CTISim Utility Program

This chapter describes the CTISim utility program. It includes:

- An overview of CTISim
- Instructions on how to use CTISim
- Limitations of CTISim
- Descriptions of the messages/events supported by CTISim

CTISim Overview

CTISim is a simulator that can be used in place of the CTI Server to begin your development before you have access to a CTI Server. CTISim is limited in some ways but should enable you to get a head start on development.

CTISim works like CTI Server in many respects:

- You connect to CTISim in the same way that you connect to CTI Server.
- You use the CTI Server message set to communicate with CTISim.
- An agent workstation (desktop) application developed for CTI Server will generally work with CTISim.
- An agent workstation (desktop) application developed using CTISim will work with CTI Server, barring switch-specific dependencies. Since CTISim does not involve a switch, it does not reflect switch-specific dependencies.
- CTISim can interact with CTITest; that is, you can use CTITest, in place of a CTI client application, to interact with CTISim.

Note CTISim does not work with CTI Bridge (All Events) applications.

CTISim is not meant to exactly reproduce the stream of events that you would get in any one environment. In real life, this stream of events depends upon the switch, agent configuration, and other things.

The CTISim program can be found in the tools directory of the Cisco CTI CD.

How to Use CTISim

You can run CTISim on the machine where you are doing your development. In such a case, when you are asked—by your CTI client—to provide SideAHost/SideBHost and SideAPort/SideBPort, use localhost and port 199, respectively. (See the "Sample Output from DispCTI" section on page 5-14 for an example of such a configuration setting using CTITest.) CTISim will be listening on port 199. For the Peripheral ID, use 1.

To start CTISim, simply double-click on CTISim.exe. CTISim is then ready to send and receive messages through port 199.

One approach to developing with CTISim is to use two instances of CTITest.exe (described in Chapter 5, "CTITest Utility Program") to control two devices/instruments and have the application you are developing control the third device. CTITest should be configured to use CTISim instead of the real CTI Server. You can use CTITest for Agent 1 (agent id 1) to make a call to Agent 2 (agent id 2) by performing a make_call with a dialed number of 1 (dial the agent id of the agent you want to receive the call).

CTISim Limitations

The CTISim utility has the following limitations:

- Supports CTI Server protocols 3, 5, and 6 only.
- All masks sent in with OPEN_REQ are ignored. CTISim assumes a client_events service with all messages and agent states.
- MonitorIDs are always 0.

- Heartbeat is not supported.
- PeripheralID is always the same as that sent to OPEN_REQ.
- PeripheralType is always PT_ASPECT.
- LineHandle is always 0.
- LineType is always 1.
- ServiceNumber and ServiceID are always set to NULL_SERVICE.
- Little or no validation is done on messages.
- Single-Step (Blind) Conference or Transfer is not supported.
- The following requests are not currently supported:

    ALTERNATE_CALL_REQ

    CLIENT_EVENT_REPORT_REQ

    DEFLECT_CALL_REQ

    MAKE_PREDICTIVE_CALL_REQ

    QUERY_AGENT_STATE_REQ

    QUERY_DEVICE_INFO_REQ

    QUERY_SKILL_GROUP_STATISTICS_REQ

    RECONNECT_CALL_REQ

    SEND_DTMF_SIGNAL_REQ

    SNAPSHOT_CALL_REQ

    USER_MESSAGE_REQ
- The following confirmations/events are not currently sent:

    AGENT_PRE_CALL_ABORT_EVENT

    AGENT_PRE_CALL_EVENT

    ALTERNATE_CALL_CONF

    CALL_CONNECTION_CLEARED_EVENT

    CALL_DEQUEUED_EVENT

    CALL_DIVERTED_EVENT

    CALL_FAILED_EVENT

    CALL_ORIGINATED_EVENT

CALL_QUEUED_EVENT

CALL_REACHED_NETWORK_EVENT

CALL_SERVICE_INITIATED_EVENT

CLIENT_EVENT_REPORT_CONF

DEFLECT_CALL_CONF

MAKE_PREDICTIVE_CALL_CONF

QUERY_AGENT_STATE_CONF

QUERY_DEVICE_INFO_CONF

QUERY_SKILL_GROUP_STATISTICS_CONF

RECONNECT_CALL_CONF

SEND_DTMF_SIGNAL_CONF

SNAPSHOT_CALL_CONF

SYSTEM_EVENT

USER_MESSAGE_CONF

USER_MESSAGE_EVENT

Message/Event Support

All successful message requests (REQ) will trigger the appropriate confirmation (CONF) to be sent back to the application making the request. The CONF will always contain the same InvokeID as the request.

