COMMUNICATIONS SERVER REQUIREMENTS

1. Communications server must be capable of running in the background as a daemon and outputting all status and error messages to a logfile.
2. The Communications Server must be capable of communicating simultaneously to as many clients as request a data stream, limited only by the hardware or network platform.
3. At startup, Communications Server will read a plain text configuration file containing the following parameters, and set options accordingly:
    a. CONTROL_PORT
    b. UDP_PORT
    c. MULTICAST_PORT
    d. BROADCAST_PORT
    e. UDP_ENABLE
    f. MULTICAST_ENABLE
    g. BROADCAST_ENABLE
    h. PACKET_INTERVAL (milliseconds)
    i. PRUNE_INTERVAL (milliseconds)
    j. DATABASE_HOST
    k. DATABASE_TABLE
    l. DATABASE_USERNAME
    m. DATABASE_PASSWORD
    n. LOGFILE_PATH
4. The Communications Server will open a connection to DATABASE_TABLE on DATABASE_HOST using DATABASE_USERNAME and DATABASE_PASSWORD or it will exit with an error condition written to LOGFILE_PATH.
5. The Communications Server will open a port on CONTROL_PORT and listen for incoming control messages from the client. Control messages can be implemented via TCP and will be either:
    a. CLIENT_READY
    b. CLIENT_OFFLINE
6. When a CLIENT_READY message is received, the Communications Server must:
    a. Insert a client online message into DATABASE_TABLE with the IP_ADDRESS, CLIENT_NAME (received in control message) of the device that sent the CLIENT_READY message and the currently configured PACKET_INTERVAL (3.h). PACKET_TYPE should be recorded as 7 in the database. The timestamp used for the database should be the server's time.
    b. Begin sending packets at PACKET_INTERVAL on all enabled ports (per the configuration file _ENABLE entries) to the IP address of the source of the CLIENT_READY message.
    c. The format of these packets is as implemented previously in sender.c.
7. The Communications Server will receive ACK messages from the client. These ACK messages sent from the client will include the CLIENT_NAME, CLIENT_TIMESTAMP and

CLIENT_LOCATION. When an ACK message is received, the Communications Server must:

    a. Insert CLIENT_NAME, CLIENT_TIMESTAMP and CLIENT_LOCATION into the DATABASE_TABLE with a PACKET_TYPE of 4.

8. When the Communications Server receives a CLIENT_OFFLINE message from a client on CONTROL_PORT, the Communications server must:
    a. Insert a client offline message into DATABASE_TABLE. This should include the IP_ADDRESS and CLIENT_NAME as was set up in the CLIENT_READY case when communication began. The PACKET_TYPE should be recorded as 8 in the database.
    b. Cease communications with the specific client on all ports.
9. If an ACK message is not received from a client in PRUNE_INTERVAL time, the Communications Server must:
    a. Cease communications with the specific client on all ports.
    b. Initiate the same write to the database as in the CLIENT_OFFLINE case except with a PACKET_TYPE as 9.
10. The Communications Server must remain listening on the CONTROL_PORT and prepared to initiate the processes for additional clients at all times.

GPS CLIENT CHANGES

These are all changes and enhancements to the existing GPS client code. The changes are intended to read vital configuration changes from the database to allow for centralized administration of multiple clients as well as introduce a startup sequence whereby the client identifies itself as online to the server and listens for data. The UDP response of the client to the server is also fleshed out in this revision. All other core functionality of the GPS client as already implemented should remain the same.

1. At startup client must make a connection to database server defined in gpsclient.conf and connect to CONFIGURATION TABLE.
    a. If unable to connect to database server, client should retry 5 times at 30 second intervals.
    b. If after 5 intervals the client is unable to reach the database server, it should exit with an appropriate error written to the local event log.
2. The client should query the CONFIGURATION TABLE for the CLIENT-NAME as configured in the gpsclient.conf
    a. The database server should have a record keyed by CLIENT-NAME which will return configuration parameters.
    b. The client should receive from the database query:
        i. UNICAST_PORT, previously ucast-port in gpsclient.conf
        ii. MULTICAST_PORT, previously mcast-port in gpsclient.conf
        iii. MULTICAST_GROUP, previously mcast-group-addr in gpsclient.conf
        iv. BROADCAST_PORT, previously bcast-port in gpsclient.conf

        v. PACKET_VALIDATION, previously packet-validation in gpsclient.conf, a boolean value indicating whether the client requires packets to be of the designated format to be recorded as received
        vi. LOCATION_WRITE_INTERVAL, the interval in milliseconds which the client should write the packet type 0 location record to the database. This is currently hard-set at 5000 but should use this variable pulled from the database instead.
        vii. SERVER_IP, the IP address of the server which will be sending the unicast messaging to the client
        viii. SERVER_CONTROL_PORT, the port that the server will be listening on for control messages
        ix. SERVER_RETRY_INTERVAL, the number of milliseconds the client should wait without receiving data before resending a CLIENT_READY message to initiate data stream
    c. If the query does not return a record, the client should exit with an error message that the client is not configured correctly in the database.
3. Client should configure itself with the values in 2.b.i through 2.b.v using the same behavior as if the gpsclient.conf data was used.
4. Client should execute logic to self-determine IP address for writing in the database.
5. Client proceeds as before, opening listening ports as configured in 2b, setting up buffer for database writes, and connecting to database table
6. Client should open a TCP connection to SERVER_IP on SERVER_CONTROL_PORT and send a CLIENT_READY message. Client should retry at SERVER_RETRY_INTERVAL until data is received on UNICAST_PORT.
7. At this point, client will begin receiving data on at least the unicast port and potentially the other ports if configured on the server. If at any point the client has not received data from the server for SERVER_RETRY_INTERVAL seconds, client should resend CLIENT_READY message to the SERVER_IP on SERVER_CONTROL_PORT (this allows the client to restart communications if pruned by server). The CLIENT_READY message must contain:
    a. CLIENT_NAME (as configured in gpsclient.conf)
8. When the client receives a message from the server on UNICAST_PORT, it must reply to the server's unicast message with an ACK response. The response must contain:
    a. CLIENT_NAME
    b. CLIENT_LOCATION (from GPS)
    c. CLIENT_TIMESTAMP (from GPS)
9. If at any time, the client stops receiving data from the server for more than SERVER_RETRY_INTERVAL, it should re-determine IP address and re-initiate the CLIENT_READY process every five seconds until data is received again.
10. When the client is shut down gracefully (via shutdown from the OS when running as a daemon), the client should send a CLIENT_OFFLINE message to SERVER_IP on SERVER_CONTROL_PORT which must contain:
    a. CLIENT_NAME (as configured in gpsclient.conf)