Session Object

The Client Interface Library Session object is used to establish a connection to an active CTI OS server. The main functions of the Session object are:

- Managing the connection to the CTI OS Server
- Distributing events to the appropriate objects and event subscribers
- Creating and managing the collections of Agent, Call, and SkillGroup objects
- Automatically recovering from failures

Typically, an application has a single instance of the Session object, which all other CIL objects use to send and receive events. However, there are no restrictions on the number or types of Session objects one application can employ. It is possible, and sometimes desirable, to establish and manage multiple independent Sessions, for example to use multiple current event streams. If there is more than one Session object monitoring the same Agent or Call, each Session object receives its own events. The order in which events are received is not guaranteed when there are multiple Session objects.

For more information about using the Session object to connect with CTI OS Server, see CTI OS Server Connection in Building Your Custom CTI Application.

The Session object creates new Call, Agent, and SkillGroup objects upon receipt of an event for that object if the targeted object does not already exist. The Session object maintains collections of all Agents, Calls, SkillGroups, and WaitObjects. Object lifetime is managed by the Session object, and thus it is important that the client application not delete the objects, which would render the object reference invalid and lead to unpredictable results. When the Session is Released, the connection to CTI OS server is dropped. Any remaining Agent, Call, Skill Group, or WaitObjects are released.

The remainder of this chapter describes the data properties and interface methods of the Session object.

Session Object Properties

The following table lists the available Session properties.

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

Table 1: Session Properties

Keyword	Type	Description
CCMBasedSilentMonitor	INT	If this value is present and set to true, supervisor applications initiate silent monitor using the Agent.SuperviseCall() method. Agent applications do not need to do anything. If this value is not present, or set to false, supervisor and agent applications need to invoke silent monitor using the SilentMonitorManager object. This property only applies to the COM CIL.
ConnectedSince	INT	Time of day in milliseconds when connected.
ConnectionMode	INT	eAgentConnection (1), eMonitorConnection (2), or eNotConnected (0).
CtiosA	STRING	Name or TCP/IP address passed as CTI OS server A.
CtiosB	STRING	Name or TCP/IP address passed as CTI OS server B
CurrentAgent	object reference	Returns reference to current Agent object set by the SetAgent method. Object reference is incremented by one and must be released when no longer used.
CurrentCall	object reference	Valid only if in Agent Connect mode. When there is more than one call, this references the current call. The current call is the call selected. For more information, see CurrentCall in Call Object
CurrentPort	INT	TCP/IP address of the current connected CTI OS server. Can be port A or B.
CurrentServer	STRING	Name or TCP/IP address of the current connected CTI OS server. The value is blank when the client is not connected to any server. The name may be blank while attempting to reconnect after a lost connection. Otherwise, the name of the server is the name of CTI OS server A or B.
ForcedDisconnect	INT	The presence of this keyword,
Heartbeat	INT	Heartbeat time, expressed in seconds. If not set, default heartbeats are configurable on the CTI OS server.
LastError	INT	Last error code, if any. Otherwise this value is 0.

Keyword	Type	Description
MaxHeartbeats	INT	Max heartbeats that can be missed before switching CTI OS servers. Default is 3 missed heartbeats.
MessageFilter	STRING	The filter that controls the events received by the CIL.
Object References	ARGUMENTS	Array of object references maintained by the session object. Typically includes Agent References, CallReferences, and SkillGroupReferences. Can also include EmailReferences or ChatReferences.
PortA	INT	TCP/IP port for ctiosA.
PortB	INT	TCP/IP port for ctiosB.
TryingPort	INT	TCP/IP address of the server where a connection is being attempted. Can be port A or B.
TryingServer	STRING	Contains the name or TCP/IP address of the server where a connection is being attempted. The value is blank if no connection is being attempted (see CurrentServer). The name of the server is the name of CTI OS server A or B.
TryingSince	INT	Time of day in milliseconds when try began.

Methods

The following table lists the available session object methods.

Table 2: Session object methods

Method	Description
AddEventListener	Subscribes a .NET IGenericEvents object as a listener on a particular subscriber list.
AddListener methods	Registers the subscriber for an event listener.
Connect	Establishes a connection to a CTI OS server.
CreateSilentMonitorManager	Creates a SilentMonitorManager object instance.
CreateWaitObject	Creates and returns the pointer to a new CWaitObject.
DestroySilentMonitor Manager	Deletes a SilentMonitorManager object instance.
DestroyWaitObject	Destroys the specified wait object

Method	Description
Disconnect	Closes the connection to the CTI OS server.
DumpProperties	For more information, see CtiOs Object
GetAllAgents	Returns a collection of all the agents in the session.
GetAllCalls	Returns a collection of all the calls in the session.
GetAllProperties	For more information, see CtiOs Object
GetAllSkillGroups	Returns a collection of all the skill groups in the session.
GetCurrentAgent	Returns the currently selected agent.
GetCurrentCall	Returns the currently selected call.
GetCurrentSilentMonitor	Returns a pointer to the SilentMonitorManager object instance that is set as the current manager in the CTI OS session object.
GetElement	For more information, see CtiOs Object
GetNumProperties	For more information, see CtiOs Object
GetObjectFromObjectID	Returns a Call, Agent, or SkillGroup, given the object's UniqueObjectID.
GetPropertyName	For more information, see CtiOs Object
GetPropertyType	For more information, see CtiOs Object
GetValue	For more information, see CtiOs Object
GetValueArray	For more information, see CtiOs Object
GetValueInt	For more information, see CtiOs Object
GetValueString	For more information, see CtiOs Object
IsAgent	Checks the current agent and returns true if the current agent is an agent and not a supervisor.
IsCCMSilentMonitor	The IsCCMSilentMonitor method determines whether CTI OS was configured to use CCM based silent monitor. This method is only supported with the C++, Java, COM, and .Net CILs.
IsSupervisor	Checks the current agent and returns true if the current agent is a supervisor.
IsValid	For more information, see CtiOs Object
RemoveListener methods	Unregisters the subscriber from an event listener.

Method	Description
RequestDesktopSettings	Sends a message request to the CTI OS Server to retrieve the desktop settings configured for this site.
SetAgent	Sets an agent to a session object.
SetCurrentCall	Associates the current call to a session object.
SetCurrentSilentMonitor	Sets the SilentMonitorManager object instance specified as the current manager in the CTI OS session object.
SetMessageFilter	Sets the message filter that controls the set of events sent to the CIL.
SetSupervisorSilentMonitorMode	Forces supervisors into monitored mode.
SetValue	For more information, see CtiOs Object

AddEventListener (Java and .NET Only)

The AddEventListener method subscribes an IGenericEvents object as a listener on a particular subscriber list.

