Managing Sessions

This chapter includes the following topics:

How a Client/Server connection is managed

The client uses TCP/IP protocols for network connectivity to the Unified CCX server.

TCP/IP transport services are used in a client/server arrangement. The clients are connected to specific addresses and ports on the server with a hostname/port number pair that is the IP address and the TCP port number of the server. See The Unified CCX CTI server in a high availability Unified CCX System, on page 7 for how to select an IP address and port number.

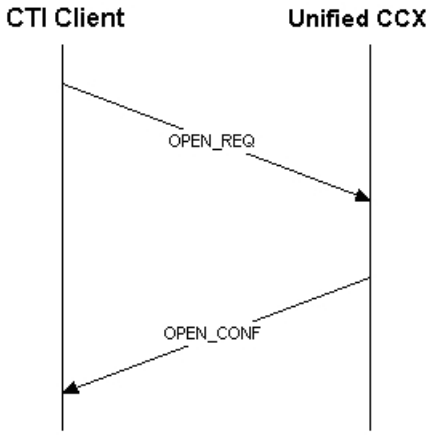
Clients attempt to connect to the server until a connection is established. Once a connection between the client and the server has been established, the connection remains in place until a failure occurs or the client closes the connection. If a failure occurs, the client may again attempt to establish a connection to the server until a new connection is established.

Connection failures may be detected by the TCP layer (see Reasons for a connection to fail, on page 2) or by the heartbeat mechanism (see How a session is maintained, on page 3).

How a session is initialized

Once a TCP connection has been established, a client attempts to initialize a communications session by sending an OPEN_REQ message, defined in OPEN_REQ, to the server. The server responds with an OPEN_CONF message, defined in OPEN_CONF, to confirm the successful establishment of a session. Figure 1: Session Initialization Message Flow, on page 2 depicts the message flow.

Figure 1: Session Initialization Message Flow



If the server rejects an OPEN_REQ message, the client must reset the TCP connection. The status code received in the rejection (see Error (E) status codes) indicates the message data that must be corrected before retrying the attempt to establish a session. It is a good practice to wait for at least 10 seconds before retrying the OPEN_REQ message.

The OPEN_REQ message contains protocol version information. The minimum version number supported by Unified CCX is 10. In general, Unified CCX maintains backward compatibility in new product releases. However, see The Unified CCX CTI Protocol Version Support matrix for complete information.

Reasons for a connection to fail

If for any reason the server determines that a new session must not be opened, it responds to the OPEN_REQ message with a FAILURE_CONF message.

The following are some of the reasons a new session might not be opened:

- If required floating data has not been provided, a FAILURE_CONF message, defined in FAILURE_CONF, is returned with the status code set to E_CTI_REQUIRED_DATA_MISSING.
- If a client attempts to open a session for CLIENT EVENTS service (that is when the client is in Agent Mode — see Two client modes for connecting with Unified CCX, on page 5) and the provided IP phone information items are not consistent with each other, a FAILURE_CONF message is returned with the status code set to E_CTI_INCONSISTENT_AGENT_DATA.
- If the indicated IP phone is already associated with a different client, the server refuses to open the new session and returns a FAILURE_CONF message with the status code set to E_CTI_DEVICE_IN_USE.

- If a connection is made to the non-master server, the server will reply with a FAILURE_CONF indicating error E_CTI_SERVER_NOT_MASTER.

Note That a session does not fail does not mean the client was granted everything requested. The server grants the client only what is available.

How a session is maintained

To detect and handle transmission failures, the client must send a HEARTBEAT_REQ message, see HEARTBEAT_REQ, to the server whenever no messages have been sent for the heartbeat interval or sooner. On receipt of a HEARTBEAT_REQ message, the server immediately responds with a HEARTBEAT_CONF message, defined in HEARTBEAT_CONF. If three heartbeats go unconfirmed, the client must declare a session failure and reset the TCP connection.

The heartbeat interval is solely determined by the client and represents a reasonable balance between the speed of failure detection and the network bandwidth consumed by heartbeat messages and their corresponding confirmations. A maximum value of 30 seconds is accepted by the server

