an agent has a call and receives a new Ringing call, they might select the Talking call on the grid. At this click, CallAppearanceMgr control calls SetCurrentCall() to make this call the Current Call. When the agent clicks the Hold control, this control calls GetCurrentCall() to obtain a call pointer through which to call the Hold() method. The agent can then select the Ringing call, which again causes the CallAppearanceMgr control to call SetCurrentCall() to make this new call the current call. Then, when the agent clicks the Answer control, this control again calls GetCurrentCall() to obtain a call pointer through which to call the Answer() method.

If your application uses Cisco's out-of-the-box button controls (for more information, see CTI OS ActiveX Controls), but not the CallAppearanceMgr grid control, you need to use SetCurrentCall() and GetCurrentCall() for the button controls to enable and disable correctly when switching between multiple calls.

Note The CurrentCall concept does not place any limitations on call control of non-current calls. All of the call behaviors implemented by method calls on the Call object work on any Call object that is available at the CIL, even if it is not the CurrentCall.

ECC Variables

The Unified ICM provides a customer-defined data layout for sending call context data with a call. This mechanism is called Expanded Call Context, or ECC. You define ECC variables in the Unified ICM Configuration Manager. You send ECC variables between Unified ICM servers in ECC payloads. After configuring an ECC variable, you must include it in an ECC payload before using it. The mechanism for accessing ECC variables from CTI OS is similar to accessing all other call variables.

To simplify the organization of properties on the Call object, the ECC variables are stored in their own Arguments structure which is nested in the Call object Arguments structure.

Passing Call Variables

- A consultative transfer is one in which the transferring or forwarding party either connects the caller to a ringing phone or speaks with the third party before connecting the caller to the third party. In a consultative transfer on the same peripheral gateway, if a variable is updated with data during the primary call, and the same variable is then updated with data during the transferred call, the call data from the initial call takes precedence and replaces the call data from the transferred call.
- For calls that are transferred between peripheral gateways, update call variables on the primary call before transferring the call. Only call variable information from the primary call is included in the route request to the other peripheral gateway. Any call variable information that you change after the call is transferred is lost because the call variable information was not included in the route request when the call was transferred.
- The Unified ICM call control variable map is a string that describes the mappings of a peripheral's call control variables to Unified ICM call control variables. You can edit this string to identify the call variables that an agent can change.

ECC Variable Value Retrieval

To retrieve an ECC variable from the Call object, first retrieve the ECC (Arguments) structure from the Call object using GetValueArray with keyword ECC. Then, retrieve the specific ECC variable required by using its name as the keyword to GetValueInt, GetValueArray, or GetValueString, depending on its type. The following is some sample code for C++ without COM:

```
Arguments * pECCData = NULL;string sMyECCVariable;
int nMyECCArrayVariable;

if (pCall->IsValid(CTIOS_ECC))
{
pCall->GetValueArray(CTIOS_ECC, &pECCData);

if (pECCData)
{
if (pECCData->IsValid("user.MyECC"))
pECCData->GetValueString->("user.MyECC", &sMyECCVariable);

if(pECCData->IsValid("user.MyArray[2]"))
pECCData->GetValueInt("user.MyArray[2]", &nMyECCArrayVariable);

pECCData->Release();
pECCData = NULL;
}
}
```

Sample code for VB without COM:

```
Dim MyECCData As CTIOSARGUMENTSLib.Arguments    Dim MyECCVariable As String
Dim MyECCArrayVariable As Integer

If MyCall.IsValid(CTIOS_ECC) = True Then
Set MyECCData = MyCall.GetValueArray(CTIOS_ECC)

If MyECCData.IsValid("user.MyECC") Then
MyECCVariable = MyECCData.GetValueString("user.MyECC")
End If

If MyECCData.IsValid("user.MyArray[2]") Then
MyECCArrayVariable = MyECCData.GetValueInt("user.MyArray[2]")
End If
End If
```

The same thing in Java is as follows:

```
if(Call != null){
Arguments rArgEcc = new Arguments();
rArgEcc =  Call.GetValueArray(CTIOS_ECC);
if(null != rArgEcc)
{
rArgEcc.NumElements();
Integer intVal =
rArgEcc.GetValueIntObj("user.MyECC");
String strVal =
rArgEcc.GetValueString("userMyArray[2]");
}
}
```

ECC Values

If you want to add ECC values to a call without deleting ones that are already set in the call, retrieve the ECC variables and then add the new ones as shown in C++ without COM:

```
Arguments & RequestArgs = Arguments::CreateInstance();
Arguments * pECCData = NULL;

// presumes that we have a Call object pointer in pCall
if (pCall->IsValid (CTIOS_ECC))
pCall->GetValueArray(CTIOS_ECC, &pECCData);

else
Arguments::CreateInstance(&pECCData);


pECCData->AddItem("user.MyECC", "FirstECCVariable");
pECCData->AddItem("user.MyArray[2]", 2222);

RequestArgs.AddItem(CTIOS_ECC, *pECCData);
pCall->SetCallData(RequestArgs);

RequestArgs.Release();
pECCData->Release();
```

