16 Transmission Data Formats

16.1 Pseudobinary B Data Format

This format is based on GOES Pseudobinary format. It is used when the user selects P*seudobinary B* as the choice for *Tx Format.* The format uses ASCII characters. Three bytes are used for each data value. To correctly decode the measurement, you need to know how many readings of each measurement are included in the transmission. There is no metadata that would describe which measurement is which.

16.1.1 Pseudobinary B for non-Random Txs

If the Pseudobinary B format is used for any transmission type except Random, it will have the fields listed below. Scheduled and Alarm transmissions will use this format.

Field	Pseudobinary B for Scheduled and Alarm Txs
BLOCK IDENTIFIER	BLOCK-IDENTIFIER is always sent as B to indicate the start of a binary data group.
GROUP ID	GROUP-ID is sent as 1 to indicate a scheduled transmission. Random and alarm transmissions send 2. Forced send 3, while retransmissions send 4.
OFFSET	Each record is prefixed with an <OFFSET>, which is a 1-byte binary encoded number indicating the number of minutes ago the most recent data was recorded.
MEASUREMENT DATA	Measurement data collected: This data contains only those measurements set up to be included in the transmission (see *Tx Data Content)*. The data values are 3 byte binary encoded signed numbers allowing a range of: -131072 to +131071. The actual 6-bit binary encoded format is described later. The value transmitted will be value * 10^RightDigits. The string /// will be sent if the data was never measured or was erased. The number of values sent for each measurement is set separately for each measurement in the Scheduled Transmissions, Num values to Tx field. Note that this is different from the 8210 where the num values to Tx was the same for each sensor. As a result, the station sends all the required values for one sensor (most recent first) before proceeding to the next measurement. In the 8210, values were interleaved.
BATTERY VOLTAGE [OPTIONAL]	This is an optional 1-byte binary encoded number representing the battery voltage of the station. This value is appended only if the correct Append Option is selected. If the battery reading is zero, then that is the first transmission since bootup. The range of the Pseudobinary number is -32 to +31 and can be converted to volts by multiplying by 0.234 and adding 10.6 allowing a range of 3.1 to 18.1 volts.

Field	Pseudobinary B for Scheduled and Alarm Txs
STATION NAME [OPTIONAL]	This is a variable length, space delimited field that contains the station name. Space delimited means there is a space before and a space after the station name. If the station name were set to SmallCreek, the transmission would contain " SmallCreek " Note the space before and after SmallCreek. The quotes are NOT included.
TX TIME [OPTIONAL]	The time is a 3 byte, binary encoded, signed number without any right digits. It represents the number of seconds since midnight.
SERIAL NO [OPTIONAL]	A space-delimited, variable length ASCII field containing the serial number. See station name above for details.
TX COUNT [OPTIONAL]	A total of six bytes are used to hold the tx count. The first 3 bytes are the total number of good txs made; the second 3 bytes the number of bad txs. Each of the two values is encoded as a binary 3-byte value with no right digits.

16.1.1.1 *Pseudobinary B Scheduled Example*

Here is a transmission with three active measurements; each one is set to include two readings:

```
B1@@Gt@Gs@Sx@Sr@@i@@iL
```

Decoding the message

Pseudobinary values	Decoded into decimal	Completely decoded	Description
B			Denotes Pseudobinary B format
1			Scheduled transmission
@	0	0 min	Delta time
@Gt	500	5.00	Stage #1
@Gs	499	4.99	Stage #2
@Sx	1272	12.72	Precip #1
@Sr	1266	12.66	Precip #2
@@i	41	41	Temp #1
@@i	41	41	Temp #2
L	12	13.4V	Battery voltage (12*0.234+10.6)

16.2 Pseudobinary C Data Format

This format is based on the pseudobinary B format. It uses slightly more bandwidth than the B format, but it is self-descriptive. It is used when the user selects P*seudobinary C* as the choice for *Tx Format.*

