Session Management

Configuring TCPIP Transport Services

TCP/IP transport services are used in CTI client/server communications. From the Windows Socket interface, enable the TCP "linger" option and set it to zero to close TCP connections immediately upon request without waiting for previously transmitted data to be acknowledged. This ensures that communications can be re-established quickly after a failure.

If possible, disable the Nagle transmit delay algorithm of TCP to ensure timely delivery of all data. (Disabling the Nagle algorithm is sometimes referred to as the TCP_NODELAY option.) Disabling this algorithm ensures that messages are always transmitted immediately upon request.

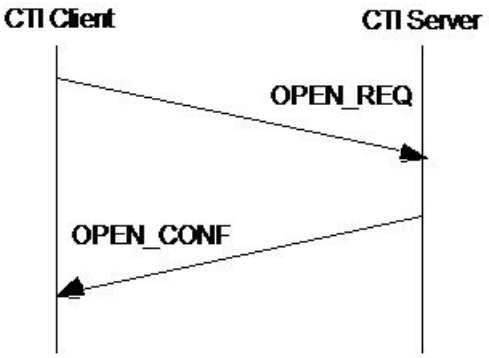
Connection Management

You should configure the CTI clients with two sets of hostname/port number pairs; one for the IP address and TCP port number of the CTI Server on side "A" and the other for the corresponding CTI Server on side "B". The CTI clients should alternately attempt to connect to each side until a connection is established. Once a connection between the CTI client and the CTI Server has been established, the connection remains in place until a failure occurs or the client closes the connection. Connection failures may be detected by the TCP layer or by the heartbeat message mechanism described later in this chapter. If a failure occurs, the CTI client should again alternately attempt to establish a connection to either side until a new connection is established.

Session Initialization

Once a TCP connection has been established, you can attempt to initialize a communications session by sending an OPEN_REQ message to the CTI Server. The CTI Server responds with an OPEN_CONF message to confirm the successful establishment of a session. This figure depicts the message flow.

Figure 1: Session Initialization Message Flow



CTI Service Masks

This table shows the CTIService masks.

Table 1: CTI Service Masks

MaskName	Description	Value
CTI_SERVICE_DEBUG	Causes all messages exchanged during the current session to be captured to a file for later analysis.	0x80000000
CTI_SERVICE_CLIENT_EVENTS	Client receives call and agent state change events associated with a specific ACD phone.	0x00000001
CTI_SERVICE_CALL_DATA_UPDATE	Client may modify call context data.	0x00000002
CTI_SERVICE_CLIENT_CONTROL	Client may control calls and agent states associated with a specific ACD phone.	0x00000004
CTI_SERVICE_CONNECTION_MONITOR	Establishment and termination of this session cause corresponding Unified CCE Alarm events to be generated.	0x00000008
CTI_SERVICE_ALL_EVENTS	Client receives all call and agent state change events (associated with any ACD phone).	0x00000010
CTI_SERVICE_PERIPHERAL_MONITOR	Client may dynamically add and remove devices and calls for which it wishes to receive call and agent state events.	0x00000020

MaskName	Description	Value
CTI_SERVICE_CLIENT_MONITOR	Client receives notification when all other CTI client sessions are opened and closed, and may monitor the activity of other CTI client sessions.	0x00000040
CTI_SERVICE_SUPERVISOR	Client may request supervisor services.	0x00000080
CTI_SERVICE_SERVER	Client identifies itself as server application.	0x00000100
CTI_SERVICE_AGENT_REPORTING	Client may request reporting and routing ARM(Agent Reporting And Management) messages.	0x00000400
CTI_SERVICE_ALL_TASK_EVENTS	Client receives all task events.	0x00000800
CTI_SERVICE_TASK_MONITOR	Client receives monitored task events.	0x00001000
CTI_AGENT_STATE_CONTROL_ONLY	Client can change agent state only. Call control is not allowed. If a client requests for CTI_SERVICE_CLIENT_CONTROL, the server may grant this flag to indicate that only agent state change is allowed.	0x00002000
Unused		0x00004000
CTI_SERVICE_UPDATE_EVENTS	Requests that this client receive update notification events. (No data)	0x00080000
CTI_SERVICE_IGNORE_DUPLICATE_AGENT_EVENTS	Request to suppress duplicate agent state events.	0x00100000
CTI_SERVICE_IGNORE_CONF	Do not send confirmations for third-party requests.	0x00200000
CTI_SERVICE_ACD_LINE_ONLY	Request not to send events for non-ACD lines. (Unified CCE only)	0x00400000
CTI_SERVICE_ACTIVE_STANDBY	When the client opens the session with CTI Server, it informs whether or not it supports the Active and Standby service.	0x02000000

OPEN_REQ Message

This table defines the OPEN_REQ message.