The same thing in VB is as follows:

```
Dim MyRequestArgs As New CTIOSARGUMENTSLib.Arguments
Dim MyECCData As CTIOSARGUMENTSLib.Arguments

If MyCall.IsValid(CTIOS_ECC) Then
Set MyECCData = MyCall.GetValueArray(CTIOS_ECC)

Else
Set MyECCData = New CTIOSARGUMENTSLib.Arguments
End If

MyECCData.AddItem("user.MyECC", "FirstECCVariable")
MyECCData.AddItem("user.MyArray[2]", 2222)

MyRequestArgs.AddItem("ECC", MyECCData)

MyCall.SetCallData(MyRequestArgs)
```

The same thing in Java is as follows:

```
Arguments rRequestArgs = new Arguments();
if(Call != null)
{
   Arguments rArgEcc = Call.GetValueArray(CTIOS_ECC);
   if(null == rArgEcc)
   {
      rArgEcc = new Arguments();
   }
   rArgEcc.SetValue("user.MyEcc", 22222);
   rArgEcc.SetValue("user.MyArray[3]", "new data");

   rRequestArgs.SetValue(CTIOS_ECC, rArgEcc);

   Call.SetCallData(rRequestArgs);
}
```

Properties

The following table lists the available Call object properties.

Note The data type listed for each keyword is the standardized data type discussed in CTI OS CIL Data Types. For more information, see Table 1 for the appropriate language specific types for these keywords.

Table 1: Call Object Properties

Keyword	Type	Description
ANI	STRING	The calling line ID of the caller.
CallerEnteredDigits	STRING	The digits entered by the caller in response to VRU prompting.
CallStatus	SHORT	The current status of the call.
CallType	SHORT	The general classification of the call type.
CallVariable1	STRING	Call-related variable data.
CallVariable2	STRING	Call-related variable data.
CallVariable3	STRING	Call-related variable data.
CallVariable4	STRING	Call-related variable data.
CallVariable5	STRING	Call-related variable data.
CallVariable6	STRING	Call-related variable data.
CallVariable7	STRING	Call-related variable data.
CallVariable8	STRING	Call-related variable data.
CallVariable9	STRING	Call-related variable data.
CallVariable10	STRING	Call-related variable data.
CallWrapupData	STRING	Call-related variable data.
ClassIdentifier	INT	Private; for internal use only.
DialedNumber	STRING	The number dialed.
DNIS	STRING	The DNIS provided with the call.
ECC	ARGUMENTS	Arguments structure of key-value pairs of ECC variables.

Keyword	Type	Description
ICMEnterpriseUniqueID	STRING	Required only when the call is pre-routed.
LineType	SHORT	Indicates the type of the teleset line.
MeasuredCallQTime	INT	Number of seconds this call was in a local queue before being delivered to the agent.
PeripheralID	INT	The Unified ICM PeripheralID of the ACD where the call activity occurred.
RouterCallKeyCallID	INT	The call key created by the Unified ICM. The Unified ICM resets this counter at midnight .
Router CallKeyDay	INT	Together with the RouterCall KeyCallID field forms the unique 64-bit key for locating this call's records in the Unified ICM database . Only provided for Post-routed and Translation-routed calls.
ServiceID	INT	The Unified ICM ServiceID of the service that the call is attributed to. May contain the special value NULL_SERVICE when not applicable or not available.
ServiceNumber	INT	The service that the call is attributed to, as known to the peripheral. May contain the special value NULL_SERVICE when not applicable or not available.
SkillGroupID	INT	The system-assigned identifier of the agent SkillGroup the call is attributed to. May contain the special value NULL_SKILL_ GROUP when not applicable or not available.
SkillGroupNumber	INT	The optional, user-defined number of the agent SkillGroup the call is attributed to, as known to the peripheral. May contain the special value NULL_ SKILL_GROUP when not applicable or not available.

Keyword	Type	Description
UniqueObjectID	STRING	An object ID that uniquely identifies the Call object.
UserToUserInfo	STRING	The ISDN user-to-user information element.

Methods

The following table lists the available Call object methods.

Table 2: Call Object Methods

Method	Description
Alternate	Places the current call on hold and retrieves a previously held call.
Answer	Answers a call that is in the alerting or ringing state.
Clear	Clears a call, dropping all parties to the call.
ClearConnection	Hangs up a call, leaving other parties in a conference call. If there are only two parties on the call it clears the call.
Conference	Either establishes a three party conference call or adds a new party to an existing conference call.
DumpProperties	For more information, see CtiOs Object
GetAllProperties	For more information, see CtiOs Object
GetCallContext	Gets data associated with the call other than call and expanded call context (ECC) variables.
GetCallData	Obtains call and expanded call context (ECC) variables.
GetElement	For more information, see CtiOs Object
GetLastError (.NET only)	Returns the last error that occurred on the calling thread.
GetNumProperties	For more information, see CtiOs Object
GetPropertyName	For more information, see CtiOs Object
GetPropertyType	For more information, see CtiOs Object
GetValue methods	Retrieve a property from the Call object based on the property's name key.