Block identifier	1	BLOCK-IDENTIFIER is always sent as C to indicate that this is the pseudobinary C format.
Group id	1	GROUP-ID can be 1 to indicate a scheduled transmission, 2 meaning an alarm transmission, 3 indicating a forced transmission, and 4 indicating a retransmission.
Measurement Delimiter	1	This byte is always a +, and it is used to denote the start of measurement data.
Measurement Index	1	This is encoded 6 bit binary encoded (see below) number which, when translated, tells the measurement index. The station assigns a measurement index (starting with 1 and ending with 16) to each user setup sensor.
Day	2	This 2 byte encoded 6 bit binary encoded (see below) number represents the Julian day of the year. The day tells when the most recent (first) sensor reading of this measurement was made.

Time	2	This 2 byte encoded 6 bit binary encoded (see below) number is a number of minutes into the day. It tells when the most recent (first) sensor reading of this measurement was made.
Interval	2	This 2 byte encoded 6 bit binary encoded (see below) number tells the measurement interval in minutes or the amount of time between readings of this measurement.
Measurement Data	3 for each sensor reading	Measurement data collected: This data contains only those measurements set up to be included in the transmission (see Tx Data Content). The data values are 3 byte binary encoded signed numbers allowing a range of -131072 to +131071. The actual 6-bit binary encoded format is described later. The value transmitted will be value * 10^RightDigits. The string /// will be sent if the data was never measured or was erased. All the required values for one sensor (most recent first) before proceeding to the next measurement. This format is similar to that used by the Sutron Satlink but different from the 8210. There will be 3 bytes of encoded data for every sensor reading. The number of readings depends on the user setup.
Additional Measurements	Variable	If more than one measurement was set up for transmission, more data will follow. Each measurement setup will have data starting with the Measurement Delimiter and ending with Measurement Data.
Final Delimiter	1	This byte is always . and it is used to denote the end of all measurement data.
Battery voltage	1	This is the battery voltage measured prior to making the transmission. The range of the number will be -32 to +31 and can be converted to volts by multiplying by 0.234 and adding 10.6 allowing a range of 3.1 to 18.1 volts.
Append options	Variable	Please see section 16.1.1 PSEUDOBINARY B FOR NON-RANDOM TXS for details.

16.2.1 Example of Pseudobinary C

To help understand the message below here is a relevant bit of setup:

M1 Right Digits	2
M1 Meas Interval	00:01:00
M1 Tx Data Content	All Logged
M2 Right Digits	1
M2 Meas Interval	00:01:00
M2 Tx Data Content	Last
Tx Time	00:00:30
Tx Interval	00:03:00

Here is the message:

```
C1+ABeHq@A@E|@FG@FM+BBeHq@A@@O.K
```

Decoding the message:

Pseudobinary values		Decoded into decimal	Completely decoded	Description
C			Denotes Pseudobinary C format	
1			Scheduled transmission	
+			Delimiter for next measurement	
A	1	M1	Measurement M1	
Be	165	June 14th	M1 day of the year of the most recent reading. For 2013, it is June 14th.	
Hq	561	09:21AM	M1 minutes into the day of the most recent reading: 9:21AM	
@A	1	1 minute	M1 measurement interval in minutes.	
@E|	380	3.80	M1 most recent sensor reading made at 09:21AM	
@FG	391	3.91	M1 sensor reading made at 09:20AM	
@FM	397	3.97	M1 oldest sensor reading made at 09:19AM	
+			Delimiter for next measurement	
B			Measurement M2	
Be	165	June 14th	M2 day of the year of the most recent reading.	
Hq	561	09:21AM	M2 minutes into the day of the most recent reading.	
@A	1	1 minute	M2 measurement interval in minutes.	
@@O	15	1.5	M2 sensor reading	
.			Delimiter for end of measurement data	
K	11	13.2V	Battery voltage (11*0.234+10.6)	

16.3 Pseudobinary D Data Format

This is another compact data format. It differs from Pseudobinary B in that it has a timestamp at the start of the message. The timestamp indicates when the transmission should have taken place and helps decode when the data was collected. Pseudobinary D is 4 bytes larger than format B.