Syntax

Java

```
int AddEventListener(IGenericEvents Listener, int iListID)
```

.NET

```
CilError AddEventListener(IGenericEvents Listener, SubscriberList iListID)
```

Parameters

Listener

The IGenericEvents object that is subscribing for events.

ListID

The ID of the subscriber list to which the Listener is to be added.

Returns

A CtiOs_Enums.CilError code indicating success or failure.

AddListener Methods (C++ Only)

The AddListener methods register the subscriber as a listener to the specified set of events.

Syntax

```
int AddEventListener(Arguments & rArguments);
int AddSessionEventListener(ISessionEvents * pSessionEvents);
int AddCallEventListener(ICallEvents * pCallEvents);
int AddAgentEventListener(IAgentEvents * pAgentEvents);
int AddSkillGroupEventListener(ISkillGroupEvents * pSkillGroupEvents);
int AddButtonEnablementEventListener(IButtonEnablementEvents *
pButtonEvents);
int AddAllInOneEventListener(IAllInOne * pAllInOneEvents);
int AddSilentMonitorEventListener(ISilentMonitorEvents *
pSilentMonitorEvents);
int AddSessionEventGenericListener(IGenericEvents * pSessionEvents);
int AddCallEventGenericListener(IGenericEvents * pCallEvents);
int AddAgentEventGenericListener(IGenericEvents * pAgentEvents);
int AddSkillGroupEventGenericListener(IGenericEvents *
pSkillGroupEvents);
int AddButtonEnablementEventGenericListener(IGenericEvents *
pButtonEvents);
int AddAllInOneEventGenericListener(IGenericEvents * pAllInOneEvents);
int AddSilentMonitorEventGenericListener(IGenericEvents *
pSilentMonitorEvents);
```

Remarks

For more information, see Subscribe for Events for in C++ in Building Your Custom CTI Application

Connect

The Connect method establishes a connection with a CTI OS server.

Syntax

C++

```
int Connect(Arguments& args)
```

COM

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

VB

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

Java

```
int Connect(Arguments args)
```

.NET

```
CilError Connect(Arguments rArgs
```

Parameters

args

An Arguments array containing the connection parameters listed in the following table:

Table 3: Connect Parameters

Keyword	Type	Description
CtiosA	STRING	Name or TCP/IP address of CTI OS server A. If this value is not provided, the value of Ctios B is used. **Note** If values of neither Ctios A or Ctios B is provided, an error is returned.
CtiosB	STRING	Name or TCP/IP address of CTI OS server B. If this value is not provided, the value of Ctios A is used. **Note** If values of neither Ctios A or Ctios B is provided, an error is returned.
PortA (optional)	INT	TCP/IP port for ctiosA, default = 42028.
PortB (optional)	INT	TCP/IP port for ctiosB, default = 42028.
Heartbeat (optional)	INT	Heartbeat time, expressed in seconds. If not set, default heartbeats are configurable on CTI OS server.

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

A successful request results in an OnConnection event.

A failure results in an OnConnectionFailure event. This means that the CIL is in failover. The CIL continues to attempt to connect, alternating between hosts CTIOS_CTIOSA and CTIOS_CTIOSB until connection succeeds, at which point CIL fires OnConnection. If the application wishes to stop failover, it must call Disconnect.

In some cases, additional failure codes and events may occur:

- Connect returns a failure code of -1 if it cannot connect with the initial side of the duplexed CTI OS server pair chosen from the connect parameters. This error code requires no action on the part of the developer as the CIL automatically attempts to connect using the parameters corresponding to the other side of the duplexed pair.
- The CIL retries the connection attempt five times and then does not attempt to reconnect any longer. The final OnConnectionFailure event contains the keyword "FinalAttempt," which informs the client application that the CIL has discontinued its attempts to reconnect.

  Note This behavior only occurs after global settings download has completed. If global settings download is not complete, the CIL continues to retry until successful.

- The Connect method fires an OnCTIOSFailure event to the client indicating the current state of the system. This is in addition to OnConnection or OnConnectionFailure.

The following error codes can occur:

- CIL_OK - no obvious errors, application should wait for an event indicating whether or not Connect succeeded.
- CIL_FAIL - initial attempt to connect with host has failed. CIL fires OnConnectionFailure and goes into failover mode. CIL continues to attempt to connect, alternating between hosts CTIOS_CTIOSA and CTIOS_CTIOSB until connection succeeds at which point CIL fires OnConnection. If application wishes to stop failover, it must call Disconnect.
- E_CTIOS_INVALID_ARGUMENT - a null Arguments parameter was supplied. Connect is aborted. No events are fired.
- E_CTIOS_MISSING_ARGUMENT - indicates that method call provided no value for both CTIOS_CTIOSA or CTIOS_CTIOSB. At least one of these values must be provided. Connect is aborted. No events are fired.
- E_CTIOS_IN_FAILOVER - a previous call to connect failed and CIL is currently in failover attempting to establish a connection. This continues until a connection is established at which point the CIL sends OnConnection indicating that previous call to Connect succeeded. If the developer wishes to call Connect again with different parameters, they must call Disconnect prior to calling Connect again.
- E_CTIOS_MODE_CONFLICT - Session is not disconnected (i.e a previous call to Connect is in progress or session is already connected). Disconnect must be called before attempting to establish another connection. CIL may fire an OnConnection event corresponding to previous call to Connect if connection was in progress but will not fire one corresponding to this method call.
- E_CTIOS_SESSION_NOT_CONNECTED - unanticipated error. Connect is aborted. No events are fired.

CreateSilentMonitorManager

The CreateSilentMonitorManager method creates a SilentMonitorManager object instance. To delete the object you must call DestroySilentMonitorManager.