Method	Description
Hold	Places a current call on hold.
IsValid	For more information, see CtiOs Object
MakeConsultCall	Places a current call on hold and makes a new call.
Reconnect	Clears the current call and then retrieves a held call.
Retrieve	Retrieves a held call.
SetCallData	Sets call and expanded call context (ECC) variables.
SendDTMFSignal	Requests the ACD to send a sequence of DTMF tones.
SingleStepConference	Performs a single step conference.
SingleStepTransfer	Performs a single step transfer.
Snapshot	Issues a server request to get the current call information, including call data and a list of associated devices and the connection state for the call of each device.
StartRecord	Starts recording of a call.
StopRecord	Stops recording of a call.
Transfer	Transfers a call to a third party.

Argument Parameters

The following rules apply to the optional_args and reserved_args parameters in Call object methods:

- In VB, you can ignore these parameters altogether. For example, you can treat the line:

```
Answer([reserved_args As IArguments]) As Long
```

  as follows:

```
Answer()
```

- To ignore these parameters in COM you must send a NULL, as shown:

```
Answer (NULL)
```

Alternate

The Alternate method combines the action of placing a talking call on hold and then retrieving a previously held call at the same device. If there are only two calls at the device, this method can be called via either the current or the held call.

Syntax

C++

```
int Alternate()
int Alternate(Arguments & reserved_args);
```

COM

```
HRESULT Alternate (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int
* errorcode );
```

VB

```
Alternate([reserved_args As IArguments]) As Long
```

Java

```
int Alternate(Arguments rArgs);
```

.NET

```
CilError Alternate(Arguments args)
```

Parameters

reserved_args

A valid Arguments object, which can be empty. Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

For switches that allow more than two calls at a device (for example G3), make this request only through the desired held call, because of the ambiguity caused by multiple held calls at the device.

You must make the Alternate request via a call whose status is either LCS_CONNECT or LCS_HELD or it fails.

The following events are received if this request is successful.

For the call making the Alternate request:

- OnAlternateCallConf event

For the originally current call:

- OnCallHeld event

For the originally held call:

- OnCallRetrieved event

The following events are received by the call making the Alternate request if this request fails:

- OnControlFailureConf event

Answer

The Answer method answers a call that is in the alerting or ringing state (i.e., call status of LCS_ALERTING).

Syntax

C++

```
int Answer()
int Answer(Arguments & reserved_args)
```

COM

```
HRESULT Answer (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int *
errorcode )
```

VB

```
Answer([reserved_args As IArguments]) As Long
```

Java

```
int Answer(Arguments rArgs)
```

.NET

```
CilError Answer(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

You can answer a call after the OnCallDelivered event is received. You must make the Answer request via a call whose call status LCS_ALERTING or it fails.

The following events are received if this request is successful:

- OnAnswerCallConf event
- OnCallEstablished event

The following events are received if this request fails:

- OnControlFailureConf event

Clear

The Clear method clears the call and drops all parties to the call.

Syntax

C++

```
int Clear()int Clear(Arguments & reserved_args);
```

COM

```
HRESULT Clear (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int *
errorcode )
```

VB

```
Clear([reserved_args As IArguments]) As Long
```

Java

```
int Clear(Arguments rArgs);
```

.NET

```
CilError Clear(Arguments args);
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

In the case of a multi-party Conference call, calling Clear() results in all of the parties to the call being hung up. (If this is not the desired behavior, see the ClearConnection method.) Under certain switches the Clear request is made via a call whose status is LCS_CONNECT or LCS_INITIATE or it fails. Many other switches allow the Clear method to be called via a call whose status is LCS_ALERTING or LCS_HOLD. It can never be made via a call whose status is LCS_NULL indicating that it is already cleared.

The following events are received if this request is successful:

- OnClearCallConf event
- OnCallCleared event

The following events are received if this request fails:

- OnControlFailureConf event

Note The Clear method is not supported on Unified CCE. Use of the Clear method with Unified CCE results in loss of third-party call control. To avoid this error, applications should use the ClearConnection method instead of Clear to hang up a call.

ClearConnection

If there are only two parties to the call, ClearConnection clears the call. However, for a multi-party conference call, only one connection is dropped, which is its own connection.

Syntax

C++

```
int ClearConnection()
int ClearConnection(Arguments & reserved_args);
```

COM

```
HRESULT ClearConnection (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/
int * errorcode)
```

VB

```
ClearConnection([reserved_args As IArguments]) As Long
```

Java

```
int ClearConnection(Arguments rArgs);
```

.NET

```
CilError ClearConnection(Arguments args);
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

As with the Clear method, under certain switches you must make the ClearConnection request via a call whose status is LCS_CONNECT or LCS_INITIATE or it fails. Many other switches allow the Clear method to be called via a call whose status is LCS_ALERTING or LCS_HOLD. It can never be made via a call whose status is LCS_NULL indicating that it is already cleared.

The following events are received if this request is successful:

- OnClearConnectionConf event
- OnCallConnectionlCleared event

If this is a two party call, these events are followed by:

- OnCallCleared event

The following events are received if this request fails:

- OnControlFailureConf event

Conference

The Conference method either begins a new conference call or adds an additional call to an existing conference call. When it begins a new conference call, it combines an original two-party call with a two-party consult call (where the two calls have a common party) into a single three party call. Only the common party (which is called the “Conference Controller”) can call this method to make the new conference call. You can call this method on either of the Conference Controller's calls.

Syntax

C++

```
int Conference();
int Conference(Arguments& optional_args)
```

COM

```
HRESULT Conference ( /*[in, optional]*/ IArguments *optional_args, (/*[out, retval]*/
int * errorcode )
```

VB