The timestamp is similar to the one in Pseudobinary C. Pseudobinary D is smaller than Pseudobinary C, and it lacks detailed timestamps that would allow one to completely reconstruct the time the data was collected from the message itself. To correctly use Pseudobinary D, the decoder needs to know the measurement setup used.

The benefit of using Pseudobinary D is being able to correctly decode data regardless of when it was sent or received. This allows stations to re-transmit old data and have it correctly interpreted by the decoder while keeping the message size at a minimum.

Name	Bytes	Description
Block identifier	1	BLOCK-IDENTIFIER is always sent as D to indicate that this is the Pseudobinary D format.
Group id	1	GROUP-ID can be 1 to indicate a scheduled transmission, 2 meaning an alarm transmission, 3 indicating a forced transmission, and 4 indicating a retransmission.
Day	2	This 2 byte encoded 6-bit binary encoded (see below) number represents the Julian day of the year. The day tells when the transmission was originally scheduled to take place.
Time	2	This 2 byte encoded 6-bit binary encoded (see below) number is a number of minutes into the day. It tells when the transmission was originally scheduled to take place.
Measurement Data	3 for each sensor reading	Measurement data collected: This data contains only those measurements set up to be included in the transmission (see Tx Data Content). The data values are 3-byte binary encoded signed numbers allowing a range of -131072 to +131071. The actual 6-bit binary encoded format is described later. The value transmitted will be value * 10^RightDigits. The string /// will be sent if the data was never measured or was erased. All the required values for one sensor (most recent first) before proceeding to the next measurement. This format is similar to that used by the Sutron Satlink but different from the 8210. There will be 3 bytes of encoded data for every sensor reading. The number of readings depends on the user setup.
Additional Measurements	Variable	If more than one measurement was set up for transmission, more data will follow. Each measurement setup will have data starting with the Measurement Delimiter and ending with Measurement Data.
Battery voltage	1	This is the battery voltage measured prior to making the transmission. The range of the number will be -32 to +31 and can be converted to volts by multiplying by 0.234 and adding 10.6 allowing a range of 3.1 to 18.1 volts.
Append options	Variable	Please see section 16.1.1 PSEUDOBINARY B FOR NON-RANDOM TXS for details.

16.3.1 Example of Pseudobinary D

To help understand the message below here is a relevant bit of setup:

M1 Right Digits	2
M1 Meas Interval	00:01:00
M1 Tx Data Content	All Logged
M2 Right Digits	1
M2 Meas Interval	00:05:00
M2 Tx Data Content	Last
Tx Time	00:00:30

Tx Interval	00:05:00

Here is the message:

```
D1D~A8@NI@NH@NG@NF@NE@DGF
```

Decoding the message:

Pseudobinary values	Decoded into decimal	Completely decoded	Description
D			Denotes Pseudobinary D format
4			Retransmission
D~	318	Nov 14th	Day of the year of the most recent reading. For 2014, it is Nov 14th.
A8	120	2:00 AM	Minutes into the day. We can tell that this transmission should have been made at 02:00 on Nov14th and parse the data accordingly.
@NI	905	9.05	Sensor M1 collected at 02:00
@NH	904	9.04	Sensor M1 collected at 01:59
@NG	903	9.03	Sensor M1 collected at 01:58
@NF	902	9.02	Sensor M1 collected at 01:57
@NE	901	9.01	Sensor M1 collected at 01:56
@DG	263	26.3	Sensor M2 collected at 02:00
F	12	12.00V	Battery voltage

16.4 Six Bit Binary Encoded Format

The 6-bit binary format is used to encode numbers into displayable ASCII characters. Fractional numbers cannot be represented, so, for instance, a battery voltage of 13.04 volts set up with 2 right digits will be sent as 1304.