Table 2: OPEN_REQ Message Format

Field Name	Value	Data Type	Byte Size
Fixed Part			
MessageHeader	Standard message header. MessageType = 3.	MHDR	8
InvokeID	An ID for this request message, to be returned in the corresponding confirm message.	UINT	4
VersionNumber	The version number of the interface requested by the CTI client. This defines the version of all messages in the message set.	UINT	4
IdleTimeout	The session idle timer, expressed in seconds. If the session is idle (no messages received) for this time, the CTI Server resets the TCP connection and awaits the establishment of a new session. This value is typically 4 times the heartbeat interval used by the CTI client. If the CTI client does not use the HEARTBEAT_REQ message, set this field to 0xFFFFFFFF.	UINT	4
PeripheralID	The Peripheral ID of the ACD whose events are of interest to the client. Required for Client Events service; otherwise, set this field to 0xFFFFFFFF.	UINT	4
ServicesRequested	A bitwise combination of the CTI Services listed in that the CTI client is requesting.	UINT	4
CallMsgMask	A bitwise combination of the Unsolicited Call Event Message Masks listed that the CTI client wishes to receive.	UINT	4
AgentStateMask	A bitwise combination of Agent State Masks that the CTI client wishes to receive.	UINT	4

Field Name	Value	Data Type	Byte Size
ConfigMsgMask	A bitwise combination of Configuration Event Masks that the CTI client wishes to receive. For bit mask values, see the CONFIG_REQUEST_EVENT message ConfigInformation field. Bit mask indicating what type of information is requested. • 1=Service Information • 2=Skill Group Information • 4=Agent Information • 8=Device Information • 16=Call Type Information • 32=Media Routing Domain Information 256 - Terminal Information	UINT	4
Reserved1	Reserved for future use; set to zero.	UINT	4
Reserved2	Reserved for future use; set to zero.	UINT	4
Reserved3	Reserved for future use; set to zero.	UINT	4
Floating Part			
ClientID (required)	The user ID of the CTI client.	STRING	64
ClientPassword (required)	The password of the user identified by ClientID. ClientID and Client Password are optionally used to authenticate the CTI client making the session open request. This field must be present even if authentication is not being used (it may be of length zero).	UNSPEC	64
ClientSignature (optional)	A character string appended to the Call Client History list when this CTI client becomes associated with a call. If not provided, the ClientID is used.	STRING	64
AgentExtension	The agent's ACD teleset extension. For CLIENT EVENTS service, the CTI Client must provide at least one of AgentExtension, AgentID, or AgentInstrument.	STRING	16
AgentID	The agent's ACD sign-in ID. For CLIENT EVENTS service, the CTI Client must provide at least one of AgentExtension, AgentID, or AgentInstrument.	STRING	12

Field Name	Value	Data Type	Byte Size
AgentInstrument	The agent's ACD instrument number. For CLIENT EVENTS service, the CTI Client must provide at least one of AgentExtension, AgentID, or AgentInstrument.	STRING	64
ApplicationPathID	The ID of an application path which contains configured MRD Peripheral combinations for this Unified CCE-configured application instance.	INT	4
UniqueInstanceID	Optional field. Provided by the client to identify a unique instance of a client. If a response for any request arrives from the OPC at the active CTI Server and the original client request cannot be found using the InvokeID, this field is used to find the requesting CTI Client to send the response to.	STRING	64

Related Topics

CONFIG_REQUEST_EVENT

Unsolicited Call Event Message Masks

This table lists the unsolicited call event message masks.