Syntax

C++

```
CSilentMonitorManager * CreateSilentMonitorManager(Arguments & args);
```

COM

```
HRESULT CreateSilentMonitorManager ( /*[in]*/ IArguments * args, /*[out,retval]*/
ISilentMonitorManager * * pISilentMonitor);
```

VB

```
CreateSilentMonitorManager (ByVal args as CTIOSCLIENTLIB.IArguments) As
CTIOSCLIENTLIB.ISilentMonitorManager
```

Java

```
Not available.
```

.NET

```
Not available.
```

Parameters

args

Arguments array that contains the parameters listed below. When any of these parameters are specified, the object is constructed with the corresponding property initialized to the specified value. If you want the object initialized with the default values, specify an empty array.

Table 4: CreateSilentMonitorManager parameters

Keyword	Type	Description
HeartbeatInterval	INT	Heartbeat interval for the silent monitor session.
HeartbeatTimeout	INT	Timeout for no activity.
MediaTerminationPort	INT	Required only if manager is used in monitoring mode. TCP/IP port where monitored conversation is sent for playback on system sound card.

Return Value

If successful, a CSilentMonitorManager object is returned. Otherwise, NULL is returned. To identify the specific error, check the value of the LastError Session property (Table 1: Session Properties, on page 2).

Remarks

Supported for use with Unified CCE only.

CreateWaitObject (C++ Java and .NET)

The CreateWaitObject method creates and returns the pointer to a new CWaitObject with the specified event mask.

Syntax

C++

```
CWaitObject * CreateWaitObject(Arguments & args);
```

Java

```
WaitObject CreateWaitObject(Arguments rObjParam)
```

.NET

```
WaitObject CreateWaitObject(Arguments rObjParam)
```

Parameters

args (C++). rObjParam (Java)

A reference to an Arguments object that contains the list of events the object wait for. The Arguments contain values where the keys are “Event1” through “EventN” and the values are the enumerated event IDs.

Return Values

If successful it returns a pointer to the new Wait object. Otherwise, it returns NULL.

For more information about CWaitObject, see Helper Classes

DestroySilentMonitorManager

The DestroySilentMonitorManager method deletes a SilentMonitorManager object instance.

Syntax

C++

```
int DestroySilentMonitorManager(CSilentMonitorManager * pSilentMonitor);
```

COM

```
HRESULT DestroySilentMonitorManager (/*[in]*/ ISilentMonitorManager * pSilentMonitor,
/*[out,retval]*/ int * errorcode);
```

VB

```
DestroySilentMonitorManager (ByVal pSilentMonitor As CTIOSCLIENTLIB.
ISilentMonitorManager) As Long
```

Java

```
Not available
```

.NET

```
Not available
```

Parameters

pSilentMonitor

Valid pointer to a SilentMonitorManager object created via CreateSilentMonitorManager.

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

Supported for use with Unified CCE only.

DestroyWaitObject (C++ Java and .NET)

The DestroyWaitObject method removes the specified CWaitObject from the Session and decrements its reference count.

Syntax

C++

```
void DestroyWaitObject(CWaitObject * pWaitObject)
```

Java

```
void DestroyWaitObject(WaitObject rWaitObj)
```

.NET

```
DestroyWaitObject(WaitObject rWaitObj)
```

Parameters

WaitObject

A pointer to the CWaitObject to be destroyed.

Return Values

None.

Remarks

For more information about CWaitObject, see Helper Classes

DisableSkillGroupStatistics (C++ Java and .NET)

The DisableSkillGroupStatistics method requests that sending real-time statistics to the session object be stopped.

Syntax

C++, Java

```
int DisableSkillGroupStatistics(Arguments & args)
```

.NET

```
CilError DisableSkillGroupStatistics(Arguments rArgs)
```

Parameters

args

This parameter has two required values for PeripheralId and SkillGroupNumber. For more information, see the Remarks section for a code example.

Return Value

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

Remarks

C++ code example:

```
Arguments & argsStatBroadcast =
Arguments::CreateInstance();argsStatBroadcast.AddItem(CTIOS_SKILLGROUPNUMBER,
 intSG);
argsStatBroadcast.AddItem(CTIOS_PERIPHERALID, m_periphID);
m_pSkGrStatSession->DisableSkillGroupStatistics ( argsStatBroadcast );
argsStatBroadcast.Release();
```

Disconnect

The Disconnect method disconnects the open connection to the CTI OS server. In Java and .NET, you can use the Disconnect method to interrupt failover.

Syntax

C++

```
void Disconnect (Arguments& args);
```

COM

```
HRESULT Disconnect (/* [in, optional */ IArguments *args, /*[out]*/ int * errorcode
);
```

VB

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

Java

```
int Disconnect(Arguments args)
```

.NET

```
CilError Disconnect(Arguments rArgs)
```

Parameters

args

An optional Arguments array containing the CTIOS_FORCEDDISCONNECT keyword, which forces a disconnect even if the Session object rejects the disconnect. Add this keyword to the array if the session mode is not set by SetAgent or SetSessionMode at the time of the disconnect.

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

DumpProperties

For more information about the DumpProperties method, see CtiOs Object.

EnableSkillGroupStatistics (C++ Java and .NET)

The EnableSkillGroupStatistics method starts sending real-time statistics to the session object. If the argument array is empty, then statistics for all skill groups are enabled. This is useful when a monitoring application needs to view all statistics without having to enumerate and loop over each statistic to enable it.

Syntax

C++, Java

```
EnableSkillGroupStatistics(Arguments & args)
```

.NET

```
CilError EnableSkillGroupStatistics(Arguments rArgs)
```

Parameters

args

This parameter has three required values for PeripheralId, SkillGroupNumber and SkillGroupPriority. See the Remarks section for a code example.

Return Value

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

Remarks

C++ code example:

```
Arguments & argsStatBroadcast =
Arguments::CreateInstance();
argsStatBroadcast.AddItem(CTIOS_SKILLGROUPNUMBER,intSG);
argsStatBroadcast.AddItem(CTIOS_PERIPHERALID, m_periphID);
argsStatBroadcast.AddItem(CTIOS_SKILLGROUPPRIORITY, priority);
m_pSkGrStatSession->EnableSkillGroupStatistics ( argsStatBroadcast );
argsStatBroadcast.Release();
```

GetAllAgents

The GetAllAgents method returns an array of object IDs. Each object ID is associated with an Agent object stored in the CIL.

The number of object IDs returned from this method depends on the number of agents that the CIL discovered through agent events. For example, a CIL used in an agent desktop application returns one ID, which is the ID of the agent currently logged into the desktop. A supervisor desktop returns the supervisor's ID as well as IDs for all agents on the supervisor's team. A monitor mode application filtering all agent events returns IDs for each agent known by the CTI OS Server.