- A 1 byte encoded number can range from -32 to +31.
- A 2 byte encoded number can range from -2048 to +2047
- A 3 byte encoded number can range from -131072 to +131071

Binary encoded numbers are always sent most significant bytes first. The number itself is broken down into 6-bit digits, and each digit is placed in one byte of data. The number 64 (ASCII @) is added to each digit to make it fall within the range of displayable ASCII characters. The only exception is that 127 (ASCII <DEL>) is sent as 63 (ASCII ?)

16.4.1 Example 1: Encoding the Number 10 in 1 Byte

Since 10 will fit in 6-bits, we only have to add 64 which would yield 74. So the number 10 would appear as ASCII 74 or the letter J.

16.4.2 Example 2: Encoding the Number 12345 in 3 Bytes

- First, we have to convert 12345 into binary in 6-bit pieces:

- 12345 (base 10) = 11 000000 111001 (base 2)
- Now we can convert each piece back to base 10:
  - 11 000000 111001 (base 2) = 3, 0, 57
- Finally, we add 64 to each piece and convert to ASCII:
  - 67, 64, 121 = ASCII C@y

16.4.3 Example 3. Encoding the Number -12345 in 3 Bytes

- First we have to convert -12345 into two's complement 18-bit binary: -12345 (base 10) = 111100 111111 000111 (base 2)
- Now we can convert each piece back to base 10: 111100 111111 000111 (base 2) = 60, 63, 7
- Finally, we add 64 to each piece and convert to ASCII (since the second piece is 63 we leave it alone):
  - 124, 63, 71 = ASCII |?G

16.4.3.1 *Example 4. Decoding the 3-byte string @SW:*

This is just like encoding except we follow the steps backward.

- First we convert all the characters to ASCII decimal codes:
  - ASCII @SW = 64, 83, 87
- Now we subtract 64 from each piece and convert to 6-bit binary:
  - 0, 19, 23 = 000000 010011 010111
- Finally, we combine all the bits to form one 18-bit two's complement number and convert to base 10:
  - 000000010011010111 = 1239

16.5 Pseudobinary over SMS

Some bytes that are normally used as a part of Pseudobinary transmissions are not allowed in SMS. When the station sends Pseudobinary data over SMS, those bytes are replaced according to the following table:

[	5B	1
\	5C	2
]	5D	3
^	5E	4
`	60	5
{	7B	6
|	7C	7
}	7D	8
~	7E	9
[	5B	1

16.6 SHEF and SHEFFIX Data Format

SHEF is a format that is commonly used by Sutron's satellite transmitters. It is an ASCII format that is easy to read and contains some self-descriptive information.

The format of the transmission data is:

: <LABEL1> <OFFSET> #<INTERVAL> <DATA1> <DATA1> ... <DATA1)>
: <LABEL2> <OFFSET> #<INTERVAL> <DATA2> <DATA2> ... <DATA2> ...
: <LABEL(N)> <OFFSET> #<INTERVAL> <DATA(N)> <DATA(N)> ... <DATA(N)>

LABEL	This is the *Label* entered as a part of the setup for each measurement. The label can be a SHEF two-character parameter code such as HF for gauge height or PC for cumulative precipitation, or it can be any string you enter. Refer to http://noaasis.noaa.gov/DCS/htmfiles/schefcodes.html for a list of SHEF codes commonly used.
OFFSET	This number indicates how long ago the sensor reading was made. The number is in minutes, and it refers to the most recent data. It is relative to transmission start.
INTERVAL	The interval indicates how often the measurement was made. It corresponds to the setting *Meas Interval.*
DATA	This is data collected and logged by the station through measurements. Only logged data may be transmitted. If *Tx Data Content* is set to *Exclude,* no data from that measurement will be transmitted. Like the binary formats, the SHEF format groups all the related data from one measurement. The data is transmitted in ASCII with sign and decimal point (if needed). If a data value has not yet been recorded (or has been erased) the letter M for missing data will be sent. The most recent data is always sent first. The number of values sent for each measurement is set on a measurement by measurement basis.
APPEND OPTIONS	There are a multitude of optional values that may tacked on at the end of the transmission. Please see **APPEND OPTIONS**.