```
Conference([optional_args As IArguments]) As Long
```

Java

```
int Conference(Arguments optional_args)
```

.NET

```
CilError Conference(Arguments optional_args)
```

Parameters

optional_args

An optional input parameter, which is a pointer or reference to an Arguments array that contains a member with the string value that is the UniqueObjectID of the call to which this call is conferenced. If this argument is used, add it to the Arguments parameter with the keyword of “CallReferenceObjectID”. This is only necessary in an environment where there are multiple held calls and the request is made through the talking call. If the request is made through a specific held call in this scenario, or if there are only two calls at the device, this parameter is unnecessary.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

Before making this request, the original call must be in the held state and the consult call in the talking state or the request fails. Therefore, if the calls are alternated (see Alternate), they must be alternated again to return the two calls to their appropriate states.

If there are only two calls at the device, you can call this method using either the current or held call. For switches which allow more than two calls at a device (for example G3), make this request through the desired held call to avoid the ambiguity caused by multiple held calls at the device. Otherwise, indicate the desired held call using the optional parameter.

You must make the Conference request via a call whose call status is LCS_CONNECT or LCS_HELD or it fails.

On certain switches (notably Unified CCE), only the Conference Controller (the party that first initiated the conference call) can add additional parties to an existing conference call.

The following events are received if this request is successful:

- OnConferenceCallConf event
- OnCallConferenced event

The following events are received if this request fails:

- OnControlFailureConf event

GetCallContext

The GetCallContext method returns an Arguments array containing the values for call properties other than CallVariables and ECC Variables, such as ANI, DNIS, and the other properties listed in the following table.

Syntax

C++

```
int GetCallContext(Arguments& args)
```

COM

```
HRESULT GetCallContext (/*[out,retval]*/ IArguments ** args)
```

VB

```
GetCallContext (CTIOSCLIENTLib.IArguments args)
```

Java

```
Arguments GetCallContext()
```

.NET

```
Arguments GetCallContext()
```

Parameters

args

C++, COM, and VB: An output parameter containing a reference or a pointer to an Arguments array containing any of the members in the following table that are present in the call.

Return Value

C++, COM, and VB: Default HRESULT return values. For more information, see CIL Coding Conventions.

Java/.NET: A reference to an Arguments array that, on return, holds name/value pairs from the following table. You can access any of these parameters included from the Arguments array using the associated keyword.

Table 3: GetCallContext Arguments Array Contents

Keyword	Type	Description
ANI	STRING	The calling line ID of the caller.
CallerEnteredDigits	STRING	The digits entered by the caller in response to VRU prompting.
CallType	SHORT	The general classification of the call type.
CallWrapupData	STRING	Call-related wrapup data.
ConnectionCallID	UINT	The Call ID value assigned to this call by the peripheral or the Unified ICM.
DialedNumber	STRING	The number dialed.
DNIS	STRING	The DNIS provided with the call.
ICMEnterpriseUniqueID	STRING	A unique identifier for this contact throughout the enterprise. This can track a single customer contact across multiple sites, for example, when a call is transferred between agents.
ServiceID	INT	The Unified ICM identifier for the Service to which this call was routed.
ServiceNumber	INT	The ACD number of the Service to which this call was routed.
SkillGroupID	INT	The system-assigned identifier for the SkillGroup to which this call was routed.
SkillGroupNumber	INT	An optional, user-defined number of the SkillGroup at the ACD to which this call was routed.
UniqueObjectID	STRING	A unique object ID for the call.
UserToUserInfo	STRING	The ISDN user-to-user information element.

Remarks

This is a convenience method to call and get all of a call's non-CallVariable data at one time. If only certain data members are desired, call the appropriate GetValue method for each instead.

GetCallData

The GetCallData method returns the values ofCallVariable1 through CallVariable10 and all of the ECC (Extended CallContext) variables.

Syntax

C++

```
int GetCallData(Arguments& args)
```

COM

```
HRESULT GetCallData (/*[out,retval]*/ IArguments ** args)
```

VB

```
GetCallData (CTIOSCLIENTLib.IArguments args)
```

Java

```
Arguments GetCallData()
```

.NET

```
Arguments GetCallData()
```

Parameters

args

C++, COM, and VB: An output parameter containing a reference or a pointer to an Arguments array containing the call data, as described under Remarks.

Return Value

C++, COM, and VB: Default HRESULT return values. For more information, see CIL Coding Conventions.

Java/.NET: A reference to an Arguments array that, on return, holds parameters described under Remarks.

Remarks

This is a convenience method to call and get all of a call's CallVariables (1 through 10) and ECC Call Variables at one time. If only certain call variables are desired, call the appropriate GetValue method for each instead.

Access the data in the following way:

- To access the values for individual CallVariables from the arguments parameter, use GetValueString with either the keywords of “CallVariable1” through “CallVariable10”.

To access ECC call data, use the following procedure:

- First, get the ECC variables as a whole from the arguments parameter, using GetValueArray with the keyword “ECC”. This returns another Arguments array that is nested in the Arguments array returned from GetCallData.

- To access an individual ECC scalar variable from this Arguments array, use the appropriate GetValueString, GetValueInt, etc. depending on the variable's type, using the string keyword “user.VariableName”.
- To access an individual ECC array variable from this Arguments array, use the appropriate GetValueString, GetValueInt, etc. depending on the variable's type, using the string keyword “user.ArrayName[n]”, where n is a zero based integer that notes the offset in the array.

Hold

The Hold method holds a current call.

Syntax

C++

```
int Hold()
int Hold(Arguments & reserved_args);
```

COM

```
HRESULT Hold (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int *
errorcode )
```

VB

```
Hold([reserved_args As IArguments]) As Long
```

Java

```
Arguments Hold(Arguments rArgs)
```

.NET

```
Arguments Hold(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

You must make the Hold request via a call whose call status is LCS_CONNECT or it fails.

