BTL HL7 ECG Export Rev-0.2

Usage Notes

Date: 12 August 2025
Author: Densray Designs Private Limited

Note: Please refer to BTL HL7 ECG Export _20250707_R0.1.pdf for basic operation. This document describes the changes in Rev-0.2.

1. User Control Over HL7 MSH Fields and Versioning

Users can explicitly set HL7 MSH segment fields and message version, which affects how export messages are generated and sent.

1.1. Receiving Application and Receiving Facility fields:

If set by the user, these override values from Protocol Extra Data (when present). The user-set values are always used in the MSH segment of exported HL7 messages. When Protocol Extra Data is not available, these fields will be sent as empty by default unless the user has set the values.

Relevant API calls:

```
int btlHl7ExpSetReceivingApplicationStr(BtlHl7Export_t* pHl7Exp, char* pStr);

int btlHl7ExpSetReceivingFacilityStr(BtlHl7Export_t* pHl7Exp, char* pStr);
```

1.2. Sending Application and Sending Facility fields:

If set by the user, these override the default values. BTL_ECG2 is the default string for Sending Application while Sending Facility is an empty string by default.

Relevant API calls:

```
int btlHl7ExpSetSendingApplicationStr(BtlHl7Export_t* pHl7Exp, char* pStr);

int btlHl7ExpSetSendingFacilityStr(BtlHl7Export_t* pHl7Exp, char* pStr);
```

1.3. OBX.3 Field for PDF file transfer:

By default this uses the same string as that used by BTL ConnectIN which is **"PDF^ReportFileData"**. If set by the user, it will override the default value.

Relevant API calls:

int btlHl7ExpSetObx3PdfFileTagStr(BtlHl7Export_t* pHl7Exp, char* pStr);

1.4. HL7 Message Version Field

Case 1: User has NOT set HL7 export version:

- Export **uses the HL7 version from Protocol Extra Data if available**.
- **If Protocol Extra Data is missing,** export defaults to HL7 version 2.2.
    - In this case, minimal HL7 segments are generated from the diagnostic XML (btlXmlNg) for patient identification.
    - **If btlXmlNg is not provided/ inaccessible/ not parsable, then export will fail.**
- Minimum required fields for export include at least one of the following: patient's first name, last name, or patient ID.
  If patient details are missing, the export fails with error code BTLHL7EXP_STATUS_XMLNG_NO_PAT_INFO2

Case 2: User HAS set HL7 export version:

- The export library uses this user-specified HL7 version for all messages.
- Segment generation (MSH, PID, OBX) respects this version, ensuring compliance.
- If Protocol Extra Data is absent, data is fetched and segments are generated from the diagnostic XML (btlXmlNg) as described earlier.

1.5. Logging Callback Support

A flexible logging mechanism was added to allow user-defined handling of debug and status logs.

- **btlHl7ExportPrintf(const char* fmt, ...)** formats logs into an internal buffer and:
  - Sends the log message to a registered callback function if set, or
  - Falls back to printing logs on the console if no callback is registered.
- Type definition for log callback function:

  typedef void (*BtlHl7LogCallback_t)(char* msg);
- Register a custom log callback using:

  void btlHl7RegisterLogCallback(BtlHl7LogCallback_t cb);
- This feature enables applications to capture or redirect logs as needed.

2. Example Program (Updated with Logging and HL7 Field Settings)

Note: See btlHl7EcgExportTestmain.c file in source code release folder for more details.

```
#include "btlHl7EcgExport.h"

#include <stdio.h>

// Custom log callback example

void MyLogCallback(const char* msg) {

    // Redirect log to file or custom handler; here we just prefix and print

    printf("[BTLHL7 LOG] %s", msg);

}
```

```
int main() {
    BtlHl7Export_t gBtlHl7Export;
    char ipAddr[] = "127.0.0.1";
    uint16_t port = 23727;

    // File paths
    char pdfFile[] = "hl7Test_1.pdf";
    char diagXml[] = "d60b513e-d780-4ba0-8e57-a66d2ff8d42b.diagnostics.xml";

    // Protocol extra data (example)
    char extraData[] =
      "<ProtocolExtraData Name=\"HL7\">\n"
      "<Segment id=\"MSH\">\n"
      "<HL7Version>2.3</HL7Version>\n"
      "</Segment>\n"
      "...</ProtocolExtraData>";

    // Register custom log callback
    btlHl7RegisterLogCallback(MyLogCallback);

    // Initialize export with server IP and port
    btlHl7ExportInit(&gBtlHl7Export, ipAddr, port);

    // Set user-defined Receiving Application and Facility (overrides Protocol Extra Data)
```

```
    btlHl7ExpSetReceivingApplication(&gBtlHl7Export, "MyApp");
    btlHl7ExpSetReceivingFacility(&gBtlHl7Export, "MyFacility");

    // Set user-defined HL7 export version (e.g., 2.5)
    btlHl7ExpSetHl7Version(&gBtlHl7Export, "2.5");

    // Start the export asynchronously
    int extraDataLen = strlen(extraData);
    int status = btlHl7ExportPdfReport(&gBtlHl7Export, pdfFile, diagXml, extraData,
extraDataLen);
    if (status != 0) {
        printf("Export start failed: %d\n", status);
        return 1;
    }

    // Poll export status
    while (1) {
        Sleep(2000);
        status = btlHl7GetExportStatus(&gBtlHl7Export);
        if (status == BTLHL7EXP_STATUS_INPROGRESS) {
            printf("Export in progress...\n");
        } else if (status == BTLHL7EXP_STATUS_COMPLETE) {
            printf("Export completed successfully.\n");
            break;
        } else {
            printf("Export failed or aborted with status: %d\n", status);
```

```
            break;
        }
    }

    // Shutdown thread gracefully
    btlHl7ExpShutdown(&gBtlHl7Export);
    while (btlHl7ExpIsThreadRunning(&gBtlHl7Export)) {
        Sleep(500);
    }

    return 0;
}
```

[End of Document]