FLAMP Amateur Multicast Protocol

3.0

Generated by Doxygen 1.8.7

Sat Nov 29 2014 15:37:16

Contents

Chapter 1

FLAMP Amateur Multicast Protocol (AMP-2) - Version 3.0

1.1 Authors - Version History

- Version 1.0.0 - W5ALT, Walt Fair, Jr. (Derived From)
- Version 2.0.0 - W1HKJ, Dave Freese, w1hkj@w1hkj.com
- Version 2.0.1 - W1HKJ, Dave Freese, w1hkj@w1hkj.com, 5 Oct 2012
- Version 3.0.0 - KK5VD, Robert Stiles, kk5vd@yahoo.com, 21 April 2013

1.1.1 Disclaimer

This is an unofficial extension of the original Amp specification which is maintained by the Society for the Preservation of Amateur Radio.

1.1.2 Notice

Protocol Version 3.0 or greater is not backward compatible with the previous versions.

1.2 General description

Multicasting is a non-connected protocol and the receiving stations are totally passive, there is no need for hand shaking and related overhead. The server simply repeatedly sends files, with enough information to allow them to be identified, checked for errors and stored. The clients or receiving stations, likewise, have no need to transmit, but simply listen for any files of interest, receive and saves them and checks for errors. If errors are found, the receiver simply waits until the file is sent again and attempts to retrieve the portion that was bad. No distinction is made between text and binary files. Either may be compressed, encoded using base numeric system encoding, or transmitted in original form. Information regarding compression, encoding or other file manipulation is contained within the data transmission and is not an inherent part of the protocol. In order to transmit files for remote reception, the minimum information needed is the name of the file and it's modification date-time, the method used for uncompressing the file (if compressed), the base encoding system (if used, ie: base-64 ...) and the data contained in the file. The combination of modification date-time and file name is sufficient to make each file transfer uniquely identifiable. Cycle Redundancy Checksum (CRC-16) values for each element of the protocol is used to ensure accuracy of reception. For subsequently storing missing data blocks, a block number is also necessary to indicate the data block sequence. Any amateur radio transmission mode which supports the full ANSI character set may be used as the transport layer. Since the transmission is to be done using commonly available systems, it is useful to further identify the information. Of course, as in all amateur communications, there is also the legal necessity of station identification, too.

1.3 Protocol

The Amateur Multicast Protocol (AMP-2) version 3 consists of a sequence of elements described as follows. Each protocol element consists of a keyword with character count and checksum enclosed in angle brackets ($< >$), followed by {hash} value and an optional block number and the specified data. Other information within the brackets may be included, but will be ignored by the receiving station. The optional text information may be useful for comments and other purposes. For readability, a carriage return (or anything else) may be sent following the data and it should be ignored by the receiver.

1.3.1 Protocol Elements

```
<Keyword [Byte-count Checksum]>[{HASH} | {HASH:Block-number}][data]
```

Note: All example checksums and hash value(s) depicted here may not represent actual crc values.

- **Keyword** - uniquely identifies the use of each Amp block.
- **Byte-count** - The byte count contains the number of bytes in the data block. For file names, descriptions, etc. it is the number of characters comprising the name or character string. For file content blocks, it represents the number of bytes transmitted in the data block, irregardless of how they might be handled by the client (receiver) software.
- **Checksum** - The checksum is used by the receiving station to determine the accuracy of the received data. It is a 16 bit cycle redundancy check (Crc16) value. Crc16 C++ code is found in source file crc16.h associated with the application flamp. The CS value is always transmitted as a 4 character hex string.
- **HASH** - 4 character hexadecimal number that is the Crc16 value associated with a concatenated string which consists of the date-time, colon, filename, compression, base conversion, and block size ie:

1.3.2 Concatenated Crc16 data string

```
20130316010524:ShortMessage2.txt0base12896

Example format:

|20130316010524:ShortMessage2.txt|0|base128|96|
```

```
|     DTS      :        FN       |C|   B   |BS|

DTS = Date/Time Stamp
 FN = File Name
  C = Compression 1=ON,0=OFF
  B = Base Conversion (base64, base128, or base256)
 BS = Block Size, 1 or more characters
  | = Field separator.
```

Where the date-time is the the system date-time stamp associated with the file's last modification.