Syntax

C++

```
Arguments & GetAllAgents()
```

COM

```
HRESULT GetAllAgents(/*[out, retval]*/ VARIANT *args)
```

VB

```
GetAllAgents (args As VARIANT)
```

Java

```
Arguments GetAllAgents()
```

.NET

```
Arguments GetAllAgents()
```

Parameters

args

COM/VB: A pointer to a VARIANT containing a SAFEARRAY of pointers to IAgents.

Return Values

COM/VB

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

Java/.NET

Returns NULL if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of an agent and a value that is a reference to a CilRefArg that is a pointer to the Agent object.

C++

An empty Arguments array if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of an agent and a value that is a reference to a CilRefArg that is a pointer to the Agent object.

Remarks

The following sample C++ code illustrates how to take the array returned from GetAllAgents() and use it to access the corresponding agents in the CIL's object cache. The example uses the C++ CIL:

```
Arguments &args = m_pSession->GetAllAgents() ;
// Iterate through all of the CILRefArg objects
// in the Arguments array.
//
for ( int i = 1 ; i <= args.NumElements() ; i++ )
{
    CILRefArg *pRefArg = NULL ;

    // Retrieve the CILRefArg at each position in the
    // array.
    //
```

```
    if ( args.GetElement(i, (Arg **)&pRefArg) )
    {
        if ( pRefArg != NULL )
        {
            // The value method will return a pointer
            // to the agent object referenced by the
            // CILRefArg.
            //
            CAgent *pAgent = (CAgent *)pRefArg->GetValue() ;

            cout << "-- Agent Properties --" << endl ;
            if ( pAgent == NULL )
            {
                cout << "NULL" << endl ;
            }
            else
            {
                cout << pAgent->DumpProperties().c_str() << endl ;
            }
            cout << "--" << endl ;
        }
    }
}
```

The following sample VB.NET code illustrates how to take the array returned from GetAllAgents() and use it to access the corresponding agents in the CIL's object cache. The example uses the .NET CIL:

```
Dim args As Argumentsargs = m_session.GetAllAgents()

' Iterate through all of the CILRefArg objects
' in the Arguments array.
'
Dim i As Integer
For i = 1 To args.NumElements()

    Dim refArg As CilRefArg

    ' Retrieve the CILRefArg at each position in the
    ' array.
    '
    If (args.GetElement(i, refArg)) Then

        If ((refArg Is Nothing) = False) Then

            ' The value method will return a reference
            ' to the agent object referenced by the
            ' CILRefArg.
            '
            Dim agent As Agent
            refArg.GetValue(agent)

            Console.Out.WriteLine("--")

            If (agent Is Nothing) Then
                Console.Out.WriteLine("Nothing")
            Else
                Console.Out.WriteLine(agent.DumpProperties())
            End If
```

```
            Console.Out.WriteLine("--")

        End If

    End If
Next
```

GetAllCalls

The GetAllCalls method returns an array of object IDs. Each object ID is associated with a Call object stored in the CIL.

The number of object IDs returned from this method depends on the number of calls that the CIL discovered through call events. For example, a CIL used in an agent desktop application returns IDs for all calls in which the agent is involved. A supervisor desktop returns IDs for any call in which the supervisor is involved as well as IDs for monitored calls. A monitor mode application filtering all call events returns IDs for each call known by the CTI OS Server.

Syntax

C++

```
Arguments & GetAllCalls()
```

COM

```
HRESULT GetAllCalls(/*[out, retval]*/ VARIANT *args)
```

VB

```
GetAllCalls (args As VARIANT)
```

Java

```
Arguments GetAllCalls()
```

.NET

```
Arguments GetAllCalls()
```

Parameters

args

COM /VB: A pointer to a VARIANT containing a SAFEARRAY of pointers to ICalls.

Return Values

COM/VB

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

Java/.NET

Returns NULL if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of a call and a value that is a reference to a CilRefArg that is a pointer to the Call object.

C++

An empty Arguments array if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of a call and a value that is a reference to a CilRefArg that is a pointer to the Call object.

Remarks

The following sample C++ code illustrates how to take the array returned from GetAllCalls() and use it to access the corresponding calls in the CIL's object cache. The example uses the C++ CIL:

```
Arguments &args = m_pSession->GetAllCalls() ;
// Iterate through all of the CILRefArg objects
// in the Arguments array.
//
for ( int i = 1 ; i <= args.NumElements() ; i++ )
{
    CILRefArg *pRefArg = NULL ;

    // Retrieve the CILRefArg at each position in the
    // array.
    //
    if ( args.GetElement(i, (Arg **)&pRefArg) )
    {
        if ( pRefArg != NULL )
        {
            // The value method will return a pointer
            // to the agent object referenced by the
            // CILRefArg.
            //
            CCall *pCall = (CCall *)pRefArg->GetValue() ;

            cout << "-- Call Properties --" << endl ;
            if ( pCall == NULL )
            {
                cout << "NULL" << endl ;
            }
            else
            {
                cout << pCall->DumpProperties().c_str() << endl ;
            }
            cout << "--" << endl ;
        }
    }
}
```

The following sample VB.NET code illustrates how to take the array returned from GetAllCalls() and use it to access the corresponding calls in the CIL's object cache. The example uses the .NET CIL:

```
Dim args As Argumentsargs = m_session.GetAllCalls()

' Iterate through all of the CILRefArg objects
' in the Arguments array.
'
Dim i As Integer
For i = 1 To args.NumElements()

    Dim refArg As CilRefArg

    ' Retrieve the CILRefArg at each position in the
    ' array.
    '
    If (args.GetElement(i, refArg)) Then

        If ((refArg Is Nothing) = False) Then

            ' The value method will return a reference
            ' to the call object referenced by the
            ' CILRefArg.
            '
            Dim aCall As Cisco.CtiOs.Cil.Call
            refArg.GetValue(aCall)

            Console.Out.WriteLine("--")

            Dim str As String

            If (aCall Is Nothing) Then
                Console.Out.WriteLine("Nothing")
            Else
                Console.Out.WriteLine(aCall.DumpProperties())
            End If

            Console.Out.WriteLine("--")

        End If

    End If
Next
```

GetAllProperties

For more information about the GetAllProperties method, see CtiOs Object.

GetAllSkillGroups

The GetAllSkillGroups method returns an array of object IDs. Each object ID is associated with a skill group stored in the CIL.