16.6.1 SHEF Example

Here is a message with three active measurements. Each is set to include two readings. The random buffer contains the string EXT. This string was given by an external device through the RS-232 port. Notice how much longer this message is compared to the earlier binary examples.

```
:HG 3 #15 10.20 10.15 :PC 1 #15 50 49 :TA 0 #15 -22.1 -22.0 :VB 0 12.2
```

In the example above *:HG 3 #15 10.20 10.15* means that the sensor labeled HG read the value 10.20 three minutes prior to the start of the transmission. It read 10.15 18 minutes before the start of the transmission, or 15 minutes before it read 10.20. Here is the complete decode:

:HG	Measurement labled HG
3	Reading is 3 minutes old
#15	15 minute measurement interval
10.20	HG sensor reading (most recent)

10.15	HG sensor reading
:PC	Measurement labled PC
1	Reading is 1 minute old
#15	15 minute measurement interval
50	PC sensor reading (most recent)
49	PC sensor reading
:TA	Measurement labled TA
0	Reading is less than one minute old
#15	15 minute measurement interval
-22.1	TA sensor reading (most recent)
-22.0	TA sensor reading
:VB	Battery voltage label
0	Reading is less than one minute old
12.2	Battery voltage reading

SHEFFIX is a modified version of SHEF where the data is positioned in fixed spacing so that it will line up better when displayed. Each measurement reading is given seven bytes. If a reading uses fewer bytes, it is padded with spaces. SHEIFIX transmissions are larger than SHEF transmissions.

16.6.1.1 *SHEFFIX Example*

:STAGE 0 #2 20.50 20.50 20.50 20.50 :PRECIP 3 #5 12.00 12.01 :TEMP 3 #5 23.5 23.2 :BV 1 #3 14

16.6.1.2 *SHEF Example with Min/Max data included*

When transmitting Min/Max data, the formatter includes a timestamp for each value because the Min/Max data is logged at the time of the occurrence rather than the measurement interval. Note how in the message below the MX and MN only have one value and the timestamp is different for each.

:ITEMP 12 #15 25.9018 26.0289 26.0988 26.2451 **:MX 64 #60 26.686** **:MX 101 #60 27.426** **:MN 14 #60 25.891** **:MN 131 #60 25.334** :AVG 12 #60 26.169 26.368

16.7 Sutron Standard CSV

Logs downloaded from the station will be in the Sutron Standard CSV format. It is possible to transmit data in the CSV format. However, CSV messages are large compared to SHEF and Pseudobinary.

The format was introduced in 2009 and is common to current Sutron products. The general format specification for Sutron Standard CSV format is

mm/dd/yyyy,hh:mm:ss,label,data[,units,qual][,label,data[,units,qual]]

16.7.1 Sutron Standard CSV Example

```
04/02/2012,09:23:45,STAGE,20.50
04/02/2012,09:23:50,STAGE,20.50
04/02/2012,09:23:53,Setup Change
04/02/2012,09:24:00,BV,14
04/02/2012,09:25:00,PRECIP,34.5
04/02/2012,09:25:00,TEMP,23.5
04/02/2012,09:25:00,STAGE,20.54
```

16.8 MIS

MIS format is a verbose ASCII format that uses tags. Each measurement gets its own section in MIS.

The header includes the station name and the measurement label. The body includes a timestamp and a value for each sensor reading.