Table 3: Unsolicited Call Event Message Masks

Mask Name	Description	Value
CALL_DELIVERED_MASK	Set when client wishes to receive CALL_DELIVERED_EVENT messages.	0x00000001
CALL_QUEUED_MASK	Set when client wishes to receive CALL_QUEUED_EVENT messages.	0x00000002
CALL_ESTABLISHED_MASK	Set when client wishes to receive CALL_ESTABLISHED_EVENT messages.	0x00000004
CALL_HELD_MASK	Set when client wishes to receive CALL_HELD_EVENT messages.	0x00000008
CALL_RETRIEVED_MASK	Set when client wishes to receive CALL_RETRIEVED_EVENT messages.	0x00000010
CALL_CLEARED_MASK	Set when client wishes to receive CALL_CLEARED_EVENT messages.	0x00000020
CALL_CONNECTION_CLEARED_MASK	Set when client wishes to receive CALL_CONNECTION_CLEARED_EVENT messages.	0x00000040
CALL_ORIGINATED_MASK	Set when client wishes to receive CALL_ORIGINATED_EVENT messages.	0x00000080

Mask Name	Description	Value
CALL_CONFERENCED_MASK	Set when client wishes to receive CALL_CONFERENCED_EVENT messages.	0x00000100
CALL_TRANSFERRED_MASK	Set when client wishes to receive CALL_TRANSFERRED_EVENT messages.	0x00000200
CALL_DIVERTED_MASK	Set when client wishes to receive CALL_DIVERTED_EVENT messages.	0x00000400
CALL_SERVICE_INITIATED_MASK	Set when client wishes to receive CALL_SERVICE_INITIATED_EVENT messages.	0x00000800
CALL_TRANSLATION_ROUTE_MASK	Set when client wishes to receive CALL_TRANSLATION_ROUTE_EVENT messages.	0x00001000
BEGIN_CALL_MASK	Set when client wishes to receive BEGIN_CALL_EVENT messages.	0x00002000
END_CALL_MASK	Set when client wishes to receive END_CALL_EVENT messages.	0x00004000
CALL_DATA_UPDATE_MASK	Set when client wishes to receive CALL_DATA_UPDATE_EVENT messages.	0x00008000
CALL_FAILED_MASK	Set when client wishes to receive CALL_FAILED_EVENT messages.	0x00010000
CALL_REACHED_NETWORK_MASK	Set when client wishes to receive CALL_REACHED_NETWORK_EVENT messages.	0x00020000
CALL_DEQUEUED_MASK	Set when client wished to receive CALL_DEQUEUED_EVENT messages.	0x00040000
AGENT_PRE_CALL_MASK	Set when client wished to receive AGENT_PRE_CALL_EVENT messages.	0x00080000
AGENT_PRE_CALL_ABORT_MASK	Set when client wished to receive AGENT_PRE_CALL_ABORT_EVENT messages.	0x00100000
RTP_STARTED_MASK	Set when client wished to receive RTP_STARTED_EVENT messages.	0x00200000
RTP_STOPPED_MASK	Set when client wished to receive RTP_STOPPED_MASK_EVENT messages.	0x00400000
AGENT_TEAM_CONFIG_MASK	Set when client wished to receive AGENT_TEAM_CONFIG_MASK_EVENT messages.	0x00800000

Mask Name	Description	Value
AGENT_LEGACY_PRE_CALL_MASK	Set when client wishes to receive AGENT_LEGACY_PRE_CALL_EVENT messages.	0x01000000
CALL_ATTRIBUTE_CHANGE_MASK	CALL_ATTRIBUTE_CHANGE_EVENT messages.	0x02000000
CALL_TERMINATION_MASK	Reserved	0x04000000
CALL_AGENT_GREETING_MASK	Set when client wishes to receive CALL_AGENT_GREETING_EVENT messages.	0x08000000
NETWORK_RECORDING_STARTED_MASK	Set when client wishes to receive NETWORK_RECORDING_STARTED_MASK messages.	0x10000000
NETWORK_RECORDING_ENDED_MASK	Set when client wishes to receive NETWORK_RECORDING_ENDED_MASK messages.	0x20000000
NETWORK_RECORDING_FAILED_MASK	Set when client wishes to receive NETWORK_RECORDING_FAILED_MASK messages.	0x40000000
NETWORK_RECORDING_TARGET_INFO_MASK	Set when client wishes to receive NETWORK_RECORDING_TARGET_INFO_MASK messages.	0x80000000

Agent State Masks

This table lists the agent state masks.