Syntax

C++

```
Arguments & GetAllSkillGroups()
```

COM

```
HRESULT GetAllSkillGroups(/*[out, retval]*/ VARIANT *args)
```

VB

```
GetAllSkillGroups (args As VARIANT)
```

Java, .NET

```
Arguments GetAllSkillGroups()
```

Parameters

args

C++, Java, and .NET: A pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of a skill group and a value that is a reference to a CilRefArg that is a pointer to the skill group object.

COM /VB: A pointer to a VARIANT containing a SAFEARRAY of pointers to ISkillGroups.

Return Values

COM/VB: Default CTI OS return values. For more information, see CIL Coding Conventions

Java/.NET: Returns NULL if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of a skill group and a value that is a reference to a CilRefArg that is a pointer to the skill group object.

C++: An empty Arguments array if the value requested is not found or if there is an error. If the method succeeds, it returns a pointer or a reference to an Arguments array where each member has a string key that is the UniqueObjectID of a skill group and a value that is a reference to a CilRefArg that is a pointer to the skill group object.

GetCurrentAgent

The GetCurrentAgent method returns the Agent specified when the Agent Mode connection was established. Use this method rather than GetValue(“CurrentAgent”).

Syntax

C++

```
Agent* GetCurrentAgent()
```

COM

```
HRESULT GetCurrentAgent(/*[out, retval]*/ IAgent *agent)
```

VB

```
GetCurrentAgent () As CTIOSCLIENTLib.IAgent
```

Java,.NET

```
Agent GetCurrentAgent()
```

Parameters

agent

An output parameter (return value in VB, C++, Java, and .NET) containing a pointer to a pointer to an IAgent that is the currently selected agent.

Return Values

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

Others: A pointer or reference to an agent that is the current agent. This method returns NULL if the value requested is not found or if there is an error.

The C++, Java, and .NET versions of this method return NULL if the value requested is not found or if there is an error.

GetCurrentCall

The GetCurrentCall method returns the call that is currently selected. You can use this method as a way for controls to communicate between each other which call is selected and acted on. Use this method rather than GetValue("CurrentCall").

Syntax

C++

```
CCall * GetCurrentCall()
```

COM

```
HRESULT GetCurrentCall(/*[out, retval]*/ ICall ** call)
```

VB

```
GetCurrentCall () As CTIOSCLIENTLib.ICall
```

Java/.NET

```
Call GetCurrentCall()
```

Parameters

call

An output parameter (return value in VB, C++, Java, and .NET) containing a pointer to a pointer to an ICall that is the currently selected call.

Return Values

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

Others: A pointer or reference to a Call that is the current call. This method returns NULL if the value requested is not found or if there is an error.

The C++, Java, and .NET versions of this method return NULL if the value requested is not found or if there is an error.

GetCurrentSilentMonitor

The GetCurrentSilentMonitor method returns a pointer to the SilentMonitorManager object instance that is set as the current manager in the session object.

Syntax

C++

```
CSilentMonitorManager * GetCurrentSilentMonitor();
```

COM

```
HRESULT GetCurrentSilentMonitor (/*[out,retval]*/ ISilentMonitorManager **
pSilentMonitor);
```

VB

```
GetCurrentSilentMonitor () As CTIOSCLIENTLIB. ISilentMonitorManager
```

Java,.NET

```
Not available
```

Return Values

Pointer to the current Silent Monitor Manager in the session object.

GetElement

For more information about the GetElement method, see CtiOs Object.

GetNumProperties

For more information about the GetNumProperties method, see CtiOs Object.

GetObjectFromObjectID

Given a string containing the UniqueObjectID of a call, an agent, or a skill group, the GetObjectFromObjectID method returns a pointer to the associated object.

Syntax

C++

```
bool GetObjectFromObjectID (string& uniqueObjectID,CCtiosObject ** object);
```

COM

```
HRESULT GetObjectFromObjectID (/*[in]*/ BSTR uniqueObjectID, /*[out]*/ IDispatch **
object, /*[out, retval]*/ VARIANT_BOOL * errorcode);
```

VB

```
GetObjectFromObjectID(uniqueObjectID As String, object as IDispatch) As Boolean
```

Java

```
CtiOsObject GetObjectFromObjectID(java.lang.String sUID)
```

.NET

```
System.Boolean GetObjectFromObjectID(System.String sUID, out CtiOsObject rObj)
```

Parameters

COM/C++/VB: uniqueObjectID

A string reference that contains the UniqueObjectID of the requested Call, Agent, or Skillgroup object.

.NET: sUID

A string reference that contains the UniqueObjectID of the requested Call, Agent, or Skillgroup object.

COM/C++: object

A pointer to either a CTIOSObject in C++ (which is a CILRefArg) or an IDispatch * pointing to either an ICall, an IAgent, or an ISkillGroup in COM.

.NET: rObj

A pointer to either a CTIOSObject in C++ (which is a CILRefArg) or an IDispatch * pointing to either an ICall, an IAgent, or an ISkillGroup in COM.

errorcode

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

Return Values

COM: Default HRESULT return value. For more information, see CIL Coding Conventions.

C++, VB, .NET: A boolean indicating success or failure of the method.

The Java version of this method returns NULL if the value requested is not found or if there is an error.

Remarks

Many events use UniqueObjectIDs instead of the objects themselves. Use this method to get the object if it is necessary for processing.

GetPropertyName

For more information about the GetPropertyName method, see CtiOs Object.

GetPropertyType

For more information about the GetPropertyType method, see CtiOs Object.

GetSystemStatus (Java .NET and C++ Only)

The GetSystemStatus method returns the current system status bitmask.

Syntax

Java/C++

```
int GetSystemStatus()
```

.NET

```
SystemStatus GetSystemStatus()
```

Parameters

None.

Returns

The current system status bitmask. For more information about the SystemStatus, see OnQueryAgentStateConf in Event Interfaces and Events.

GetValue Methods

For more information about the GetValue, GetValueArray, GetValueInt, and GetValueString methods, see CtiOs Object.

IsAgent

The IsAgent method determines whether the current agent is an agent rather than a supervisor.

Syntax

C++

```
bool IsAgent()
```

COM

```
HRESULT IsAgent (VARIANT_BOOL *bIsAgent)
```

VB

```
IsAgent () As Boolean
```

Java

```
boolean IsAgent()
```

.NET

```
bool IsAgent()
```

Parameters

bIsAgent

Output parameter (return parameter in VB) that returns true if the current AgentMode connection is for an agent and false if it is for a supervisor.

