# CTI Client Application Guidelines



## InvokeIDs

The CTI protocol provides an integer InvokeID field in each request message. This field is always returned in the corresponding response message. You can set the Invoke ID to a unique value for each request you sent to the server. This allows you to have multiple requests outstanding and to correctly determine which request's response has been received. For example, you can implement a simple counter that is incremented with each request.

## Heartbeat Messages

The Heartbeat Interval designates the time in seconds between sending heartbeat messages to the CTI Server. A Heartbeat Interval of -1 disables heartbeats. The default setting for application developers is -1.

You must determine the appropriate heartbeat interval for a production environment -- it depends on the application and the environment. It should represent a reasonable balance between the speed of failure detection and the network bandwidth consumed by heartbeat messages and their corresponding confirmations.

In cases where there are very few CTI clients, such as a CTI Bridge, the minimum heartbeat interval of 5 seconds should suffice. Workstation (desktop) clients should use a much larger heartbeat interval (at least 90 seconds), since these clients typically number into the hundreds and possibly thousands. However, if the TCP/IP time-out period is adequate, or if there is nothing the application can do even if it is aware that something is wrong, it may be appropriate to disable heartbeats even in a production environment.

# Generic vs ACD-Specific Applications

Although CTI Server provides a great deal of call event detail, be aware that the events reported and details provided with each call event vary depending upon the type of ACD involved and possibly the specific software version or other options configured. To remain completely generic and independent of the ACD type, the CTI client should only utilize the BEGIN_CALL_EVENT, END_CALL_EVENT, and CALL_DATA_UPDATE_EVENT messages.

In an object oriented model, you could use the BEGIN_CALL_EVENT message to construct an object that represents this specific call and initializes its contents. Any subsequent call event messages operate on the object and possibly change its state. Finally, you can use the END_CALL_EVENT to trigger any needed cleanup operations and destruction of the call object.

When you need other call event messages to satisfy the application's requirements, try to use as little event data as possible other than the event type (message type). Your application will have fewer ACD specific dependencies.

For a list of the basic differences between ACD types that are potentially visible to a CTI client, see the *CTI OS Developer Guide for Cisco Unified ICM*.

# Message Masks

CTI Server can provide much more real-time data than the typical CTI client needs. The CTI Server provides message masks to suppress the transmission of unneeded data and avoid wasting network bandwidth. You should carefully consider the network impact of the expected number of simultaneously connected CTI clients before deploying a CTI client application that unmasks a large number of messages.

# Message Order

When an event occurs that would conceptually result in two or more event messages being generated at the same time, the CTI client must be prepared to handle the messages arriving in any order. For example, an agent answering an inbound call might generate both a CALL_ESTABLISHED_EVENT and an AGENT_STATE_EVENT message. These may be received by a CTI client in either order, and other event messages may be sent to the client in between. Also, since ACD event data is often obtained from multiple sources, there can be a noticeable delay between event reports that logically occur at the same time.

# Definitions of Symbolic Constants

The symbolic constants shown in tables in this document are available in a C include file, CTILink.h, that is included with every CTI Gateway installation in the \icm\include directory.

# Side AB Selection TCPIP Connections

The following algorithm establishes TCP/IP connections with the CTI Server. This algorithm attempts to strike a balance between rapid reconnection following loss of connection and network saturation (due to

hundreds of clients attempting to connect simultaneously). The algorithm is terminated as soon as a successful TCP/IP connection is achieved:

1. Attempt to connect to the same side as the last successful connection.
2. Attempt to connect to the opposite side.
3. Generate a random integer number N between 0 and the expected number of CTI clients divided by 10.
4. Wait for N seconds. This step helps avoid “rush hour” traffic when all clients at a site are reconnecting simultaneously.
5. Attempt to connect to the same side as the last successful connection.
6. Attempt to connect to the opposite side.
7. Wait for 15 seconds.
8. Attempt to connect to the same side as the last successful connection.
9. Attempt to connect to the opposite side.
10. Wait for 30 seconds.
11. Attempt to connect to the same side as the last successful connection.
12. Attempt to connect to the opposite side.
13. Wait for 60 seconds.
14. Attempt to connect to the same side as the last successful connection.
15. Attempt to connect to the opposite side.
16. Wait for 120 seconds.
17. Repeat steps 14 – 16 until a connection is achieved.

# Alignment of Data Elements

The messages described in this document are sent as a stream of bytes. If the CTI client application uses data structures to represent the messages, be sure that the data structures do not have padding inserted to align elements on particular boundaries, such as aligning 32-bit integers so that they are located on a 4-byte boundary.

# CTI Operations During Unified CCE Failures

The Unified CCE is fault tolerant and recovers from failures quickly, but certain types of failures are not transparent and require consideration during application design:

- A failure of the active CTI Server causes all client connections to be closed. Clients may reconnect immediately (to the other side’s CTI Server in duplex configurations, or to the restarted CTI Server in simplex configurations), but clients will not receive messages for events that occurred while the client session was not open. ClientEvent clients will receive a BEGIN_CALL_EVENT for all calls that are already in progress when their session is opened.

- A failure of the data link or related software between the ACD and the Unified CCE will cause applications not to receive event messages for the duration of the outage. This type of failure is reported to all CTI clients via a SYSTEM_EVENT indicating that the peripheral (ACD) is offline. In addition, the Unified CCE may take additional action depending upon the type of failure and the ACD involved. In many cases, an END_CALL_EVENT will be sent immediately for every call that was in progress, even though the actual voice calls may still be in progress. When normal operation is restored, calls that are in progress may or may not have their call events reported, depending upon the particular type of ACD. If so, a new BEGIN_CALL_EVENT is sent for each call that will have event reporting resumed. In other cases, the calls will be allowed to linger for a short time after the failure without sending an END_CALL_EVENT. If the data link is restored within the short time interval, normal call event reporting continues (although events that occurred during the outage are not reported and the call may now be in a different state). If normal operation is not restored within the allotted time an END_CALL_EVENT is then sent for each call.
- A failure of the datalink between the Unified CCE Peripheral Gateway and the Central Controller does not prevent event messages, however, the failure does prevent use of the Unified CCE post-routing and translation-routing features.