BATS Europe
FAST PITCH Specification

12 November 2008

Overview

The BATS FAST PITCH market data protocol delivers BATS market data messages wrapped in a SOUP envelope over a TCP connection. The underlying PITCH messages used are a superset of the messages used on the BATS Exchange in the United States. Some additional "long format" messages have been added which allow for larger sizes and prices.

Participants which have been using FAST PITCH successfully on BATS Exchange will need to make minor changes to be compatible with BATS MTF FAST PITCH. Since the messages are a superset, participants which successfully use BATS MTF FAST PITCH can use the same code, unmodified, with BATS Exchange.

The BATS Europe PITCH specification is available here:

http://www.batstrading.co.uk/resources/participant_resources/BATS_Europe_PITCH_Specification.pdf

BATS has developed a compressed feed based on the FAST protocol (http://www.fixprotocol.org/fast). BATS FAST PITCH compression typically achieves 73% compression on a PITCH data stream. How this translates to actual network bandwidth depends on the number of messages sent in a single TCP/IP packet. Many factors influence the number of messages in a single TCP/IP frame including: event rate, network utilization, and network latency. BATS FAST PITCH compression will decrease network utilization by 37% to 72% with results being directly proportional to network utilization.

BATS can optionally enable Nagel's algorithm (i.e. turn off `TCP_NO_DELAY`) on a PITCH port if desired by the subscriber. At the cost of a small increase in latency this can significantly increase the number of messages per TCP/IP frame, resulting in higher effective compression.

API

BATS provides source code to a decoder written in C, along with a sample program that can read compressed PITCH from a file and produce an uncompressed output file.

The FAST reference API currently provided by FPL does not implement incremental stream decoding or the FAST_OP_DEFAULT field operator. We recommend using the supplied BATS decoder to decode BATS FAST encoded PITCH.

The decoder has two public functions:

`BATS_FASTPITCH_InitCache``(void)`	Initialize decoder.
`int` **`BATS_FASTPITCH_DecodeMessage`**`(`	Decode bytes. Returns: `BFS_NeedMoreData` (-1) `BFS_Error` (-2) input bytes decoded (> 0)
`unsigned char const *src,`	First non-decoded input byte
`unsigned srcLen,`	Length of non-decoded data available
`unsigned char *dst,`	Buffer to hold single decoded output message
`unsigned dstLen,`	Length available at dst
`unsigned *dstUsed)`	Receives amount of dst used to hold decoded output message

The decoder will return `BFS_NeedMoreData` (-1) if sufficient input bytes for a full output message are not provided. In this case you should try again after receiving more data. When retrying, the original input bytes must be supplied again with the new data appended.

If there was an unrecoverable decoding error, the decoder will return `BFS_Error` (-2), in which case the only recourse is to disconnect and reconnect to the PITCH server. An unrecoverable decode error should only occur if the decoder is not being used correctly.

If the return value is positive, it indicates the number of encoded input bytes that were consumed. The next call to the decoder should not include these already consumed bytes. A single SOUP wrapped output message is written to a destination buffer provided to the decode function. The number of bytes written to the output buffer is returned via an output parameter.

The full SOUP conversation is encoded, so all bytes received by the client socket must be passed through the decoder prior to processing the SOUP envelope. The output of the decoder may be processed by your existing SOUP/PITCH logic.

The decoder keeps its state in a static global array in `batsfastdecode.c`. This is fine if your client process is processing a single PITCH stream. If you need to decode more than one PITCH stream per process you will need to modify the initialization and decoder functions to take the state array as a parameter.

Provided source files: http://www.batstrading.com/subscriber_resources/batsfastpitch.zip

batsfastcommon.h	Included by the decoder implementation
batsfastdecode.h	Included by your application
batsfastdecode.c	Decoder implementation
batsfastdecodetest.c	Sample program that uses decoder
pitchsample.dat	Non compressed pitch sample
pitchsample.enc	Compressed pitch sample

No makefile or project files are supplied.

The sample may be built as follows:

```
gcc –o batsfastdecodetest batsfastdecodetest.c batsfastdecode.c
```

or

```
cl batsfastdecodetest.c batsfastdecode.c
```

for gcc or MSVC, respectively.

Run as:

```
batsfastdecodetest pitchsample.enc pitchsample.dec
```

Then verify that generated `pitchsample.dec` is identical to supplied `pitchsample.dat`.

Messages from Client

All messages sent from the client to the server are not encoded and have the same meaning on a compressed FAST PITCH port as a normal PITCH port.

Intellectual Property

One of the key techniques used by FAST is protected by a patent held by the Chicago Mercantile Exchange ("CME").

Please see http://www.fixprotocol.org/fastagreement for details about how this may apply to you.

Latency and CPU Analysis

The following FAST performance statistics were gathered on servers running in the BATS Exchange production environment. BATS anticipates similar savings as the BATS MTF scales production volumes.

	Slowest Hop Bandwidth (bps)	Avg PITCH Size (bytes)	Avg FAST Size (bytes)	TCP/IP Overhead (bytes)	PITCH Serialization time (microseconds)	FAST Serialization time (microseconds)	Serialization savings (microseconds)	Encoder Time (microseconds)	Decoder Time (microseconds)	FAST Time saved (microseconds)
1G	1,000,000,000	38	11	40	0.624	0.408	0.216	0.526	0.303	-0.613
100M	100,000,000	38	11	40	6.240	4.080	2.160	0.526	0.303	1.331
50M	50,000,000	38	11	40	12.480	8.160	4.320	0.526	0.303	3.491
10M	10,000,000	38	11	40	62.400	40.800	21.600	0.526	0.303	20.771
5M	5,000,000	38	11	40	124.800	81.600	43.200	0.526	0.303	42.371

	Messages per Second	Encoder Time (microseconds)	Decoder Time (microseconds)	Encoder %CPU	Decoder %CPU
	10,000	0.526	0.303	0.5%	0.3%
	20,000	0.526	0.303	1.1%	0.6%
	30,000	0.526	0.303	1.6%	0.9%
recent 1 second peak	**40,000**	**0.526**	**0.303**	**2.1%**	**1.2%**
	50,000	0.526	0.303	2.6%	1.5%