Return Values

If the current agent is an agent and not a supervisor it returns true, otherwise it returns false.

IsCCMSilentMonitor

The IsCCMSilentMonitor method determines whether CTI OS is configured to use Unified Communication Manager based silent monitor.

Syntax

C++

```
bool IsCCMSilentMonitor()
```

COM

```
HRESULT IsCCMSilentMonitor (VARIANT_BOOL * IsCCMSilentMonitor)
```

Java

```
boolean IsCCMSilentMonitor()
```

.NET

```
bool IsCCMSilentMonitor()
```

Parameters

None.

Return Values

If Unified Communication Manager based silent monitor is configured, this method returns true, otherwise it returns false.

IsSupervisor

The IsSupervisor method checks if the current agent is a supervisor.

Syntax

C++

```
bool IsSupervisor()
```

COM

```
HRESULT IsSupervisor (VARIANT_BOOL * bIsSupervisor)
```

VB

```
IsSupervisor () As Boolean
```

Java

```
boolean IsSupervisorMode()
```

.NET

```
bool IsSupervisorMode()
```

Parameters

bIsSupervisor

Output parameter (return parameter in VB) that returns true if the current AgentMode connection is for a supervisor and false if it is for an agent.

Return Values

If the current agent is a supervisor it returns true, otherwise it returns false.

IsValid

For more information about the IsValid method, see CtiOs Object.

RemoveEventListener (Java and .NET)

The RemoveEventListener method unsubscribes a Java IGenericEvents object as a listener from a particular subscriber list.

Syntax

```
int RemoveEventListener(IGenericEvents Listener, int iListID)
```

Parameters

Listener

The IGenericEvents object that is unsubscribing from events.

ListID

The ID of the subscriber list from which the Listener is to be removed.

Returns

A CtiOs_Enums.CilError code indicating success or failure.

RemoveListener Methods (C++ Only)

The RemoveListener methods unregisters the subscriber from a specified event listener.

Syntax

```
int RemoveEventListener(Arguments & rArguments);
int RemoveSessionEventListener(ISessionEvents * pSessionEvents);
int RemoveCallEventListener(ICallEvents * pCallEvents);
int RemoveAgentEventListener(IAgentEvents * pAgentEvents);
int RemoveSkillGroupEventListener(ISkillGroupEvents *
pSkillGroupEvents);
int RemoveButtonEnablementEventListener(IButtonEnablementEvents *
pButtonEvents);
int RemoveAllInOneEventListener(IAllInOne * pAllInOneEvents);
int RemoveSilentMonitorEventListener(ISilentMonitorEvents *
pSilentMonitorEvents);
int RemoveSessionEventGenericListener(IGenericEvents *
pSessionEvents);
int RemoveCallEventGenericListener(IGenericEvents * pCallEvents);
int RemoveAgentEventGenericListener(IGenericEvents * pAgentEvents);
int RemoveSkillGroupEventGenericListener(IGenericEvents *
pSkillGroupEvents);
int RemoveButtonEnablementEventGenericListener(IGenericEvents *
pButtonEvents);
int RemoveAllInOneEventGenericListener(IGenericEvents *
pAllInOneEvents);
int RemoveSilentMonitorEventGenericListener(IGenericEvents *
pSilentMonitorEvents);
```

Remarks

For more information, see Subscribe for Events for in C++ in Building Your Custom CTI Application.

RequestDesktopSettings

The RequestDesktopSettings method sends a request to the CTI OS Server to download the configuration settings defined for a desktop application.

Syntax

C++

```
int RequestDesktopSettings(Arguments& args)
```

COM

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

VB

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

Java

```
int RequestDesktopSettings(int desktopType)
```

.NET

```
CilError RequestDesktopSettings(Arguments rArgs)
```

Parameters

args

C++, COM, VB, and .NET: Input parameter in the form of a pointer or reference to an Arguments array containing one number. This number has a keyword of “DesktopType” and an integer value that is either:

- eAgentDesktop (0)
- eSupervisorDesktop (1)

Java: desktopType

0 for agent

1 for supervisor

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

A successful RequestDesktopSettings request results in an OnGlobalSettingsDownloadConf event. For more information about the OnGlobalSettingsDownloadConf event, see OnFailure Event.

SetAgent

The SetAgent method assigns an agent to this Session object. Set the following properties for the Agent object used as a parameter in this method:

- CTIOS_AGENTID
- CTIOS_PERIPHERALID

To sign on a mobile agent, you must set the following parameters:

- CTIOS_AGENTCALLMODE
- CTIOS_AGENTREMOTENUMBER

Syntax

C++

```
int SetAgent(CAgent& agent)
```

COM

```
HRESULT SetAgent(/*[in]*/IAgent *agent, /*[out, retval]*/ int * errorcode)
```

VB

```
SetAgent (agent As CTIOSCLIENTLib.IAgent) As Long
```

Java

```
int SetAgent(Agent agentObject)
```

.NET

```
CilError SetAgent(Agent NewAgent)
```

Parameters

agent

The agent to be assigned to the Session object.

errorcode

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

Return Values

If the SetAgent request is successful, it returns a CIL_OK CtiOs_Enums.CilError code and sends an OnSetAgentMode event to the client application.

In CTI OS Release 7.1(1) , the SetAgent request returns the following error codes:

- CIL_FAIL – The request to authenticate failed. The SetAgent request is not sent.
- E_CTIOS_SET_AGENT_SESSION_DISCONNECT_REQUIRED – You attempted to execute SetAgent for a session in monitor mode. The SetAgent request is not sent. To correct, execute the Disconnect method to disconnect the session, then execute the SetAgent method.
- E_CTIOS_AGENT_ALREADY_IN_SESSION – You attempted to assign an agent that has already been assigned to this session. The SetAgent request is not sent.

Note In the above error cases, the SetAgent request is not sent to the CTI OS server, and the client application does not receive any events in return.

- CIL_OK – The SetAgent request was sent to the CTI OS server.

In Java only, if SetAgent () is called on a session where the current agent is different from the agent that SetAgent is trying to set, the following occurs:

- The CIL automatically does a Disconnect on the current session object to Reset an agent.
- An OnCloseConnection event is received.
- A Connect is then performed.
- An OnConnection event is received, and the new agent is set.