The example below illustrates the MIS format for one measurement. If more than one measurement were being transmitted, each measurement would have a section like the one below.

```
<STATION>000DOLPHIN</STATION><SENSOR>00PR</SENSOR><DATEFORMAT>YYYY/MM/DD</D
ATEFORMAT>
2017/05/24;111500;82.03
2017/05/24;110000;82.04
2017/05/24;104500;83.11
2017/05/24;103000;83.14
```

In that example

- STATION and SENSOR and DATEFORMAT are tags that are always present.
- 000DOLPHIN is based on the Station Name which was set to DOLPHIN. The 000 are padding to fill out to 10 bytes. If the Station Name were longer than 10 bytes, only the first 10 would be in the STATION tag.
- 00PR is based on the measurement Label, which was set to PR. 00 is padding to fill out to 4 bytes. Up to 4 bytes of the Label can fit into the SENSOR tag.
- YYYY/MM/DD is the user set Date Format setting.
- 111500 is the timestamp in HHMMSS format which is 11:15:00
- 82.03 is the sensor reading. If data is missing, --- is transmitted.

16.9 ASCII Column

The ASCII Column format is another means of encoding transmission data. A message formatted with this method would look like this:

```
<CRLF>
Meas1_Data(1) Meas2_Data(1) ... MeasN_Data(1)<CRLF>
Meas1_Data(2) Meas2_Data(2) ... MeasN_Data(2)<CRLF>
Meas1_Data(n) Meas2_Data(n) ... MeasN_Data(n)<CRLF>
```

- <CRLF> is a carriage return, line feed
- Meas1_Data()..MeasN_Data are the data from the active measurements included in the transmission.
- All the data on one line are for the same time. A space separates the values. The newest data is sent first.
- (n) is the number of values included in the transmission as set by the user.
- The format for any value is ASCII with the right digits set by the user.
- No decimal point is shown if the value is zero.
- Add a minus sign in front of the value if the value is negative.
- There are no leading zeroes.
- If the value is missing, two forward slashes are transmitted instead //.

A typical message will appear as follows:

```
41.79 60.99 12.99 12.43 .20 8.08 7.1 195.2
41.79 60.99 12.99 12.43 1.20 8.08 7.1 195.2
41.79 60.99 12.99 12.43 -2.20 8.08 7.1 195.2
```

Normally the DCP will be set up, so all the sensors are collected at uniform times. If this is not the case, the formatter will still work but will have varying amounts of data on each line (each time) based on whatever sensors are collected at each time. This will produce confusing results if the additional sensors are collected at an interval of the other sensors. The results should be OK if the additional sensors are collected at a time separate time.

16.10 ASCII Sensor

This method of formatting uses four digits for each measurement readings. Measurement readings are multiplied by 100 before being transmitted.

```
<CRLF>
IDIDIDID JJJHHMMSS<CRLF>
Meas1_Data(1) Meas1_Data(2) ... Meas1_Data(n)<CRLF>
Meas2_Data(1) Meas2_Data(2) ... Meas2_Data(n)<CRLF>
MeasN_Data(1) MeasN_Data(2) ... MeasN_Data(n)<CRLF>
optional Battery before transmission<CRLF>
optional Battery under load<CRLF>
```

- <CRLF> is a carriage return, line feed
- IDIDIDID is the eight-byte hexadecimal satellite ID
- JJJ is a three digit Julian Day, which is the number of days of the year. January 1st is Julian Day one, and February 1st is Julian Day 32. This number is always three digits (0 padded)
- HHMMSS are the hours minutes and seconds. This is the time of the scheduled transmission. It is always 6 bytes (0 padded).
- Meas1_Data()..MeasN_Data are the data from the active measurements included in the transmission.
- Measurement data is always four digits followed by a space (0 padded).

- All values are multiplied by a 100.
- No decimal point is used for measurement data.
- Range of transmitted data is 0 to 9999 (numbers out of range are transmitted as 0000 or 9999). This means that the sensor data should be between 0 and 99.99 before formatting.

If the user elects to append battery voltage, one of the two will be appended:

- Two lines of battery voltage information will be added. Both lines take the same format: "12.5<CR><LF>" (three digits of battery voltage and a decimal point). The first line is the battery voltage prior to this transmission. The second line is the battery voltage during the last transmission.