Table 4: Agent State Masks

Mask Name	Description	Value
AGENT_LOGIN_MASK	Set when client wishes to receive “login” AGENT_STATE_EVENT messages.	0x00000001
AGENT_LOGOUT_MASK	Set when client wishes to receive “logout” AGENT_STATE_EVENT messages.	0x00000002
AGENT_NOT_READY_MASK	Set when client wishes to receive “not ready” AGENT_STATE_EVENT messages.	0x00000004
AGENT_AVAILABLE_MASK	Set when client wishes to receive “available” AGENT_STATE_EVENT messages.	0x00000008
AGENT_TALKING_MASK	Set when client wishes to receive “talking” AGENT_STATE_EVENT messages.	0x00000010

Mask Name	Description	Value
AGENT_WORK_NOT_READY_MASK	Set when client wishes to receive “work not ready” AGENT_STATE_EVENT messages.	0x00000020
AGENT_WORK_READY_MASK	Set when client wishes to receive “work ready” AGENT_STATE_EVENT messages.	0x00000040
AGENT_BUSY_OTHER_MASK	Set when client wishes to receive “busy other” AGENT_STATE_EVENT messages.	0x00000080
AGENT_RESERVED_MASK	Set when client wishes to receive “reserved” AGENT_STATE_EVENT messages.	0x00000100
AGENT_HOLD_MASK	Set when client wishes to receive “hold” AGENT_STATE_EVENT messages.	0x00000200
AGENT_ACTIVE_MASK	Set when client wishes to receive “active” AGENT_STATE_EVENT messages.	0x00000400
AGENT_PAUSED_MASK	Set when client wishes to receive “paused” AGENT_STATE_EVENT messages.	0x00000800
AGENT_INTERRUPTED_MASK	Set when client wishes to receive “interrupted” AGENT_STATE_EVENT messages.	0x00001000
AGENT_NOT_ACTIVE_MASK	Set when client wishes to receive “not active” AGENT_STATE_EVENT messages.	0x00002000

OPEN_CONF Message

This table defines the OPEN_CONF message.

Table 5: OPEN_CONF Message Format

Field Name	Value	Data Type	Byte Size
Fixed Part			
MessageHeader	Standard message header. MessageType = 4.	MHDR	8
InvokeID	Set to the value of the InvokeID from the corresponding OPEN_REQ message.	UINT	4
ServicesGranted	A bitwise combination of the CTI Services listed in that the CTI client has been granted. Services granted may be less than those requested.	UINT	4
MonitorID	The identifier of the event monitor created by the OPEN_REQ, or zero if no monitor was created.	UINT	4

Field Name	Value	Data Type	Byte Size
PGStatus	The current operational status of the Peripheral Gateway. Any nonzero indicates a component failure or communication outage that prevents normal CTI operations.	UINT	4
ICMCentral ControllerTime	The current Central Controller date and time.	TIME	4
PeripheralOnline	The current Unified CCE on-line status of the agent's peripheral, when Client Events service has been granted. Otherwise, set this value to TRUE only when all peripherals monitored by the PG are on-line.	BOOL	2
PeripheralType	The value is set as the first condition that applies: **1.** Type of the peripheral that matches with the PeripheralID (if client sends the PeripheralID in the OPEN_REQ) in the OPEN_REQ. **2.** For the ClientEvents service clients, the type of the peripheral to which the agent belongs. (CTI_SERVICE_CLIENT_EVENTS gets the agent information from the OPEN_REQ.) **3.** If none of the above is present, the type of the agent peripheral that is configured in the PG for that CTI Server. **Note** Unified CCE does not support multiple agent peripherals on one PG. For such an unsupported configuration, the PeripheralType that is chosen might be incorrect.	USHORT	2
AgentState	The current state of the associated agent phone (Client Events Service only).	USHORT	2
DepartmentID	Department ID of the Agent	INT	4

Field Name	Value	Data Type	Byte Size
SessionType	Whether the connection/session is Active or Standby.	USHORT	• 0 - Unknown • 1 - Active • 2 - Standby
Floating Part			
AgentExtension (Client Events Service Only)	The agent's ACD device extension, when Client Events service has been granted and the agent is currently signed in on the ACD.	STRING	16
AgentID (Client Events Service Only)	The agent's ACD sign-in ID, when Client Events service has been granted and the agent is currently signed in on the ACD.	STRING	12
AgentInstrument (Client Events Service Only)	The agent's ACD instrument number, when Client Events service has been granted and the agent is currently signed in on the ACD.	STRING	64
NumPeripherals	The number of PeripheralID/info (FltPeripheralID/MultilineAgentControl) pairs specified in the floating part of the message. This field is 0 for non-CCE peripherals, or if PeripheralID is specified in the OPEN_REQ message.	USHORT	2
FltPeripheralID	The peripheralID for the next field (MultilineAgentControl).	UINT	4
MultilineAgentControl	Specifies if multiline agent control is available on the peripheral named in the preceding FltPeripheralID field. 0 = single line only, 1 = multiline enabled.	USHORT	2