Developers Note: The goal is to create a hash value based on the transmitted file's configuration. The order of the fields can be varied and the only requirement is they must be included in the CRC16 calculation.

Block-number. The block number is only needed while sending file contents. For transmission, the file is divided into convenient sized blocks (64 bytes is recommended) and each block is numbered starting with 1 for the first block sent, 2 for the second, etc. The use of a block number allows the client software to keep track of what part of the file has been correctly received and which parts are still incorrect, so that the needed blocks can be received during a subsequent transmission.

1.3.3 Protocol blocks

<**PROG # CS**>*{HASH}*Name and version of software used to generate the Amp text stream.

<**ID # CS**>*{HASH}*Callsign other info. Every transmission should begin with the station identification. In addition, at regular intervals (at least every 10 minutes and at the end of a communication in the US), there is the legal requirement for station identification. In order to separate the identification function from other data, the ID keyword should be used. In many cases the receiver is free to ignore this keyword, unless needed for logging or other purposes. The main reason for including station identification as a separate keyword block is in order to separate identification data from file data in the middle of file transmission and reception.

Example:

```
<ID 15 C2BC>(EA54}W1HKJ Toney, AL
```

<**FILE # CS**>*{HASH}*date-time:file-name, the modification date-time and name of the file to be transmitted is sent as data following this keyword.

Example:

```
<FILE 29 2499>{EA54}20120915022802:jabberwocky.txt
```

<**SIZE # CS**>*{HASH}FSIZE NBLOCKS BLKSIZE*. The size of the file, in bytes, the number of data blocks to be transmitted and the block size is sent as data following this keyword in normal ASCII decimal. The following example represents a file that is 200 bytes long and will be sent in 4 data blocks. Note that the 8 for character count indicates the length of the data, which is "200 4 64".

The data blocks will be 64, 64, 64 and 8 bytes long.

Example:

```
<SIZE 15 F244>{2499}200 4 64
```

<**DESC # CS**>*{HASH}* Textual description. An ASCII description of the file to be transmitted is sent as data following this keyword. This is an optional field.

<**DATA # CS**>*{HASH:BlkNum}*. The actual data contents of the file to be transmitted is sent as data following this keyword. File contents are usually sent in consecutive DATA blocks, each with a block number starting with 1, a character count (i.e. block size) and check sum. This a change from the original Amp protocol. The Block Number must be a part of the data line so that it is included in the Crc-16 checksum. Data transmissions may also consist of a partial list of DATA blocks.

Example 1:

```
<DATA 72 240C>{2499:1}[b256:start]526
=LZMA:0:0:3-]:0:0:0:4:0=l=--(-t;=-h-zw-------a)-
```

Example 2:

```
<DATA 72 A5E6>{2499:2})F----------,R-:3--=--k/:AMA--{-7-=---\9---\-:C-cO--)
```

(Example 1 contains an embedded line feed character)

(Example 2 contains compressed, base256 encoded data; unprintable characters represented by '-' and '=')

<**CNTL # CS**>*{HASH:control-word}*. This keyword indicates that a transmission control word is the data load. The control word may be either EOF (end of file) or EOT (end of transmission).

Examples:

```
<CNTL 5 9016>{2499:EOF}
<CNTL 5 301A>{2499:EOT}
```

1.3.4 Multiple file transfers

Several files may be transfered within a single Amp transmission. The protocol insures that each file and it's data blocks are uniquely identifiable so that an error free recreation of the file is possible at the transfer destination.