AGENT_STATE_EVENT

Indicates that a change in agent state has occurred.

- PeripheralID = 1
- PeripheralType = PT_ASPECT
- SkillGroupState and AgentState are always the same value.
- StateDuration = 0

- SkillGroupNumber = NULL_SKILL_GROUP
- SkillGroupID = NULL_SKILL_GROUP
- SkillGroupPriority = 0
- EventReasonCode = 0

Agent id, ext and instrument from OPEN_REQ are always sent.

ANSWER_CALL_REQ

Used to connect an alerting call at the alerted device.

CTISim does not support the AgentInstrument floating field. It assumes that the agent that issues the ANSWER_CALL_REQ is the one that should answer the call.

CTISim sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION if it cannot find the call described by ConnectionCallID, ConnectionDeviceIDType, or ConnectionDeviceID.

It sends back AGENT_STATE_EVENT of AGENT_STATE_TALKING to tell the requester that he is now talking. It also sends it to the calling party to tell him that he is talking also.

It also sends CALL_ESTABLISHED_EVENT to both parties.

It sends ANSWER_CALL_CONF.

BEGIN_CALL_EVENT

Used to associate a call with the CTI client.

- NumCTIClients = 1
- Named variables and named arrays are supported (protocol 6)
- CallType depends on what was used in method that created this call
- ConnectionDeviceIDType = 0
- ConnectionCallID is unique (callids are incremented throughout the CTISim session)
- ConnectionDeviceID is set to the calling parties extension and is sent

The following floating values are also sent if they are present in the request that created the call:

ANI

CED

CallVar 1 => 10

UUI

DNIS

DN

WrapUp

NamedVars

NamedArrays

CALL_CLEARED_EVENT

Sent when a call is terminated.

- ConnectionDeviceIDType = 0
- CallID from BEGIN_CALL_EVENT
- LocalConnectionState is undefined
- EventCause = CEC_NONE
- ReleasingDeviceID is never sent

CALL_CONFERENCED_EVENT

Sent when calls are joined into a conference call.

The details of this event are described under CONFERENCE_CALL_REQ.

- CallType = CALLTYPE_CONSULT_CONFERENCE
- PrimaryDeviceType = 0
- PrimaryCallID = HeldCall's ConnectionCallID
- NumParties = 1

- SecondaryDeviceType = CONNECTION_ID_STATIC
- SecondaryCallID = ActiveCall's ConnectionCallID
- ControllerDeviceType = DEVID_DEVICE_IDENTIFIER
- AddedPartyDeviceType = DEVID_DEVICE_IDENTIFIER
- PrimaryDeviceID = agent's instrument
- SecondaryDeviceID = agent's instrument
- ControllerDeviceID = agent's instrument
- AddedPartyDeviceID = other party on ActiveCall's instrument
- ConnectedPartyCallID[0] = ActiveCall's ConnectionCallID;
- ConnectedPartyDeviceType[0] = HeldCall's ConnectionDeviceIDType;
- ConnectedPartyDeviceID[0] = HeldCall's ConnectionDeviceID

CALL_DATA_UPDATE_EVENT

Sent when changes to the call context data occur.

Some requests cause this event to be sent (sometimes more than once). The contents of this event are described in more detail in descriptions of the requests that cause it to be sent.

The fields that may be updated during this event are:

- Calltype
- Callid
- Deviceid
- Deviceidtype
- NamedVars
- NamedArrays

Note CTISim does not update ANI, UUI, DNIS, DN, CED, routercallkeyday/id, callvar1->10, wrapup, signature and timestamp.

- NumCTIClients = 1

CALL_DELIVERED_EVENT

Sent when a call arrives at the agent's teleset.