If the CTI Server determines that a new session should not be opened, it responds to the OPEN_REQ message with a FAILURE_CONF message. If necessary floating data has not been provided, a FAILURE_CONF message is returned with the status code set to E_CTI_REQUIRED_DATA_MISSING.

A CTI client might try to open a session for Client Events service and the provide device information items that are inconsistent with each other. Then, a FAILURE_CONF message is returned with the status code set to E_CTI_INCONSISTENT_AGENT_DATA. If the ACD device is already associated with a different CTI client, the CTI Server refuses to open the new session and returns a FAILURE_CONF message. The status code in the message is set to E_CTI_DEVICE_IN_USE. If the ACD device is already associated with the same CTI client, the existing session is terminated and the CTI Server opens the new session.

During an OPEN_REQ of an ALL_EVENTS client session, the CTI Server responds with an OPEN_CONF message to confirm the successful establishment of a session. In addition to the OPEN_CONF,

SYSTEM_EVENT messages are sent to the ALL_EVENTS client, per peripheral, to indicate the status of each peripheral associated with the PG.

If the CTI Server rejects an OPEN_REQ message, reset the TCP connection. The status code received in the rejection indicates the message data to correct before retrying to establish a session.

Normally, you receive a response to the OPEN_REQ message within 5 seconds. Some failure scenarios cause all connected CTI clients to lose their connection to the CTI Server. This causes them to then reconnect and reopen their sessions. In the worst case situations, there could be hundreds or even thousands of simultaneous OPEN_REQ messages sent to the CTI Server, causing significant response delays. For this reason, allow at least 30 seconds before considering a lack of response to the OPEN_REQ message as a failure to open the session. In larger configurations of more than 500 clients, allow 60 seconds or more. Then reset the TCP connection, reconnect, and retry the OPEN_REQ after a short delay.

Related Topics

Constants and Status Codes

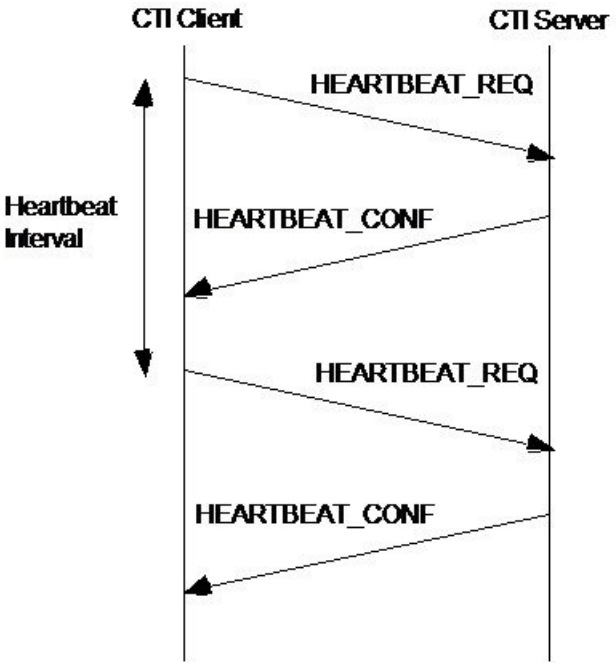
Session Maintenance

Compared to some other protocols, TCP/IP is relatively slow at detecting and recovering from communication path failures. If an IP packet is dropped within the network, retransmission does not occur until the sender notices a time-out. This time-out period is long enough to allow for worst-case round-trip delays and network congestion. If you need more rapid error detection, you may send an optional HEARTBEAT_REQ message to the CTI Server whenever no heartbeat interval messages have been sent. Upon receipt of a HEARTBEAT_REQ message, the CTI Server immediately responds with a HEARTBEAT_CONF message. If three heartbeats go unconfirmed, the CTI client declares a session failure and resets the TCP connection.