The following events are received if this request is successful:

- OnHoldCallConf event
- OnCallHeld event

The following events are received if this request fails:

- OnControlFailureConf event

MakeConsultCall

The MakeConsultCall method initiates the combined action of placing the associated current call on hold and then making a new call. By default, the call context data (including call variables) of the current call is used to initialize the context data of the new consultation call. The application can override some or all of the original call context in the consultation call by providing the desired values in this request.

The simplest form of the request only requires a dialed number and a consult type. The request can also include optional parameters, as listed in the following table.

Syntax

C++

```
int MakeConsultCall (Arguments& args))
```

COM

```
HRESULT MakeConsultCall (/*[in]*/ IArguments *args, /*[out, retval]*/ int * errorcode)
```

VB

```
MakeConsultCall (args As CTIOSCLIENTLib.IArguments) As Long
```

Java

```
int MakeConsultCall(Arguments args)
```

.NET

```
CilError MakeConsultCall(Arguments args)
```

Parameters

args

An output parameter of either a reference or a pointer to an Arguments array that contains parameters from the following table. Any of these parameters included are added to the Arguments array using the associated key word.

Table 4: MakeConsultCall Parameters

Parameter	Type	Description
DialedNumber (required)	STRING, maximum length 40	Dialed number; the number to be dialed to establish the new call.
ConsultType (required)	INT	A value specifying whether this consult call is in preparation for either a transfer or a conference, as specified in the ConsultType Table.
CallPlacementType (optional)	STRING, maximum length 40	A value specifying how the call is to be placed identified in Table 5: CallPlacementType Values, on page 20.

Parameter	Type	Description
CallMannerType (optional)	INT	A value specifying additional call processing options identified in Table 6: CallMannerType Values, on page 20.
CallOption (optional)	INT	A value from Table 7: CallOption Values, on page 20 specifying additional peripheral-specific call options.
FacilityType (optional)	INT	A value from Table 8: FacilityType Values, on page 21 indicating the type of facility to be used.
Priority (optional)	BOOL	Set this field to TRUE if the call should receive priority handling.
PostRoute (optional)	BOOL	When this field is set to TRUE, the Post-Routing capabilities of the Unified ICM determine the new call destination.
UserToUserInfo (optional)	STRING, maximum length 40	The ISDN user-to-user information.
CallVariable1 (optional)	STRING, maximum length 40	Call variable data that is set in the new call in place of the corresponding data in the current call.
...	...	...
CallVariable10 (optional)		
ECC	ARGUMENTS	ECC data that is set in the new call in place of the corresponding data in the current call.
CallWrapupData (optional)	STRING, maximum length 40	Call-related wrapup data.
FacilityCode (optional)	STRING, maximum length 40	A trunk access code, split extension, or other data needed to access the chosen facility.
AuthorizationCode (optional)	STRING, maximum length 40	An authorization code needed to access the resources required to initiate the call.
AccountCode (optional)	STRING, maximum length 40	A cost-accounting or client number used by the peripheral for charge-back purposes.

Table 5: CallPlacementType Values

CallPlacementType	Description	Value
CPT_UNSPECIFIED	Use default call placement.	0
CPT_LINE_CALL	An inside line call.	1
CPT_OUTBOUND	An outbound call.	2
CPT_OUTBOUND_NO_ ACCESS_CODE	An outbound call that does not require an access code.	3
CPT_DIRECT_POSITION	A call placed directly to a specific position.	4
CPT_DIRECT_AGENT	A call placed directly to a specific agent.	5
CPT_SUPERVISOR_ASSIST	A call placed to a supervisor for call handling assistance.	6

Table 6: CallMannerType Values

CallMannerType	Description	Value
CMT_UNSPECIFIED	Use default call manner.	0
CMT_POLITE	Attempt the call only if the originating device is idle.	1
CMT_BELLIGERENT	Always attempt the call, disconnecting any currently active call.	2
CMT_SEMI_POLITE	Attempt the call only if the originating device is idle or is receiving dial tone.	3

Table 7: CallOption Values

CallOption	Description	Value
COPT_UNSPECIFIED	No call options specified, use defaults.	0
COPT_CALLING_ AGENT_ONLINE	Attempt the call only if the calling agent is “online” (available to interact with the destination party).	1
COPT_CALLING_ AGENT_RESERVED	Attempt the call only if ACDNR on the calling agent's set is activated.	2