- ConnectionDeviceIDType = 0
- CallID from BEGIN_CALL_EVENT
- LineHandle 0
- LineType 1
- ServiceNumber NULL_SERVICE
- ServiceID NULL_SERVICE
- AlertingDeviceType = DEVID_DEVICE_IDENTIFIER
- CallingDeviceType = DEVID_DEVICE_IDENTIFIER
- CalledDeviceType = DEVID_DEVICE_IDENTIFIER
- LastRedirectDeviceType = DEVID_NONE
- LocalConnectionState = LCS_CONNECT
- EventCause = CEC_NONE

Optional floating values that are always sent:

- AlertingDeviceID (alerting agent's extension)
- CallingDeviceID (this agent's instrument)
- CalledDeviceID (alerting/other agent's instrument)

LastRedirectDeviceID is never sent.

CALL_ESTABLISHED_EVENT

Sent when a call is answered at the agent's teleset.

- ConnectionDeviceIDType = 0
- CallID from BEGIN_CALL_EVENT
- LineHandle 0
- LineType 1
- ServiceNumber NULL_SERVICE

- ServiceID NULL_SERVICE
- AnsweringDeviceType = DEVID_DEVICE_IDENTIFIER
- CallingDeviceType = DEVID_DEVICE_IDENTIFIER
- CalledDeviceType = DEVID_DEVICE_IDENTIFIER
- LastRedirectDeviceType = DEVID_DEVICE_IDENTIFIER
- LocalConnectionState = LCS_CONNECT
- EventCause = CEC_NONE
- SkillGroupNumber = NULL_SKILL_GROUP
- SkillGroupID = NULL_SKILL_GROUP
- SkillGroupPriority = 0

Optional floating values that are always sent:

- AnsweringDeviceID (called agent’s instrument)
- CallingDeviceID (this agent’s instrument)
- CalledDeviceID (called agent’s instrument)

CALL_HELD_EVENT

Sent when a call is placed on hold at the agent’s teleset.

- ConnectionDeviceIDType = 0
- CallID from BEGIN_CALL_EVENT
- LocalConnectionState = LCS_HOLD
- HoldingDeviceType = DEVID_DEVICE_IDENTIFIER
- EventCause = CEC_NONE

HoldingDeviceID is set to the extension of the agent that issued the HOLD_CALL_REQ.

CALL_RETRIEVED_EVENT

Sent when a call is taken off hold.

- ConnectionDeviceIDType = 0
- CallID from BEGIN_CALL_EVENT
- LocalConnectionState = LCS_CONNECT
- RetrievingDeviceType = DEVID_DEVICE_IDENTIFIER
- EventCause = CEC_NONE

RetrievingDeviceID is set to the extension of the agent that issued the RETRIEVE_CALL_REQ.

CALL_TRANSFERRED_EVENT

Sent when a call is transferred to another destination.

The details of this event are described under TRANSFER_CALL_REQ.

- CallType = CALLTYPE_CONSULT_OFFERED
- PrimaryDeviceType = HeldCall's ConnectionDeviceIDType
- PrimaryCallID = HeldCall's ConnectionCallID
- NumParties = 1
- SecondaryDeviceType = CONNECTION_ID_STATIC
- SecondaryCallID = ActiveCall's ConnectionCallID
- TransferringDeviceType = DEVID_DEVICE_IDENTIFIER
- TransferredDeviceType = DEVID_DEVICE_IDENTIFIER
- PrimaryDeviceID = HeldCall's PrimaryDeviceID
- SecondaryDeviceID = other party on active call's instrument
- TransferringDeviceID = agent's instrument
- TransferredDeviceID = other party on active call's instrument
- ConnectedPartyCallID[0] = ActiveCall's ConnectionCallID
- ConnectedPartyDeviceType[0] = HeldCall's ConnectionDeviceIDType

- ConnectedPartyDeviceID[0] = HeldCall's ConnectionDeviceID

CLEAR_CALL_REQ

Used to release all devices from the specified call.

CTISim makes sure that the call that is being cleared exists. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the call exists, CTISim does the following:

- It sends CALL_CLEARED_EVENT to calling party and called party.
- It sends AGENT_STATE_EVENT with AGENT_STATE_AVAILABLE to calling party and called party.
- It sends CLEAR_CALL_CONF.
- It sends END_CALL_EVENT to calling party and called party.