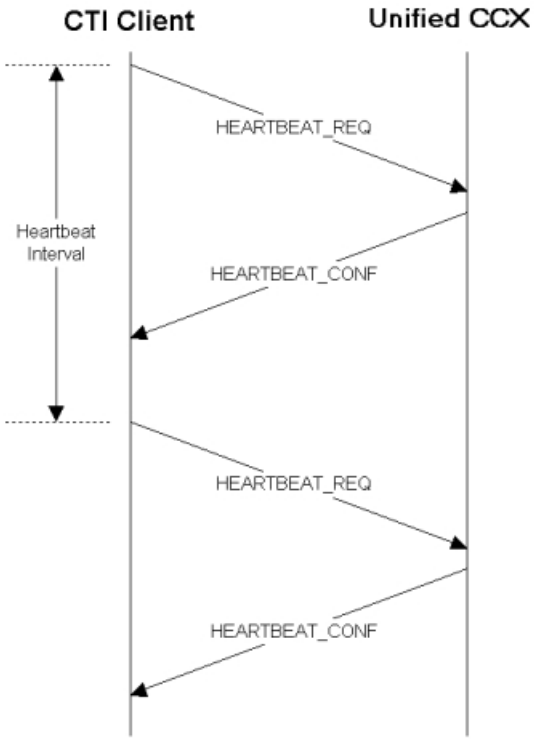
The server does not initiate HEARTBEAT_REQ messages. Failure detection on the server is accomplished using the IdleTimeout value from the OPEN_REQ message. When heartbeat messages are implemented, the client must set this value to four times the heartbeat interval. If the server does not receive any messages (including HEARTBEAT_REQ messages) for this period of time, the server declares a session failure and resets the TCP connection.

The server may respond to a HEARTBEAT_REQ message with a FAILURE_CONF message. This indicates to the client that the server is off-line, and the client must reset the TCP connection.

Note Heartbeat messages must be sent regardless of line activity or in-activity.

Figure 2: Heartbeat Message Flow, on page 4 depicts the heartbeat message flow.

Figure 2: Heartbeat Message Flow



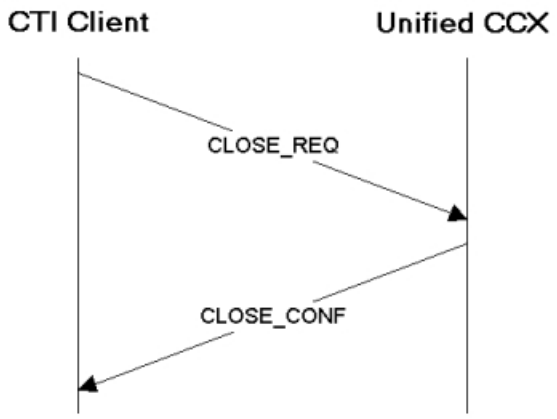
Note You need to synchronize the time of the client and the server and you need to make sure the intervals between the heartbeats are identical.

How a session is ended

The client may initiate the graceful termination of a communication session. The client sends a CLOSE_REQ message, defined in CLOSE_REQ; the server responds with a CLOSE_CONF message, defined in CLOSE_CONF. On receipt of the CLOSE_CONF message, the client resets the TCP connection. The client should wait up to 5 seconds for the CLOSE_CONF message before resetting the connection.

Figure 3: Session Termination Message Flow, on page 5 depicts the session termination message flow.

Figure 3: Session Termination Message Flow



Two client modes for connecting with Unified CCX

Unified CCX provides support for two different ways of connecting to clients:

bridge mode (that is, all events mode)

A bridge mode client receives all agent-state and call events for all logged in agents in the system.

A bridge mode client receives all unsolicited messages and response messages that the client invoked by request messages. The client does not receive response messages invoked by other clients.

agent mode (that is, client mode)

An agent-mode client represents an agent's connection and receives only events that apply to the agent represented by the connection.

An agent mode client receives unsolicited messages that may affect the agent. For example, if an agent device is in a conference call, call events generated by any device in this conference call may be received by this client. In addition, the client receives all response messages that are a result of request messages that the client sent.

You set the mode in the OPEN_REQ message through specifying a mask:

- The CTI_SERVICE_ALL_EVENTS (0x00000010) selection in the Service Masque specifies the client is in bridge mode.
- The CTI_SERVICE_CLIENT_EVENTS (0x00000001) selection in the Service Masque specifies the client is in agent mode. A client connected in the agent mode uses the bit "CTI_SERVICE_CLIENT_EVENTS" in the Service Mask.

These two bits in the mask are mutually exclusive, which means that a client can either be connected in bridge mode or in agent mode.

Note Client connection modes are determined when a session is initialized.

Once you set a mode for a session, you cannot change it during that session. To change the mode, you need to create a new session.

How the client selects the types of messages wanted

Unified CCX is capable of providing much more real-time data than the typical client needs. Message masks have been provided to avoid wasting network bandwidth by suppressing the transmission of unneeded data. Carefully consider the network impact of the expected number of simultaneously connected clients before deploying a client application that unmasks a large number of messages.

Within the OPEN_REQ message, there are four separate mask types for selecting the type of messages wanted and filtering out unwanted messages.

