TECHNICAL DOCUMENT

END OF DAY DATA - CSV SPECIFICATION

CAPITAL MARKET
(LEVEL 3)

29 JUN 2012

NSE Data & Analytics Limited
EXCHANGE PLAZA,
PLOT NO. C/1, G BLOCK,
BANDRA-KURLA COMPLEX,
BANDRA (E), MUMBAI 400 051.
INDIA.
COPYRIGHT NOTICE

All rights reserved. No part of this document may be reproduced or transmitted in any form and by any means without the prior permission of NSE Data & Analytics Limited.
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

   a) **Code** – Code field indicating type of data.
   b) **Length** – Length field indicating the length of the packet that is being sent. (Can be ignored by CSV users)
   c) **Sequence Number** – Field indicating the sequence number 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(\r) 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 securities up to the order book depth of 20. These packets are sent during the normal market hours. These packets would not be sent during Pre-Open session. The Online Index field in this packet indicates the value of the CNX Nifty when the particular trade occurred. These packets would be sent for series EQ, BE, DR and DE.

HEADER:
- a) Code ‘CV’
- b) Length 1057
- c) Sequence Number XXXX

ASCII DATA:
The format of the ASCII data sent is as follows:

<table>
<thead>
<tr>
<th>FIELD TYPE</th>
<th>FIELD WIDTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Symbol</td>
<td>10 Chars</td>
</tr>
<tr>
<td>b) Series</td>
<td>2 Chars</td>
</tr>
<tr>
<td>c) Market Type</td>
<td>1 Char (‘N’-Normal)</td>
</tr>
<tr>
<td>d) Time Stamp</td>
<td>11 Chars</td>
</tr>
<tr>
<td>e) MARKET_DEPTH_BUY_ORDER_INFO [20]</td>
<td></td>
</tr>
<tr>
<td>i. Best Buy-Order Price</td>
<td>10 Chars</td>
</tr>
<tr>
<td>ii. Best Buy-Order Quantity</td>
<td>12 Chars</td>
</tr>
<tr>
<td>f) MARKET_DEPTH_SELL_ORDER_INFO [20]</td>
<td></td>
</tr>
<tr>
<td>i. Best Sell-Order Price</td>
<td>10 Chars</td>
</tr>
<tr>
<td>ii. Best Sell-Order Quantity</td>
<td>12 Chars</td>
</tr>
<tr>
<td>g) Last Traded Price</td>
<td>10 Chars</td>
</tr>
<tr>
<td>(Last Trade Price for a token would be ZERO till first occurrence of the trade on that token.)</td>
<td></td>
</tr>
<tr>
<td>h) Last Traded Quantity</td>
<td>12 Chars</td>
</tr>
<tr>
<td>(Last Trade Quantity for a token would be ZERO till first occurrence of the trade on that token.)</td>
<td></td>
</tr>
<tr>
<td>i) Total Traded Quantity</td>
<td>12 Chars</td>
</tr>
<tr>
<td>j) Security Status</td>
<td>1 Char (Suspended (S) or Blank)</td>
</tr>
</tbody>
</table>
TRAILER:
Trailer contains checksum and a CR(\r') will terminate the data packet
3. **CONTACT INFO**

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Contact Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE Data &amp; Analytics Ltd</td>
<td><a href="mailto:marketdata@nse.co.in">marketdata@nse.co.in</a></td>
<td>91-22-26598385</td>
</tr>
<tr>
<td>Technical Support</td>
<td><a href="mailto:infofeed_support@nse.co.in">infofeed_support@nse.co.in</a></td>
<td>-</td>
</tr>
</tbody>
</table>
4. CHECKSUM

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;
        ucChk[1] -= 1;
}
```

The checksum calculation involves defining constants for the checksum calculation and then implementing a function that calculates the checksum by iterating over the data, performing bit shifts and XOR operations, and checking if the checksum matches any of the predefined values.
return(uAccum);
}