CallOption	Description	Value
COPT_CALLING_ AGENT_NOT_ RESERVED	Attempt the call only if ACDNR on the calling agent's set is not activated.	3
COPT_CALLING_ AGENT_BUZZ_BASE	Applies a buzz to the base of the telephone set as the call is initiated.	4
COPT_CALLING_ AGENT_BEEP_HSET	Applies a tone to the agent headset as the call is initiated.	5
COPT_SERVICE_ CIRCUIT_ON	Applies a call classifier to the call (ACM ECS)	6

Table 8: FacilityType Values

FacilityType	Description	Value
FT_UNSPECIFIED	Use default facility type.	0
FT_TRUNK_GROUP	Facility is a trunk group.	1
FT_SKILL_GROUP	Facility is a skill group or split.	2

Table 9: AnsweringMachine Values

AnsweringMachine	Description	Value
AM_UNSPECIFIED	Use default behavior.	0
AM_CONNECT	Connect call to agent when call is answered by an answering machine.	1
AM_DISCONNECT	Disconnect call when call is answered by an answering machine.	2
AM_NONE	Do not use answering machine detection.	3
AM_NONE_NO_ MODEM	Do not use answering machine detection, but disconnect call if answered by a modem.	4
AM_CONNECT_NO_MODEM	Connect call when call is answered by an answering machine, disconnect call if answered by a modem.	5

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

You must make the MakeConsultCall request via a call whose call status is LCS_CONNECT or it fails. Calling MakeConsultCall successfully results in the same events as a successful MakeCall called on the agent.

The following events are received if this request is successful:

For the call making the MakeConsultCallRequest:

- OnMakeConsultCallConf event
- OnCallHeld event

For the newly created outgoing consult call:

- OnBeginCall event
- OnServiceInitiated event
- OnCallOriginated event
- OnCallDelivered event

For the new connection that is ringing as a result of the consult call:

- OnBeginCall event
- OnCallDelivered event

The following events are received if this request fails:

- OnControlFailureConf event

Reconnect

The Reconnect method combines the action of releasing a current call and then retrieving a previously held call at the same device. If there are only two calls at the device, this method can be called via either the talking or the held call.

Syntax

C++

```
int Reconnect()
int Reconnect(Arguments & reserved_args)
```

COM

```
HRESULT Reconnect (/*[in,optional]*/ IArguments * reserved_args, (/*[out, retval]*/ int
* errorcode )
```

VB

```
Reconnect([reserved_args As IArguments]) As Long
```

Java

```
int Reconnect(Arguments rArgs)
```

.NET

```
CilError Reconnect(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

For switches that allow more than two calls at a device (for example G3), make this request only through the desired held call because of the ambiguity caused by multiple held calls at the device.

You must make the Alternate request via a call whose status is either LCS_CONNECT or LCS_HELD or it fails.

The following events are received if this request is successful:

For the call making the Reconnect request:

- OnReconnectCallConf event

For the originally current call:

- OnCallConnectionCleared event
- OnCallCleared event
- OnCallEnd event

For the originally held call:

- OnCallRetrieved event

The following events are received by the call making the Alternate request if this request fails:

- OnControlFailureConf event

Retrieve

The Retrieve method unholds a held call.

Syntax

C++

```
int Retrieve()
int Retrieve(Arguments & reserved_args)
```

COM

```
HRESULT Retrieve (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int
* errorcode )
```

VB

```
Retrieve([reserved_args As IArguments]) As Long
```

Java

```
int Retrieve(Arguments rArgs)
```

.NET

```
CilError Retrieve(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

You must make the Retrieve request via a call whose call status is LCS_HELD or it fails.

The following events are received if this request is successful:

- OnRetrieveCallConf event
- OnCallRetrieved event

The following events are received if this request fails:

- OnControlFailureConf event

SendDTMFSignal

The SendDTMFSignal method requests that the ACD send a sequence of DTMF tones.

Syntax

C++

```
int SendDTMFSignal(Arguments& args)
```

COM

```
HRESULT SendDTMFSignal (/*[in]*/ args *arguments, /*[out, retval]*/ int * errorcode)
```

VB

```
SendDTMFSignal (args As CTIOSCLIENTLib.IArguments, errorcode As Long)
```

Java

```
int SendDTMFSignal(Arguments rArgs)
```

.NET

```
CilError SendDTMFSignal(Arguments args)
```

Parameters

args

An input parameter of either a reference or a pointer to an Arguments array containing parameters from following table. You can add any of these parameters included to the Arguments array using the associated key word.

Table 10: SendDTMFSignal Parameters

Parameter	Type	Description
DTMFString (required)	STRING. maximum length 32	The sequence of tones to be generated.
ToneDuration (optional)	INT	Specifies the duration in milliseconds of DTMF digit tones. Use 0 to take the default. Can be ignored if the peripheral is unable to alter the DTMF tone timing.
PauseDuration (optional)	INT	Specifies the duration in milliseconds of DTMF inter-digit spacing. Use 0 to take the default. Can be ignored if the peripheral is unable to alter the DTMF tone timing.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

- The OnSendDTMFSignalConf event is received if this request succeeds.
- The OnControlFailureConf event is received if this request fails.

SetCallData

The SetCallData method sets any or all of a call's CallVariables (1 through 10) and ECC data at one time.