ServicesRequested

A bitwise combination of the Services that the client is requesting.

For example, if a client wants to use CALL_DATA_UPDATE, CLIENT_CONTROL and CLIENT_EVENTS, the Service Mask will be 0x7 which is a bit wise combination of 0x00000002 + 0x00000004 + 0x00000001. See CTI service masks for a list of all the service masks that you can use in an OPEN_REQUEST message.

In the OPEN_CONF message, the ServicesGranted field contains the services that Unified CCX provides to the client. The client must check this field and not assume that services requested are always granted.

CallMsgMask

A bitwise combination of the unsolicited call event message masks that the client wishes to receive.

For example, if a client wants to receive CALL_DELIVERED_EVENT and CALL_ESTABLISHED_EVENT, the CallMsgMask will be 0x5 which is a bit wise combination of 0x00000001 + 0x00000004. See Unsolicited call-event message masks for a list of all the unsolicited call event messages you can use in an OPEN_REQ message.

AgentStateMask

A bitwise combination of agent state masks that the client wishes to receive.

For example, if a client wants to receive all AgentStateEvents, the AgentStateMask will be 0x3FFF which is a bit wise combination of Agent State Mask. See Agent state masks for a list of all the agent state masks that you can use in an OPEN_REQ message.

ConfigMsgMask

A bitwise combination of configuration events that the client wishes to receive.

For example, if a client wants to receive all type configuration update messages, the ConfigMsgMask will be 0x0000001F. See Configuration-Information masks for a list of all the configuration event masks you can use in an OPEN_REQ message.

These masks represent the type of messages the client wishes to receive. Clients may also control what configuration events to receive in the configuration messages.

The masks specified in the OPEN_REQ are used to ensure that clients do not receive unexpected messages. See Masks used in the OPEN_REQ message for full descriptions of each message mask.

Note Whatever services are granted or not granted is reflected in the returned OPEN CONF message.

Reasons for the Unified CCX to send SYSTEM_EVENT messages

Unified CCX sends a SYSTEM_EVENT message (see SYSTEM_EVENT) to all clients to indicate its status when:

- Unified CCX changes its status.
- The agent's device changes its status.

System Event messages are not controlled by bridge or agent mode and are sent to all clients.

The Unified CCX CTI server in a high availability Unified CCX System

High availability is a feature in CRS from release 4.0 onwards, except for the CRS 4.5 release.

A Unified CCX cluster consists of identically configured servers. Only one server is tagged as the master server. The master server makes decisions global to the cluster, like keeping track of calls, agent states, and maintaining socket connections to all CTI clients. At run-time only one of the servers can be the master server at a given time. The standby servers do not process any events on their own, with the possible exception of system events applicable to the local server.

Note While CRS 4.0 and 4.1 support up to 10 servers, CRS 4.5 supports only one server and CRS 5.0 supports 2 servers.

Since a standby server does not manage any CTI clients, it does not receive any CTI events. If an attempt is made to connect to the slave server with an OPEN_REQ message, it responds with a FAILURE_CONF message with a status code of E_CTI_SERVER_NOT_MASTER.

If the master server fails, one of the standby servers becomes the new master server. Since the servers are configured identically, there is no need for a fail back unless the new master server does not have the same capacity as the original one. This would be the case if a different type of hardware is used to install the Unified CCX software in the cluster. Agents are re-connected to the new master server in case of failure of the master server. The TCP port number is specified in the System Parameters web page in the UCCX Administration. The field is *RmCm TCP Port**.

Connect CTI Sessions in a clustered Unified CCX

The following procedure connects a CTI session in a clustered Unified CCX with a given a set of IP addresses and ports which represent the nodes in the cluster:

Procedure

Step 1 Select the first CCX IP address and port.

Step 2 When the CTI client attempts to open a session at the specified IP address and port, there may be any of the following three scenarios:

- Session opens and CTI client receives OPEN_CONF. Go to Step 4, on page 8.
- Socket error / no response.
- Receive FAILURE_CONF.

Step 3 Select next IP address and port. Go to Step 2, on page 8

Step 4 The CTI client should send periodic HEARTBEAT_REQs and should receive messages

Step 5 When the CTI client detects a socket error, or missing HEATBEAT_CONF, or receive a FAILURE_CONF/SYSTEM_EVENT, it should close the current session and go to Step 3, on page 8.

Note CCX will send FAILURE_CONF/SYSTEM_EVENT in various failure scenarios. The HEARTBEAT_CONF is unlikely to be missed, however if CTI client misses the HEARTBEAT_CONF, the CTI client should wait until 2-3 HEARTBEAT_CONFs are missed consecutively.