You determine the appropriate heartbeat interval for a production environment—It depends on the application and the environment. Find a reasonable balance between the speed of failure detection and the network bandwidth consumed by heartbeat messages and confirmations. In cases with few CTI clients, such as a CTI Bridge, the minimum heartbeat interval of 5 seconds should suffice. Workstation (desktop) clients usually need a larger heartbeat interval (at least 90 seconds), since there are typically hundreds or thousands of clients. A Heartbeat Interval of –1 disables heartbeats. The default setting for application developers is –1. However, if the TCP/IP time-out period is adequate or if the application can do nothing during a failure, you can choose to disable heartbeats in a production environment.

This figure depicts the heartbeat message flow.

Figure 2: Heartbeat Message Flow



This table defines the HEARTBEAT_REQ message:

Table 6: HEARTBEAT_REQ Message Format

Field Name	Value	Data Type	Byte Size
MessageHeader	Standard message header. MessageType = 5.	MHDR	8
InvokeID	An ID for this request message, to be returned in the corresponding confirm message.	UINT	4

This table defines the HEARTBEAT_CONF message:

Table 7: HEARTBEAT_CONF Message Format

Field Name	Value	Data Type	Byte Size
MessageHeader	Standard message header. MessageType = 6.	MHDR	8
InvokeID	Set to the value of the InvokeID from the corresponding HEARTBEAT_REQ message.	UINT	4

The CTI Server does not begin HEARTBEAT_REQ messages. The CTI Server detects failures using the IdleTimeout value from the OPEN_REQ message. If you are using heartbeat messages, the CTI client should set the IdleTimeout value to four times the heartbeat interval. If the CTI Server receives no messages (including HEARTBEAT_REQ messages) for this period, the CTI Server declares a session failure and resets the TCP connection.

The CTI Server may respond to a HEARTBEAT_REQ message with a FAILURE_CONF. This indicates to the CTI client that the CTI Server is off-line, and the CTI client resets the TCP connection.

Session Termination

The CTI client may begin the graceful termination of a communication session by sending a CLOSE_REQ message. The CTI Server responds with a CLOSE_CONF message. Upon receipt of the CLOSE_CONF message, the CTI client can reset the TCP connection. The CTI client should wait up to 5 seconds for the CLOSE_CONF message before resetting the connection.

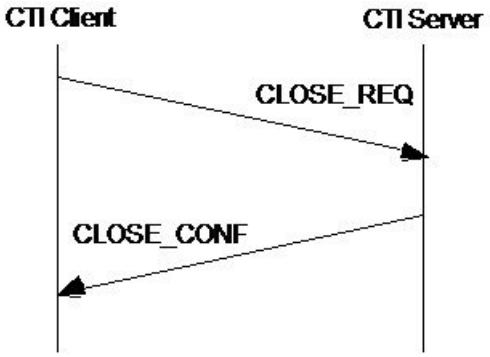
The CTI Server may indicate that it no longer wishes to communicate with the client through an unsolicited FAILURE_EVENT message. The Status field in the message is set to E_CTI_CTI_SERVER_OFFLINE. Upon receipt of this message, the CTI client closes the session.

The CLOSE_REQ message includes a status code that indicates the reason for closing the session. You can set the status code to one of the following:

- E_CTI_NO_ERROR—If the CTI client began the request to end the session.
- E_CTI_CTI_SERVER_OFFLINE—If the CTI Server is no longer online.
- E_CTI_TIMEOUT—If the CTI Server does not respond to a request message within the time-out period.

The following figure describes the Session Termination Message Flow:

Figure 3: Session Termination Message Flow



This table defines the CLOSE_REQ message:

Table 8: CLOSE_REQ Message Format

Field Name	Value	Data Type	Byte Size
MessageHeader	Standard message header. MessageType = 7.	MHDR	8
InvokeID	An ID for this request message, returned in the corresponding confirm message.	UINT	4
Status	A status code indicating the reason for closing the session.	UINT	4

This table defines the CLOSE_CONF message:

Table 9: CLOSE_CONF Message Format

Field Name	Value	Data Type	Byte Size
MessageHeader	Standard message header. MessageType = 8.	MHDR	8
InvokeID	Set to the value of the InvokeID from the corresponding CLOSE_REQ message.	UINT	4

Related Topics

Failure Indication Messages

PG and CTI Server Graceful Shutdown

Graceful shutdown allows administrators to perform firmware upgrades, apply security patches, and apply engineering specials (ES) without the need for a maintenance window. During maintenance mode, the active PG and CTI Server can gracefully hand off processes in progress to their peers, while maintaining call and agent state.