Note

- When writing a custom application, in any language, call variables are not blanked out if it they are set to a NULL value. While the application attempts to clear any call variable using a NULL value, the CTI OS server application ignores the NULL value call variables and does not pass them to the CTI Server application. As a result, the call variables set to NULL are not reset.
- To clear the value of a call variable, set its value to a blank character. Setting the call variable to a single space character places a space in the call variable's values for the duration of the call. This space is considered a NULL value by the application.

Syntax

C++

```
int SetCallData(Arguments& args)
```

COM

```
HRESULT SetCallData (/*[in]*/ args *arguments, /*[out]*/ int * errorcode)
```

VB

```
SetCallData (args As CTIOSCLIENTLib.IArguments, errorcode As Long)
```

Java

```
int SetCallData(Arguments rArgs)
```

.NET

```
CilError SetCallData(Arguments args)
```

Parameters

args

An input parameter of either a reference or a pointer to an Arguments array containing parameters described under Remarks for GetCallData.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

You must specify the data for all elements in the Arguments array, not just those elements that you want to change. Failure to do so causes the unchanged elements to disappear.

The following events are sent if this request succeeds:

- OnSetCallDataConf
- OnCallDataUpdate

The OnControlFailureConf event is sent if this request fails.

SingleStepConference

The SingleStepConference method initiates a one-step conference without the intermediate consultative call so that when the called party answers, they are joined in the current call. This method requires a DialedNumber argument. This method is not supported under all switches.

Note The SingleStepConference method is not supported for the Unified CCE .

Syntax

C++

```
int SingleStepConference(Arguments& args)
```

COM

```
HRESULT SingleStepConference (IArguments *args, int * errorcode)
```

VB

```
SingleStepConference (args As CTIOSCLIENTLib.IArguments, errorcode As Long)
```

Java

```
int SingleStepConference(Arguments rArgs)
```

.NET

```
CilError SingleStepConference(Arguments args)
```

Parameters

args

An output parameter of either a reference or a pointer to an Arguments array containing parameters from the following table. You can add any of these parameters included to the Arguments array using the associated keyword.

Table 11: SingleStepConference Parameters

Parameter	Type	Description
DialedNumber (required)	STRING, maximum length 40	Dialed number; the number to be dialed to establish the new call.
CallPlacementType (optional)	STRING, maximum length 40	A value specifying how the call is to be placed identified in Table 5: CallPlacementType Values, on page 20.
CallMannerType (optional)	INT	A value specifying additional call processing options identified in Table 6: CallMannerType Values, on page 20.

Parameter	Type	Description
AlertRings (optional)	INT	The maximum amount of time that the call's destination will remain alerting, specified as an approximate number of rings. A zero value indicates that the peripheral default (typically 10 rings) should be used.
CallOption (optional)	INT	A value from Table 7: CallOption Values, on page 20 specifying additional peripheral-specific call options.
FacilityType (optional)	INT	A value from Table 8: FacilityType Values, on page 21 indicating the type of facility to be used.
AnsweringMachine (optional)	INT	A value from Table 9: AnsweringMachine Values, on page 21 specifying the action to be taken if the call is answered by an answering machine.
Priority (optional)	BOOL	Set this field to TRUE if the call should receive priority handling.
PostRoute (optional)	BOOL	When this field is set to TRUE, the Post-Routing capabilities of the Unified ICM determines the new call destination.
UserToUserInfo (optional)	STRING, maximum length 40	The ISDN user-to-user information.
CallVariable1 (optional)	STRING, maximum length 40	Set call variable data in the new call in place of the corresponding data in the current call.
...	...	...
CallVariable10 (optional)		
ECC	ARGUMENTS	Set ECC data in the new call in place of the corresponding data in the current call.
FacilityCode (optional)	STRING, maximum length 40	A trunk access code, split extension, or other data needed to access the chosen facility.
AuthorizationCode (optional)	STRING, maximum length 40	An authorization code needed to access the resources required to initiate the call.

Parameter	Type	Description
AccountCode (optional)	STRING, maximum length 40	A cost-accounting or client number used by the peripheral for charge-back purposes.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

The DialedNumber is the only required member necessary in the Arguments parameter. A SingleStepConference request fails if the call's status is not LCS_CONNECT.

The following events are received if this request is successful:

- OnAgentStateChange event (Hold)
- OnCallHeld event
- OnAgentStateChange event (Talking)
- OnBeginCall event
- OnServiceInitiated event
- OnCallOriginated event
- OnCallDelivered event
- OnCallConferenced event
- OnCallEnd event
- ConferenceCallConf event

The OnControlFailureConf event is received if this request fails.

SingleStepTransfer

The SingleStepTransfer method initiates a one-step transfer without the intermediate consultative call. When the called party answers the call, the called party is talking to the party to be transferred and the transferring party drops out of the call. The method requires a DialedNumber argument.