This is an example transmission:

```
010570F2 108183053
0250 0250 0250 0250
2231 2232 2233 2234
12.2
12.1
```

- On the first line 010570F2 represents the hexadecimal satellite ID. 108 is the Julian Day (for the year 2005, that is April 18th). 183053 means 18 hours 30 minutes and 53 seconds.
- The second line (0250 0250 0250 0250) shows four readings of the first sensor. This measurement had four same readings. Before formatting, that reading was 2.50. It was multiplied by 100 and padded with a leading zero.
- The third line (2231 2232 2233 2234) shows four different readings of the second sensor. That sensor's readings were originally 22.31, 22.32, 22.33, and 22.34 respectively. They were multiplied by 100 and padded with a leading zero.
- The fourth line (12.2) is the battery reading in Volts prior to transmission.
- The last line (12.1) is the battery reading in Volts during the last transmission.

16.11 TCP/IP Session

A station equipped with a cell modem will use scheduled transmissions periodically connect to a server and deliver sensor data. See the section 13.2 CELLULAR TELEMETRY and section 11.1.6 TX FORMAT for more details.

Relevant settings include *Protocol, Main Server, Backup Server, Server Port, and Server Password.*

When *Protocol* is set to *Hydromet Cloud,* the station will engage in handshaking with the server. The station will provide the server with information about the transmission that includes the station name, password, reason for the transmission, and error checking information. The server will acknowledge the transmission.

If the server does not acknowledge the transmission, the station will retry the transmission until it succeeds or until it times out.

When *Protocol* is set to *None,* the station will not engage in handshaking with the server.

If you are using a custom server, you will want to set *Protocol* to *None.* Either that or have the custom server acknowledge the transmission.

16.11.1 Hydromet Cloud Protocol

When the station connects to a server, some information is sent to the server before the sensor data is delivered. This information is meant to help the server decide what to do with the sensor data that is en route. The information includes details such as the sensor name and whether the transmission is a result of an alarm event.

After the sensor data is delivered to the server, the server may log into the station and issue commands to check status, change setup, download more data, or any other command line activity.

16.11.2 Course of Events for the Hydromet Cloud Protocol

1. The station connects to main or backup server.
2. The station sends Session Type Code<cr> (see below.)
3. The station sends Station Name<cr>.
4. The station sends Report Type Code<cr> to indicate purpose of connection (see below.)
5. The station sends transmission data (if any) in whatever format the user chose, such as SHEF or pseudobinary.
6. The station sends ETX (0x03) to mark end of data.
7. The station sends a 3 byte pseudobinary encoded CRC16 of previous data and the server password (sometimes called Shared Secret.)
8. The station waits for the server to either acknowledge or to initiate a command line session. The server has 60 seconds to do either one.
    a. If the server does neither, the station will retry the transmission until it succeeds or until it times out.
9. To acknowledge, the server needs to send a single byte: <cr>
    a. Once the server sends the byte, the station will disconnect.
10. To initiate a command line session, the server sends user login command !LOGIN=username,password<cr> and waits for the station to reply (ETX.)
    a. If user login matches, the station enters command-line session.
    b. Server issues pending commands, and the station processes and responds accordingly.

c. When command processing is complete, the server disconnects.

16.11.2.1 *Session Type Code*

Session type defines the processing that occurs on connection. Currently, there is only one session type, 1. This type is defined by the Course of Events above, and details that follow.

16.11.2.2 *Report Type Code*

Report Type code describes the purpose of the transmission:

- 0 = self-timed
- 2 = entering alarm
- 4 = exiting alarm
- 6 = no data, command session only

16.12 Iridium Telemetry Header

When the station sends data over the Iridium satellite network, it uses the header described below.

The header starts with a single byte in the ASCII printable range to make it easy to interpret the content. Most transmissions will just have one header byte.

The header byte identifies the packet type in terms of the content of the packet, e.g., self-timed, self-timed extended, entering alarm, entering alarm extended, etc.