CLEAR_CONNECTION_REQ

Used to release a specific device connection from the specified call.

Treated just like CLEAR_CALL_REQ.

CTISim makes sure that the call that is being cleared exists. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the call exists, CTISim does the following:

- Sends CALL_CLEARED_EVENT to calling party and called party.
- Sends AGENT_STATE_EVENT with AGENT_STATE_AVAILABLE to calling party and called party.
- Sends CLEAR_CONNECTION_CONF.
- Sends END_CALL_EVENT to calling party and called party.

CLOSE_REQ

Used to terminate a communication session.

No validation is done. Sends CLOSE_CONF.

CONFERENCE_CALL_CONF

Confirms successful completion of the CONFERENCE_CALL_REQ.

NewConnectionCallID, NewConnectionDeviceType, and NewConnectionDeviceID are set to the active call's values.

NumParties is set to 1.

- LineHandle 0
- LineType 1

No floating values for connected parties are sent.

CONFERENCE_CALL_REQ

Used to conference an existing held call with another active call.

Single-step (blind) conference is not supported by CTISim.

CTISim makes sure the held and active calls exist. If one or both do not exist, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the calls both exist, CTISim does the following:

- It sends CALL_DATA_UPDATE_EVENT with new calltype of CALLTYPE_CONSULT_CONFERENCE for active call. This is sent to the calling party and the called party on the active call.
- It sends AGENT_STATE_EVENT (AGENT_STATE_TALKING) to the other participant on the active call (not the one who issued the CONFERENCE_CALL_REQ).
- It sends CALL_CONFERENCED_EVENT to the calling party and the called party on the active call.
- It sends CONFERENCE_CALL_CONF with the same invokeid.

CONSULTATION_CALL_REQ

Used to request the combined action of placing an active call on hold and then making a new call.

CTISim makes sure the active call exists. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the call exists, CTISim does the following:

- It creates a new call with CallType based on the CallPlacementType sent in. If CallPlacementType is CPT_UNSPECIFIED, CPT_LINE_CALL or CPT_DIRECT_AGENT, the CallType is set to CALLTYPE_AGENT_INSIDE. Otherwise, the calltype is set to CALLTYPE_ACD_IN.
- It inserts the call's ANI with the instrument of the agent on the other end of the active call.
- It sends a CALL_HELD_EVENT for the active call to the agent making the request and the other agent on the active call.
- It sends AGENT_STATE_EVENT of AGENT_STATE_HOLD to the requesting agent and to the agent on other end of call that was just held.
- It makes sure that the agent being dialed is known. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. Then it sends a BEGIN_CALL_EVENT to both parties (the requester and the party being called).
- It sends a CONSULTATION_CALL_CONF.
- It sends CALL_DELIVERED_EVENT to the agent making the request and the agent receiving the new call.

CONTROL_FAILURE_CONF

Sent if a request is unsuccessful, but only under particular conditions. Those conditions are described in the REQ sections.

END_CALL_EVENT

Sent when the association between a call and the CTI client is dissolved.

All values are the same as those in BEGIN_CALL_EVENT.

HOLD_CALL_REQ

Used to place an existing call on hold.

CTISim makes sure that the call exists. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the call exists, CTISim does the following:

- It sends a CALL_HELD_EVENT to the calling party and the called party.
- It sends HOLD_CALL_CONF.

MAKE_CALL_REQ

Used to initiate a call between two devices.

If CTISim cannot find an agent with an id equal to the dialed number passed in, it sends CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If CTISim can find the called device, it does the following:

- It sends a BEGIN_CALL_EVENT with CallType based on the CallPlacementType in the request. If CallPlacementType is CPT_UNSPECIFIED, the CallType is CALLTYPE_ACD_IN. If CallPlacementType is CPT_LINE_CALL or CPT_DIRECT_AGENT, then the CallType is CALLTYPE_AGENT_INSIDE. Otherwise, the CallType is CALLTYPE_ACD_IN.
- It sets ANI to the instrument of the called party.
- It sends the BEGIN_CALL_EVENT to the calling party and the called party.
- It sends MAKE_CALL_CONF.
- It sends CALL_DELIVERED_EVENT to both parties on the call.

