Cscape Call Center Software HE500OSW500

Products Specifications and Installation Data

1 OVERVIEW

The Cscape Call Center performs the following functions:

- Answers incoming calls and performs the necessary communications with the calling controller.
- Requests data from the calling controller
- Transfers data to the calling controller
- Downloads a ladder program to the calling controller
- Uses a database as the interface for controlling transfer operations.
- Uses the standard CsCAN OPC Server version 4.21 or higher as a data interface

2 INSTALLATION

1. Install the standard CsCAN OPC Server version 4.21 or higher
2. Copy CSCANSIM.DLL and SIMCFG.TXT in the same folder as the OPC Server. (\Program Files\Horner APG, LLC\CsCAN OPC Server)
3. Create an ODBC DSN (Data Source Name) that points to the database that contains the operating tables.
   a. Double click Start\Control Panel\ODBC Data Sources
   b. For Windows 95/98 click the User DSN tab / For Windows NT and 2000 click the System DSN tab.
   c. Click ADD and follow the necessary steps to create a DSN for the specific database.
4. Using a text editor (Notepad), open SIMCFG.TXT
   a. Modify DSN= by adding the DSN name used in the previous step. (EX: DSN=CSCAN)
   b. Modify NumPorts= to reflect the number of serial ports that will be used starting at COM1. (Ex: NumPorts=1)
   c. Modify ModemHangupBitLocation= to specify which bit in the calling device will be used to signify a complete transaction. (Ex: ModemHangupBitLocation=R2000.16)
   d. Modify ClockUpdate= to either update the calling device's clock every time it calls in or skip up dating the clock. (EX: ClockUpdate=1 [1=YES 0=NO])

5. Create the necessary operating tables in the database. Standard SQL commands can be used to create the tables or a GUI interface can also be used if one is proved for the database. The following information details the table names, table fields, and field types. The tables were originally generated in a MySQL database.
Care must be taken to create the tables as shown using the same table names, field names, and similar data types.

SQL: CREATE TABLE RegisterList (id INT UNSIGNED, StartRegister VARCHAR(10), NumRegister INT UNSIGNED)

Table Name: RegisterList	
Field	**Field**
Id	int(10) unsigned **auto_increment**
StartRegister	varchar(10)
NumRegisters	int(10) unsigned

SQL: CREATE TABLE SendData (ControllerID INT UNSIGNED, RegisterName VARCHAR(10), RegisterValue INT UNSIGNED, TimeStamp TIMESTAMP)

Table Name: SendData	
Field	**Type**
ControllerID	int(10) unsigned
RegisterName	varchar(10)
RegisterValue	int(10) unsigned
TimeStamp	timestamp(14)

SQL: CREATE TABLE Download (ControllerID INT UNSIGNED, FileLocation VARCHAR(100), TimeStamp TIMESTAMP)

Table Name: Download	
Field	**Type**
ControllerID	int(10) unsigned
FileLocation	varchar(100) or BLOB
TimeStamp	timestamp(14)

SQL: CREATE TABLE Faults (ControllerID INT UNSIGNED, Port VARCHAR(10), Fault INT UNSIGNED, TimeStamp TIMESTAMP)

Table Name: Faults	
Field	**Type**
ControllerID	int(10) unsigned
Port	varchar(10)
Fault	int(10) unsigned
TimeStamp	timestamp(14)

SQL: CREATE TABLE Data (ControllerID INT UNSIGNED, RegisterName VARCHAR(10), RegisterValue INT UNSIGNED, TimeStamp TIMESTAMP)

Table Name: Data	
Field	**Type**
ControllerID	int(10) unsigned
RegisterName	varchar(10)
RegisterValue	int(10) unsigned
TimeStamp	timestamp(14)

3 THEORY OF OPERATION:

Each communication port, COM1 through COMn, is seen to the OPC Server as a virtual OCS node. The virtual OCS nodes correlate to the communication port number where net1_node1 is COM1, net1_node2 is COM2, etc. The number of available virtual OCS nodes is dependent on the value specified for NumPorts in SIMCFG.TXT (See INSTALLATION section).

When the OPC Server starts, the CSCANSIM.DLL creates a virtual OCS node for each communication port. Each virtual OCS node contains a virtual OCS register buffer that the OPC server can read and write. The virtual OCS node is responsible for handling data to and from a given OCS when it calls.

The data to and from the OCS is handled using tables in a database. There are three separate data transactions that can take place when an OCS calls, request register data, send register data, or send a ladder program. The data transactions are handled in the following sequence.

a. Request Register Data:

The virtual OCS node uses the RegisterList table in the database to determine what register data to requested from the calling OCS. The table contains the starting register and the number of consecutive registers to request. The requested OCS data is stored in two locations, the corresponding registers in the virtual OCS register buffer and the Data table in the database. The virtual OCS register buffer contains a snap shot of the data that was retrieved from the last calling OCS and is always overwritten with the next calling OCS data. The Data table contains a history of register values by controller id and time stamp.

b. Send Register Data:

The virtual OCS node uses the SendData table in the database to determine the if register data is to be sent and what register data to send to the calling OCS. The table contains the controller id, register name and register value. If the database contains a matching controller id to the calling OCS, the register value is sent to the corresponding register location. If there are no errors during the data transfer then the item is deleted from the SendData table. If everything transfers successfully then all of the entries for a given controller id will be deleted from the SendData table.

c. Send Program:

The virtual OCS node uses the Download table in the database to determine what ladder program to send to the calling OCS. The table contains the controller id and program location. If the database contains a matching controller id to the calling OCS, the program is loaded and compared to the program information in the calling OCS. If there is a variation, the ladder program is downloaded to the calling OCS. If the programs match then the send program function is skipped.

d. Faults:

If a the fault value changes state, %I1 in the virtual OCS register buffer is updated with the fault value and the fault value, controller id (if one exists) and communication port are logged to the Fault table in the database. The fault value is bit mapped using the following bits.

8	7	6	5	4	3	2	1
CSCAN DLL FAIL	PUT PROGRAM FAIL	PUT DATA FAIL	GET DATA FAIL	GET SERIAL NUMBER FAIL	MODEM ERROR	COM INIT ERROR	DATABASE ERROR

16	15	14	13	12	11	10	9
X	X	X	X	RUN MODE ERROR	CLOCK UPDATE ERROR	FILE VERIFY ERROR	FILE LOAD ERROR

BIT VALUE (IN HEX)	ERROR/FAULT	DESCRIPTION
0x0001	DATABASE_ERROR	Problem initialize database or table/element missing
0x0002	COM_INIT_ERROR	Problem initialize com port
0x0004	MODEM_ERROR	Problem initialize modem
0x0008	GET_SERIALNUM_FAIL	Problem get serial number from OCS
0x0010	GET_DATA_FAIL	Problem get register data from OCS
0x0020	PUT_DATA_FAIL	Problem write register data to OCS
0x0040	PUT_PROGRAM_FAIL	Problem download program (Generic)
0x0080	CSCAN_DLL_FAIL	Problem initialize CSCAN DLL
0x0100	FILE_LOAD_ERROR	Problem loading file for download
0x0200	FILE_VERIFY_ERROR	Problem verifying file size & crc after download
0x0400	CLOCK_UPDATE_ERROR	Problem updating OCS clock
0x0800	RUN_MODE_ERROR	Problem setting OCS to RUN mode

4 TECHNICAL ASSISTANCE

For assistance, contact Technical Support at the following locations.
Please visit our website for manual updates.

North America:
(317) 916-4274
www.heapg.com

Europe:
(+) 353-21-4321-266
www.horner-apg.com