1.4 Transfer Examples

1.4.1 Original text:

```
 1. This quick brown fox jumped over the lazy dogs.
 2. This quick brown fox jumped over the lazy dogs.
 3. This quick brown fox jumped over the lazy dogs.
 4. This quick brown fox jumped over the lazy dogs.
 5. This quick brown fox jumped over the lazy dogs.
 ,...
40. This quick brown fox jumped over the lazy dogs.
```

1.4.2 Text data transfer

```
QST DE KK5VD
<PROG 18 E9CE>{0EE2}FLAMP X.X.X
<FILE 28 13C7>{0EE2}20130323070339:Fox.txt
<ID 29 DB98>{0EE2}KK5VD Madison AL EM64or
<SIZE 16 8816>{0EE2}2080 22 96
<DATA 104 CDFF>{0EE2:1} 1. This quick brown fox jumped over the lazy dogs.
 2. This quick brown fox jumped over the laz
<DATA 104 8B9A>{0EE2:2}y dogs.
 3. This quick brown fox jumped over the lazy dogs.
 4. This quick brown fox jumped over
<DATA 104 BD97>{0EE2:3} the lazy dogs.
 5. This quick brown fox jumped over the lazy dogs.
,...
<DATA 73 B61A>{0EE2:22} lazy dogs.
40. This quick brown fox jumped over the lazy dogs.

<CNTL 10 8E8D>{0EE2:EOF}
<CNTL 10 2E81>{0EE2:EOT}
QST DE KK5VD K
```

1.4.3 Compressed text data transfer

The above file was compressed using the LZMA algorithm and then base encode using base-64 encoding. Base-64 insures that all characters in the transmission will be contained within the supported character set of most available amateur radio transport layers (digital modem types).

```
QST DE KK5VD
<PROG 18 E9CE>{0EE2}FLAMP X.X.X
<FILE 28 13C7>{0EE2}20130323070339:Fox.txt
<ID 29 DB98>{0EE2}KK5VD Madison AL EM64or
<SIZE 14 B649>{0EE2}221 4 64
<DATA 72 E5D1>{0EE2:1}[b64:start]AUxaTUEAAAggXQAAAAQAEAxB/TEWllgk3J5mKpYEOyC+
<DATA 72 DEEE>{0EE2:2}VvFVmqBX/av7rhbtp4H4xoY39GsVdk5WZmH3Jm/CULrcdlB39rhWJUU
<DATA 72 57A9>{0EE2:3}l3yWPIV2q4fQDgazFXD/IqYKt7x07GamZsVFZORzk/TLgX4vBXqTc86
<DATA 37 44A4>{0EE2:4}Wpmo3mbURumL0Xj8cIE
[b64:end]
<CNTL 10 8E8D>{0EE2:EOF}
<CNTL 10 2E81>{0EE2:EOT}
QST DE KK5VD K
```

The same file compressed and then base-256 encoded (unprintable characters are shown as '-' or '=')

```
QST DE KK5VD
<PROG 18 E9CE>{0EE2}FLAMP X.X.X
<FILE 28 13C7>{0EE2}20130323070339:Fox.txt
<ID 29 DB98>{0EE2}KK5VD Madison AL EM64or
<SIZE 14 E7B2>{0EE2}195 4 64
<DATA 72 B4BB>{0EE2:1}[b256:start]150
:1LZMA:0:0:8 ]:0:0:0:4:0==A-1=-X$--f*-:4; -Si-7:
<DATA 72 7B22>{0EE2:2}6-:9[-Vj-_----[--:7-==---U-9Y-----:9B-q-A---X-=:9-4-s--]
<DATA 72 5F45>{0EE2:3}8=-U----*----------O-.:5---=M-::(c--!a%jf-y-Q=-/E---:4
[b256:e
<DATA 11 B776>{0EE2:4}nd]
<CNTL 10 8E8D>{0EE2:EOF}
<CNTL 10 2E81>{0EE2:EOT}
QST DE KK5VD K
```

Additional header information Programs using this protocol may add information both preceding and trailing the protocol blocks.

For example:

```
QST QST de W1HKJ Toney, Alabama
```