OPEN_REQ

Sent to initialize a communication session.

CTISim ignores IdleTimeout, PeripheralID, ServicesRequested, CallMsgMask, AgentStateMask, clientid, clientpwd, clientsignature.

CTISim does the following:

- It accepts as given the agent extension, id and instrument.
- If protocolversion is not 3, 5 or 6, it sends FAILURE_CONF with a failure code of E_CTI_INVALID_VERSION.
- It sends OPEN_CONF where servicesgranted=servicesrequested, monitorid=0, pgstatus=0, time is set to this machine time, peripheral online=true, PeripheralType is PT_ASPECT and id, ext and inst are as specified in the request.
- If it is the first successful open, it assumes the agent is available. If it is not the first open, agent state is last state known for agent. CTISim sends AGENT_STATE_EVENT accordingly.
- Then, for all calls in existence, where this agent is the calling or the called party, CTISim sends a BEGIN_CALL_EVENT for the call with the current call status.

Only VersionNumber is validated.

QUERY_AGENT_STATISTICS_REQ

Allows a CTI client to obtain the current call handling statistics for the client's agent.

CTISim responds with a QUERY_AGENT_STATISTICS_CONF, in which the statistics are always respectively set to 0, 1, 2, …; that is, the first statistic is set to 0, the second to 1, and so on.

RELEASE_CALL_REQ

Sent to indicate that you are finished with a call and that all call variable and call wrapup data updates have been made.

CTISim responds with a RELEASE_CALL_CONF.

RETRIEVE_CALL_REQ

Used to retrieve an existing held connection.

CTISim makes sure that the call being retrieved exists. If not, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If the call exists, CTISim does the following:

- It sends CALL_RETRIEVED_EVENT to the called party and the calling party on the call.
- It sends RETRIEVE_CALL_CONF.
- It sends AGENT_STATE_EVENT (AGENT_STATE_TALKING) to both parties on the call.

SET_AGENT_STATE_REQ

Used to change an ACD agent's state.

CTISim responds with SET_AGENT_STATE_CONF.

Then it sends an AGENT_STATE_EVENT with the new agent state in AGENT_STATE_REQ.

SET_CALL_DATA_REQ

Used to set one or more call variables and/or call wrapup data.

CTISim does not do any validation. It does send a SET_CALL_DATA_CONF. (A CALL_DATA_UPDATE event should be sent to all concerned parties, but this is not as yet supported.)

SNAPSHOT_DEVICE_REQ

Used to retrieve information on a specified device.

CTISim sends SNAPSHOT_DEVICE_CONF with the same invokeid as the request. And for each call on which the agent calling this function is either the caller or the called, it sends: ConnectionCallID, ConnectionDeviceIDType, ConnectionDeviceID, LocalConnectionState.

TRANSFER_CALL_REQ

Used to transfer a held call to an active call.

Single-step (blind) transfer is not supported by CTISim.

CTISim makes sure that the held and active calls exist. If one or both do not exist, it sends back a CONTROL_FAILURE_CONF with FailureCode of CF_GENERIC_OPERATION. If both calls exist, CTISim does the following:

- It sends CALL_DATA_UPDATE_EVENT to the agent making the request. The event is sent for the active call. It contains the new calltype of CALLTYPE_CONSULT. It also sends this event to the other party on the active call.
- It sends AGENT_STATE_EVENT of AGENT_STATE_TALKING to both parties on the active call.
- It sends CALL_DATA_UPDATE_EVENT for the active call where ConnectionCallID, ConnectionDeviceIDType and ConnectionDeviceID are changed to be those of the held call and the calltype is CALLTYPE_CONSULT_OFFERED. This event is sent to both parties on the active call.
- It sends CALL_TRANSFERRED_EVENT to both parties on the active call.
- It sends the agent making the request AGENT_STATE_EVENT of AGENT_STATE_HOLD.
- It sends END_CALL_EVENT for the active call to the agent making the request.
- It sends TRANSFER_CALL_CONF.
- It sends END_CALL_EVENT for the held call to the agent making the request.
- It sends the agent making the request AGENT_STATE_EVENT of AGENT_STATE_AVAILABLE.