TECHNICAL DOCUMENT

END OF DAY DATA - CSV SPECIFICATIONS

FUTURE & OPTIONS MARKET
(LEVEL 3)

29 JUN 2012

DOTEX INTERNATIONAL LIMITED
EXCHANGE PLAZA,
PLOT NO. C/1, G BLOCK,
BANDRA-KURLA COMPLEX,
BANDRA (E), MUMBAI 400 051.
INDIA.

COPYRIGHT NOTICE

1. DATA DETAILS

A single packet contains Header, data and trailer. The interpretation of the data depends on the code field.

1.1 THE HEADER

The header consists of following fields

Code – 2 char code field indicating type of data.

Length – 2- byte length field (Hex) indicating the length of the packet that is being sent. (Can be ignored by CSV users)

Sequence Number – 4- byte field indicating the sequence number (Hex) of the ASCII text being sent. (Can be ignored by CSV users)

1.2 THE DATA

The following information is provided in data block -

a) Market Update Information: The structure for the Market Update information is mentioned below in section 2.1.

1.3 TRAILER

The trailer is a two-byte checksum. A CR will terminate the block of data. (Can be ignored by CSV users)

2. DATA STRUCTURE DETAILS

2.1 MARKET DEPTH INFORMATION

These packets contain the latest order and trade information of F&O contracts up to the order book depth of **20**. These packets are sent during the market hours.

HEADER

a) Code	'**FV**'
a) Length	1064
b) Sequence Number	XXXX

DATA

	FIELD TYPE	**FIELD WIDTH**
1)	Instrument	Char 6 Bytes
2)	Symbol	Char 10 Bytes
3)	Expiry date	Char 11 Bytes
4)	Strike Price	Char 10 Bytes
5)	Option Type	Char 2 Bytes
6)	Market Type	Char 1 Byte
7)	Time Stamp	Char 11 Bytes
8)	MARKET_DEPTH_BUY_ORDER_INFO	[**20**]
	a. Best Buy-Order Price	Char 10 Bytes
	b. Best Buy-Order Quantity	Char 12 Bytes
9)	MARKET_DEPTH_SELL_ORDER_INFO	[**20**]
	a. Best Sell-Order Price	Char 10 Bytes
	b. Best Sell-Order Quantity	Char 12 Bytes
10)	Last Traded Price (Last Trade Price for a token would be ZERO till first occurrence of the trade on that token.)	Char 10 Bytes
11)	Total Traded Quantity	Char 12 Bytes
12)	Security Status	Char 1 Byte (If Suspended "S" or Blank)
13)	Opening Price	Char 10 Bytes
14)	High Price	Char 10 Bytes
15)	Low Price	Char 10 Bytes
16)	Close Price	Char 10 Bytes
17)	Average Traded Price	Char 10 Bytes
18)	Total Buy Quantity	Char 12 Bytes
19)	Total Sell Quantity	Char 12 Bytes
20)	Total Turnover	Char 25 Bytes

The above fields are comma separated. Hence the field widths are not applicable for CSV users.

TRAILER

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

3. CONTACT INFO

Name	Email	Contact Number
DOTEX Business	**dotex@nse.co.in**	91-22-26598385
Technical Support	**infofeed_support@nse.co.in** .	-

4. NOTE

Security Descriptor comprises Instrument Name, Symbol, Expiry Date, Strike Price & Option Type. Symbol indicates the index on which the FUTURES or OPTIONS contract is based (viz. CNX NIFTY) in case of Index Futures or Index Options or any stock (like ACC) in case of Future / Options on Individual stocks.

In the Pre-Open phase of the market, the Order Price can be 0.00. This indicates that the order placed is an At Open Order (ATO) and the actual Order Price will be calculated by the system when the market opens. Even though no trade takes place in the Pre-Open phase the Last Traded Price and the Open Price will change as the orders are placed in the system. The final open price for the security is calculated after Pre-Open period has ended. The Total Traded Quantity will be zero in the Pre-Open phase. In the Market Open phase also the Total Traded Quantity will be zero when orders are placed but no trades take place. The Last Traded Price will never be zero in the Normal Market.

5. CHECKSUM ALGORITHM

The **Checksum** routine followed for Info Vendor Feed is as follows:

```
// Following are the defines for checksum calculation
#define DC1           17
#define DC3           19
#define CR            13
#define LF            10
#define POLY    0x1021
// End of defines

unsigned check_sum (cData, iLength)
char *cData;
int iLength;
{
unsigned uAccum = 0;
unsigned uData;
unsigned char ucChk [2];
int i, j;

        for (i=0;i<iLength;i++){
                        uData = *(cData+i);
                        uData <<= 8;
                        for (j=8; j>0; j--){
                                if ((uData^uAccum)&0x8000)
                                        uAccum=(uAccum<<1)^POLY;
                                /* SHIFT AND SUBTRACT POLY */
                                else
                                        uAccum<<=1;
                                uData<<=1;
                        }
        }

        ucChk[0] = uAccum>>8;
        if (ucChk [0] == DC1 || ucChk [0] == DC3 || ucChk [0] == CR ||
ucChk[0] == LF)
                        UcChk [0] -= 1;
        UcChk [1] = uAccum&0xFF;
        if (ucChk [1] == DC1 || ucChk [1] == DC3 || ucChk[1] == CR || ucChk
[1] == LF )
                        UcChk [1] -= 1;
        uAccum = ucChk [1];
        uAccum = (uAccum<<8) + ucChk[0];

        return(uAccum);
}
```

6. EXAMPLE: FUNCTION FOR DECOMPRESSION

```
lzo_decomp (char cInputBuf [], unsigned int ipLength, char cOutputBuf [],
unsigned *opLength, unsigned short * lzo_errorcode)
{
        int r;
        Char mess [50];
        r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength,
                                (unsigned char *) cOutputBuf, opLength,
                                 NULL);

        If ( r == LZO_E_OK)
        {
         Print (mess," Decompressed %lu bytes back into %lu bytes\n",
                                    (long) ipLength, (long) *opLength);

         Return true;
        }

        Else
        {
        OutputDebug ("Internal error - decompression failed");
        Return false;
        }
}
```