In Java only, if the SetAgent request is unsuccessful it returns one of the following CtiOs_Enums.CilError codes:

- E_CTIOS_INVALID_SESSION -- if session is not connected.
- E_CTIOS_PROP_ATTRIBUTES_ACCESS_FAILED -- if unable to get the connection mode property
- E_CTIOS_SET_AGENT_SESSION_DISCONNECT_REQUIRED -- if SetAgent request was during a Monitor Mode session. The client application calls Disconnect first to clean up the connection mode and then calls Connect again.
- E_CTIOS_AGENT_ALREADY_IN_SESSION -- if the agent is already assigned to the session object. The client application calls Disconnect first to clean up the connection mode and then calls Connect again.
- E_CTIOS_ARGUMENT_ALLOCATION_FAILED -- if the application is unable to allocate memory.
- E_CTIOS_PROP_ATTRIBUTES_ACCESS_FAILED -- if an error occurred while accessing agent properties.

SetCurrentCall

The SetCurrentCall method assigns a call as the session's current call.

Syntax

C++

```
int SetCurrentCall(CCall& call)
```

COM

```
HRESULT SetCurrentCall (/*{in]*/ICall *call, /*[out, retval]*/ errorcode
```

VB

```
SetCurrentCall (call As CTIOSCLIENTLib.ICall)
```

Java

```
int SetCurrentCall(Call callObject)
```

.NET

```
CilError SetCurrentCall(Call rCall)
```

Parameters

call

Call to assign as current call.

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

A successful request results in an OnCurrentCallChanged event.

In Java and .NET, if the Call object specified in the call parameter is already the current call, the OnCurrentCallChanged event is not fired to the client application and a E_CTIOS_CALL_ALREADY_CURRENT_IN_SESSION code is returned.

SetCurrentSilentMonitor

The SetCurrentSilentMonitor method sets the SilentMonitorManager object instance specified as the current manager in the CTI OS session object.

Syntax

C++

```
int SetCurrentSilentMonitor(CSilentMonitorManager * pSilentMonitor);
```

COM

```
HRESULT SetCurrentSilentMonitor (/*[in]*/ ISilentMonitorManager * pSilentMonitor,
/*[out,retval]*/ int * errorcode);
```

VB

```
SetCurrentSilentMonitor (ByVal pSilentMonitor As CTIOSCLIENTLIB. ISilentMonitorManager)
As Long
```

Java

```
Not available
```

.NET

```
Not available
```

Parameters

pSilentMonitor

Valid pointer to a SilentMonitorManager object created via CreateSilentMonitorManager.

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

Supported for use with Unified CCE only.

SetMessageFilter

The SetMessageFilter method specifies a filter for the CTI OS Server to use to determine which events are sent to a monitor mode client.

Syntax

C++

```
int SetMessageFilter(string filter)
```

COM

```
HRESULT SetMessageFilter(/*{in]*/ BSTR filter, /*[out, retval]*/ int* errorcode)
```

VB

```
SetMessageFilter (filter As String, retVal As Long)
```

Java

```
int SetMessageFilter(Arguments messageFilter)
```

.NET

```
CilError SetMessageFilter(Arguments rArgs)
```

Parameters

filter

A string containing the message filter, as explained in the section Notes on Message Filters, on page 34.

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 Session receives an OnMonitorModeEstablished event when the filter is set on the server.

SetSupervisorMonitorMode

You can use the SetSupervisorSilentMonitorMode method to force supervisors into monitored mode. It is used, for example, by the CTI OS Agent desktop to indicate that supervisors logging on to the Agent Desktop can be monitored.

Syntax

C++

```
int SetSupervisorSilentMonitorMode (Arguments & args);
```

COM

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

VB

```
SetSupervisorSilentMonitorMode (args As CTIOSCLIENTLib.IArguments);
```

Java/.NET

```
Not available
```

Parameters

args

Arguments array that contains the following parameters.

Table 5: SetSupervisorSilentMonitorMode Arguments Array Parameters

Keyword	Type	Description
CTIOS_SILENTMONITOR FORCEMONITOREDMODE	INT	One of the following values: 1 -- supervisors can be monitored 0 (default) -- supervisors are put in monitoring mode and cannot be monitored

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.

Notes on Message Filters

A message filter is a condition that an event must meet to be sent to the client. It consists of a keyword/value pair, as explained in the following sections.

Note Two filter mode applicationss are allowed for each CTI OS Server.

Message Filter Syntax

The CTI OS Server's event filter mechanism enables the rapid creation of powerful CTI integration applications. The event filter allows the developer to create a custom event notification stream using a simple filter expression. The filter expression is sent from the Client Interface Library (CIL) to the CTI OS server to request an event stream. The CTI OS server's event filter parses the filter expression, and registers the CIL client for all events that match any of the filter's conditions.

To set a filter expression, the Session object's SetMessageFilter() method is used:

```
'put filter expression in hereDim sFilterExpression As String

'call SetMessageFilter
m_session.SetMesageFilter sFilterExpression
```

The general form for a filter expression is key=value.

Simple Example

The most basic event filter is for all events for a specific agent. CTI OS uniquely identifies an Agent object by UniqueObjectID (refer to CIL architecture chapter for explanation of the UniqueObjectID). To establish an event stream for a unique agent, the following syntax is used:

```
sFilterExpression = "UniqueObjectID=agent.5000.22866"
```

In this example, the key is the UniqueObjectID, and the value is agent.5000.22866. This is not the same filter expression created when a CIL client connects to CTI OS in Agent Mode. When a CIL client connects to CTI OS in agent mode, the filter includes events for the agent as well as call events for the agent's extension.

General Form of Filter Syntax

The event filter syntax is expressed in the following general form:

```
key1=value1 [,value2, ...] [; key2=valueA [,valueB, ...] ...]
```

In this form, the filter expression must start with a key name (key). Following the key must be an equal sign (=), and at least one value (value1) must be specified. Optionally, additional values (e.g. value2, ...) for the same key can be specified (optional parts of the expression are indicated with square brackets []). This is interpreted as a logical OR among all of the values specified for that key, e.g. if any of those values is found, the condition is satisfied.

For example, a filter expression with one key and multiple values might look like the following:

```
sFilterExpression = "AgentID=22866, 22867, 22868"
```

The interpretation of this filter is to match on any event with AgentID of 22866, 22867, or 22868.