Iridium modem buffers are limited in size. "Extended" types mean multiple packets are required to transmit the entire message. Extended packets have information in the header that allows them to be stitched together easily. The extended packet types include a comma-delimited sub-header to describe the subset of data being sent. The first sub-header differs from all subsequent sub-headers, in that it includes the total size of the data being sent.

The telemetry header may contain the station name if the setting *Tx Station Name* is enabled. Please read below for more details.

16.12.1 Packet Structure

Packets are composed of

- packet-type
- sub-header
- data

The packet-type is a single byte:

Hex Value	ASCII	Description
0x30	0	Self-timed
0x31	1	Self-timed extended
0x32	2	Entering alarm
0x33	3	Entering alarm extended
0x34	4	Exiting alarm
0x35	5	Exiting alarm extended
0x36	6	Command response
0x37	7	Command response extended
0x38	8	Forced transmission
0x39	9	Forced transmission extended
...		Reserved for future use
0x7D	}	User defined
0x7E	~	Look to next byte for meaning
0xFF		Binary data, reserved for future use

The sub-header has the following comma-delimited fields:

- ,id,start-byte,total-bytes,station-name:

packet-type	Numeric ASCII character-defining packet type. See type definitions in table, below
id	Numeric ASCII text defining the message id. Starts at 0. Rolls over after 99.
total-bytes	Numeric ASCII text defining the total number of data bytes to be sent (data only, does not include overhead bytes)
start-byte	Numeric ASCII text defining which byte of total-bytes is the start byte of the current packet. Starts at 0.
station-name	Optional ASCII field that has station name. Formatted as ,N=STATION NAME

16.12.2 Iridium Header Examples

The following examples illustrate how to use the new header.

16.12.2.1 *Example 1*

Message requiring one packet (i.e., non-extended), formatted pseudobinary B interleaved, containing 6 values (42, 69, alternating).

Self-timed	0B1@AAhAktAAhAktAAhAkt
Entering Alarm	2B2@AAhAktAAhAktAAhAkt
Exiting Alarm	4B3@AAhAktAAhAktAAhAkt

16.12.2.2 *Example 2*

The same data in SHEF format:

Self-timed	0:HG 0 #1 42.00 42.00 42.00 :EM 0 #1 69.00 69.00 69.00
Entering Alarm	2:HG 0 #1 42.00 42.00 42.00 :EM 0 #1 69.00 69.00 69.00
Exiting Alarm	4:HG 0 #1 42.00 42.00 42.00 :EM 0 #1 69.00 69.00 69.00

16.12.2.3 *Example 3*

An extended command response where the total size of the command response is 512 bytes (note: total size is of the response itself, and does not include the overhead of the telemetry headers used to convey it). The example uses a message id of 0:

Packet 1	7,0,0,512:bytes 0 thru 319
Packet 2	7,0,320:bytes 320 thru 511

16.12.3 Iridium Tx Station Name

There is an option to include the station name in the Iridium Header. The setting is called *Iridium Tx Station Name,* and it defaults to disabled. That setting is accessible only via the terminal (LinkComm->Tools Menu->Terminal). To enable it, type

Iridium Tx Station Name = On

- After all the headers, system will append ,N=Station Name:
- If the station name were Two Creeks, it would be ,N=Two Creeks:

Standard	With Station Name
0B1@AAhAktAAhAktAAhAkt	0,N=Two Creeks:B1@AAhAktAAhAktAAhAkt
4:HG 0 #1 42.00 42.00 42.00 :EM 0 #1 69.00 69.00 69.00	4,N=Two Creeks::HG 0 #1 42.00 42.00 42.00 :EM 0 #1 69.00 69.00 69.00
7,0,0,512:bytes 0 thru 319	7,0,0,512,N=Two Creeks:bytes 0 thru 319
7,0,320:bytes 320 thru 511	7,0,320:bytes 320 thru 511

For extended packets, only the first message carries the station name. The name field comes after all the other headers.