Syntax

C++

```
int SingleStepTransfer(Arguments& args)
```

COM

```
HRESULT SingleStepTransfer (/*[in]*/ IArguments * args, /*[out, retval]*/ int * errorcode)
```

VB

```
SingleStepTransfer (args As CTIOSCLIENTLib.IArguments, errorcode As Long)
```

Java

```
int SingleStepTransfer(Arguments rASrgs)
```

.NET

```
CilError SingleStepTransfer(Arguments args)
```

Parameters

args

An output parameter of either a reference or a pointer to an Arguments array containing parameters from Table 11: SingleStepConference Parameters, on page 27. You can add any of these parameters included to the Arguments array using the associated keyword.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Snapshot

The Snapshot method issues a server request to retrieve the current call information. If values are passed in the optional args parameter, the snapshot request returns the server's current call values for only the requested arguments. Otherwise all call information is returned, including the fields described under GetCallContext and GetCallData. For more information about OnCallDataUpdate, see OnCallDataUpdate in Event Interfaces and Events.

Syntax

C++

```
int Snapshot()
int Snapshot(Arguments & optional_args)
```

COM

```
HRESULT Snapshot (/*[in,optional]*/ IArguments * optional_args, (/*[out, retval]*/ int
* errorcode )
```

VB

```
Snapshot([optional_args As IArguments]) As Long
```

Java

```
int Snapshot(Arguments rArgs)
```

.NET

```
CilError Snapshot(Arguments Args)
```

Parameters

optional_args

An input parameter of either a pointer or a reference to an Arguments array.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

The current information about the call is received in the OnCallDataUpdate event.

- The OnCallDataUpdate event is received if this request is successful.
- The OnControlFailureConf event is received if this request fails.

StartRecord

The StartRecord method is used to start recording a call.

Syntax

C++

```
int StartRecord()
int StartRecord(Arguments & reserved_args);
```

COM

```
HRESULT StartRecord (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/
int * errorcode )
```

VB

```
StartRecord([reserved_args As IArguments]) As Long
```

Java

```
int StartRecord(Arguments rArgs)
```

.NET

```
CilError StartRecord(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

Calling this method causes the CTI Server to forward the request to one or more server applications that have registered the “Cisco:CallRecording” service as described in the CTI Server Message Reference Guide (Protocol Version 14) for Cisco Unified ICM/Contact Center Enterprise & Hosted (Protocol Version 14) for Cisco Unified ICM/Contact Center Enterprise & Hosted. It fails if there is no recording server available to CTIServer.

- The OnStartRecordingConf event is received if this request is successful.
- The OnControlFailureConf event is received if this request fails.

StopRecord

The StopRecord method is used to stop recording a call.

Syntax

C++

```
int StopRecord()
int StopRecord(Arguments & reserved_args);
```

COM

```
HRESULT StopRecord (/*[in,optional]*/ IArguments *reserved_args, (/*[out, retval]*/ int
* errorcode )
```

VB

```
StopRecord([reserved_args As IArguments]) As Long
```

Java

```
int StopRecord(Arguments rArgs)
```

.NET

```
CilError StopRecord(Arguments args)
```

Parameters

reserved_args

Not currently used, reserved for future use.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Value

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

Calling this method causes the CTIServer to forward the request to the server application with the SessionID received in the OnStartRecordingConf event if non-zero, or if that SessionID is zero, to one or more server applications that have registered the “Cisco:CallRecording” service as described in the CTI Server Message Reference Guide (Protocol Version 14) for Cisco Unified ICM/Contact Center Enterprise & Hosted. It fails if there is no recording server available to CTIServer.

- The OnStopRecordConf event is received if this request is successful.
- The OnControlFailureConf event is received if this request fails.

Transfer

The Transfer method transfers a call to a third party. You can call this method on either the held original call or the current consult call. If the device has only these two calls, the optional parameter is not necessary. At the end of a successful transfer, both of these calls are gone from the device. For more information, see the Conference method.

Syntax

C++

```
int Transfer();
int Transfer(Arguments& optional_args)
```

COM

```
HRESULT Transfer ( [in, optional] IArguments *optional_args, (/*[out, retval]*/ int *
errorcode )
```

VB

```
Transfer([optional_args As IArguments]) As Long
```

Java

```
int Transfer(Arguments rArgs)
```

.NET

```
CilError Transfer(Arguments args)
```

Parameters

optional_args

An optional input parameter containing a member with a string value that is the UniqueObjectID of the call that is participating in the transfer. If this argument is used, add it to the Arguments parameter with the keyword of “CallReferenceObjectID”. This is only necessary in an environment where there are multiple held calls and the request is made through the current call. If the request is made through a specific held call in this scenario, or if there are only two calls at the device, this parameter is unnecessary.

errorcode

An output parameter (return parameter in VB) that contains an error code from Table 1.

Return Values

Default CTI OS return values. For more information, see CIL Coding Conventions.

Remarks

Before making this request, the original call must be in the held state and the consult call in the talking state or the request fails. Therefore, if the calls are alternated (for more information, see Alternate), they must be alternated again to return the two calls into their appropriate states.

If there are only two calls at the device, call this method using either the current or held call. For switches that allow more than two calls at a device (for example G3), make this request only through the desired held call to avoid the ambiguity caused by multiple held calls at the device. Otherwise, indicate the desired held call by using the optional parameter.

You must make the Transfer request via a call whose call status is LCS_CONNECT or LCS_HELD or it fails.

The following events are received by the transfer initiator if this request is successful:

- OnCallTransferred event
- OnCallEnd event
- OnCallEnd event
- OnAgentStateChange event
- OnTransferCallConf event

The OnControlFailureConf event is received if this request fails.