The CTI Servers are deployed in an Active-Standby model. The Agent PG OPC sends the initial configuration to both sides (active and standby). The Agent PG OPC also ensures that there is only one active CTI Server in the system and retries both sides until one side is active.

Both the active and standby CTI Servers accept the client connection. The standby CTI server accepts only clients with CTI Server protocol version 24 or later and connects with a new service mask (CTI_SERVICE_ACTIVE_STANDBY). The standby CTI Server accepts only OPEN_REQ, HEARTBEAT_REQ and CLOSE_REQ messages from clients and responds to those requests with appropriate responses. The server sends a STANDBY_ACTIVE_EVENT_MSG to inform clients that it is changing from standby to active.

When the standby CTI Server restarts, it reloads the agent and call snapshots from the OPC before it processes current or new events. The standby CTI Server receives new events after the current state is loaded.

Note CTI server updates CTI clients with events corresponding to CALL_PARTY_UPDATE_IND (CALL_DELIVERED_EVENT and CALL_ESTABLISHED_EVENT with EventCause 50 (CEC_CALL_PARTY_UPDATE_IND)). These events are sent to CTI_SERVICE_ALL_EVENTS to build the call state properly on the CTI_SERVICE_ALL_EVENTS client after one side of CTI server/PG is gracefully shut down.

STANDBY_ACTIVE_EVENT Message

Standby CTI Server informs the clients when it is changing from Standby to Active.

ACTIVE_MAINTENANCE_REQ Message

This request is sent from Active CTI Server to all the clients who have opened the session with the Service Mask 0x02000000. This request is to affirm if the client is ready to accept the PG going into Maintainence Mode.

This table defines the ACTIVE_MAINTENANCE_RESP message.

Table 10: ACTIVE_MAINTENANCE_RESP Message Format

Field Name	Value	Data Type	Byte Size
InvokeID	An ID for this request message, to be returned in the corresponding confirm message.	UINT	4

ACTIVE_MAINTENANCE_RESP Message

This is a response from clients for ACTIVE_MAINTENANCE_REQ_MSG request. This response indicates whether or not it accepts the PG Maintenance Mode. The CTI Server expects this response with in 5secs from receiving the request. If there is no response received, then it is considered as the negative acknowledgement from the client

Table 11: ACTIVE_MAINTENANCE_RESP Message Format

Field Name	Value	Data Type	Byte Size
InvokeID	An ID for the response that corresponds to the request.	UINT	4
MaintenanceModeAccepted	Client's response for the Maintenance Mode. • 1 - Maintenance Mode Accepted. • 0 - Maintenance Mode Rejected.	BOOL	2

ACTIVE_MAINTENANCE_EVENT Message

This event indicates the final decision of the PG and whether it is going to Maintenance Mode or not. This decision depends on the responses from all the clients to which the ACTIVE_MAINTENANCE_REQ_MSG request message is sent. If any client negatively acknowledged the ACTIVE_MAINTENANCE_REQ_MSG, PG Maintenance Mode will be cancelled. This event is sent from the Active CTI Server.

Table 12: ACTIVE_MAINTENANCE_EVENT Message Format

Field Name	Value	Data Type	Byte Size
MaintenanceModeStatus	Indicates whether the PG and CTI Server continuing with the Maintenance Mode or not. • 1 - MaintenanceModeContinue. • 0 -MaintenanceModeCancel.	BOOL	2

STOPPING_REQUESTS_TO_THIS_SIDE_IND Message

Clients send this message to the CTI Server that went to Maintenance Mode to indicate that it will no longer send any requests to this side. Typically, clients are expected to send this message after it receives the STANDBY_ACTIVE_EVENT_MSG from the Standby CTI Server. Once CTI Server in Maintenance Mode receives this message, it will disconnect the socket. CTI Server expects this message with in 5secs from the time it sent ACTIVE_MAINTENANCE_EVENT_MSG to indicate that it is continuing with the Maintenance Mode