Multiple Filters

You can combine multiple filters expressions (as described above) to create more complex expressions. The general form allows for any number of filters to be concatenated using the semicolon (;), which produces a logical AND effect.

For example, a filter expression with multiple keys, each with multiple values might look like the following:

```
sFilterExpression = “AgentID=22866, 22867, 22868; SkillGroupNumber=20, 21”
```

The interpretation of this filter is to match on any event with AgentID of 22866, 22867, or 22868 and with SkillGroupNumber of 20 or 21.

Filters for Specific Events

One of the most powerful types of event filters for custom applications are filters that work on specific events.

An example of such an application is an “all agents” real time display, listing the agent states of all known agents at the call center, using the eAgentStateEvent to receive agent updates. To request specific events, use the MessageID keyword, and the numeric value for the event ID that you wish to receive:

```
' register for all eAgentStateEventssFilterExpression = “MessageID = 30”
```

You can also obtain multiple specific events. For example, consider an all calls real-time display application, using eCallBeginEvent and eCallEndEvent to add or remove calls from a view:

```
' register for all eCallBeginEvents, eCallEndEventssFilterExpression =
“MessageID = 23, 24”
```

Events Not Allowed in Filter Expressions

You cannot use the following events in filter expressions:

- ePreLogoutEvent
- ePostLogoutEvent
- eOnConnection
- eOnConnectionClosed
- eOnConnectionFailure
- eOnHeartbeat
- eOnMissingHeartbeat
- eOnCurrentCallChanged
- eOnCurrentAgentReset
- Events that are part of the IMonitoredAgentEvents interface or the IMonitoredCallsInterface. This includes the following events:
  - eOnMonitoredAgentStateChange
  - OnMonitoredAgentInfoEvent
  - OnMonitoredCallDeliveredEvent
  - OnMonitoredCallEstablishedEvent
  - OnMonitoredCallHeldEvent
  - OnMonitoredCallRetrievedEvent
  - OnMonitoredCallClearedEvent
  - OnMonitoredCallConnectionClearedEvent
  - OnMonitoredCallOriginatedEvent
  - OnMonitoredCallFailedEvent
  - OnMonitoredCallConferencedEvent
  - OnMonitoredCallTransferredEvent
  - OnMonitoredCallDivertedEvent

- OnMonitoredCallServiceInitiatedEvent
- OnMonitoredCallQueuedEvent
- OnMonitoredCallTranslationRouteEvent
- OnMonitoredCallBeginEvent
- OnMonitoredCallEndEvent
- OnMonitoredCallDataUpdateEvent
- OnMonitoredCallReachedNetworkEvent
- OnMonitoredCallDequeuedEvent
- OnMonitoredAgentPrecallEvent
- OnMonitoredAgentPrecallAbortEvent

To circumvent this restriction, use an equivalent message in the filter expression (for example, OnAgentStateEvent instead of OnMonitoredAgentStateChange) and check in the message handler for the CTIOS_MONITORED parameter to be TRUE.

```
void CMyEventSink::OnAgentStateEvent(Arguments & argParams)
{

     if (argParams.IsValid(CTIOS_MONITORED) &&
argParams.GetValueBoolean(CTIOS_MONITORED))

    {

        //Do process the event

    }

}
```

Keyword	Description
FilterTarget=SkillGroupStats (for more information, see Skill Group Statistics, on page 37)	When set, this filter indicates that the CTI OS server forward skill group statistics to the client application, whether or not any agents are logged in.
HideSilentMonitorCallEvents (for more information, see CCM-Based Silent Monitor Calls, on page 38)	This keyword is used to block call events for silent monitor calls in monitor mode applications.

Skill Group Statistics

One of the most common applications for a filter mode application is the processing of only skill group statistics. For this purpose, the specialized filter FilterTarget=SkillGroupStats is defined. When set, this filter indicates that the CTI OS server forward skill group statistics to the client application, whether or not any agents are logged in.

After the filter is set, the client application needs to invoke the EnableSkillGroupStatistics(...) method for each skill group that it is expecting to receive statistics. To stop receiving statistics for a given skill group, the application must invoke the DisableSkillGroupStatistics method.

```
'register to receive skill group
statisticssFilterExpression="FilterTarget=SkillGroupStats"
'call SetMessageFilter
m_session.SetMessageFilter sFilterExpression
'Enable statistics for skills 78,89 and 90 in peripheral 5004
Private Sub m_Session_OnMonitorModeEstablished(ByVal pArguments As
Arguments)
Dim m_Args = new  Arguments
   'For Skill 78
    m_Args.AddItem "SkillGroupNumber",78
    m_Args.AddItem "PeripheralID",5004
    m_session.EnableSkillGroupStatistics m_Args
   'For Skill 89
    m_Args.Clear
    m_Args.AddItem "SkillGroupNumber",89
    m_Args.AddItem "PeripheralID",5004
    m_session.EnableSkillGroupStatistics m_Args
    'For Skill 90
    m_Args.Clear
    m_Args.AddItem "SkillGroupNumber",90
    m_Args.AddItem "PeripheralID",5004
    m_session.EnableSkillGroupStatistics m_Args
   'Don't need arguments any more
   Set m_Arg = Nothing
End Sub
Private Sub MyObjectClass_OnCleanupApplication()
    Dim m_Args = new  Arguments
   'For Skill 78
    m_Args.AddItem "SkillGroupNumber",78
    m_Args.AddItem "PeripheralID",5004
    m_session.DisableSkillGroupStatistics m_Args
   'For Skill 89
    m_Args.Clear
    m_Args.AddItem "SkillGroupNumber",89
    m_Args.AddItem "PeripheralID",5004
    m_session.DisableSkillGroupStatistics m_Args
    'For Skill 90
    m_Args.Clear
    m_Args.AddItem "SkillGroupNumber",90
    m_Args.AddItem "PeripheralID",5004
    m_session.DisableSkillGroupStatistics m_Args
   'Don't need arguments any more
   Set m_Arg = Nothing
End Sub
```

CCM-Based Silent Monitor Calls

If a monitor mode application does not wish to receive events for silent monitor calls, it can include the "HideSilentMonitorCalls" keyword in the filter given to CtiOsSession.SetMessageFilter() to tell CTI OS Server to hide events for silent monitor calls. For more information about how to use this filter, see All Calls Sample.NET .