

# TECHNICAL DOCUMENT

REAL TIME DATA

CAPITAL MARKET  
(LEVEL 2)

**24 JAN 2012**



DOTEX INTERNATIONAL 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 DotEx Ltd.

# CONTENTS

<b>Chapter</b>	<b>Contents</b>	<b>Page No.</b>
1.	INTRODUCTION	4
2.	CONNECTION DETAILS	5
2.1.	STRUCTURAL DIAGRAM	5
2.2.	ONLINE REQUIREMENTS	5
2.3.	STEPS FOR AUTHENTICATION AND RECEIVING FEED	6
2.4.	STEPS FOR DECOMPRESSING FEED	9
3.	DATA DETAILS	11
3.1	THE HEADER	11
3.1.1	CODE	11
3.1.2	LENGTH	11
3.1.3.	SEQUENCE NUMBER	11
3.2	DATA BODY	11
3.3	TRAILER	11
4.	DATA STRUCTURE DETAILS	12
4.1.	LOGIN REQUEST	12
4.1.1.	INFO HEADER	12
4.1.2.	ASCII DATA	12
4.1.3.	INFO TRAILER	12
4.2.	LOGON RESPONSE	12
4.2.1.	COM HEADER	12
4.2.2.	INFO HEADER	12
4.2.3.	ASCII DATA	12
4.2.4.	INFO TRAILER	12
4.3.	MARKET OPEN	13
4.4.	MARKET CLOSE	13
4.5.	NEW UPDATE	14
4.6.	BROADCAST MESSAGE	15
4.7.	MULTIPLE INDICES BROADCAST	16
4.8.	SECURITY DESCRIPTOR	16
4.9.	END OF THE DAY STATUS	17
4.10.	INDEX INFORMATION	18
4.11.	CORPORATE ACTIONS	18
4.12.	END OF THE FEED	19
4.13.	HEARTBEAT SIGNAL	20
4.14.	SECURITY INFORMATION	21
4.15.	POST CLOSE SESSION START	21
4.16.	POST CLOSE SESSION END	22
5.	CONTACT INFO	23
6.	NOTE	24
6.1.	NORMAL MARKET	24
6.2.	AUCTION MARKET	24
7.	CHECKSUM	25
8.	EXAMPLE: FUNCTION FOR DECOMPRESSION	26

# REAL TIME DATA TECHNICAL SPECIFICATION

## CAPITAL MARKET

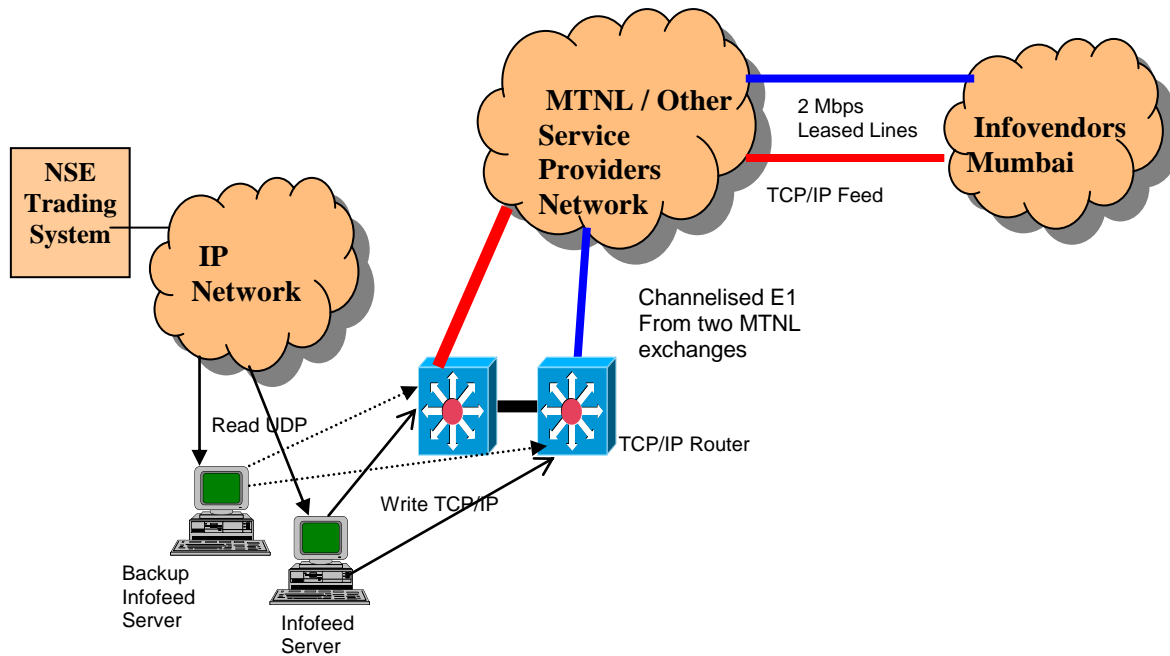
### 1. INTRODUCTION

DotEx international Ltd. disseminates NSEIL's real time broadcast data to various information agencies. It provides the 3 different types of data to vendors, i.e. Real Time Data, Snapshot Data and End of Day Data. The real time data is a packet broadcast available in TCP/IP format, where as the snapshot data and End of day data is available in the form of files. The Infofeed server provides NSEIL real time broadcast data. The information agencies connect to the Infofeed Server through 2 Mbps Leased Lines. These leased lines are terminated on Infofeed Router and their data specific pneumatic calls are forwarded to Infofeed server. The Infofeed server accepts these pneumatic calls and creates a socket connection. The TCP/IP data flows to the information agencies through these socket connections.

## 2. CONNECTION DETAILS

### 2.1. Structural Diagram

The structural diagram of Real Time data connection has been explained below -



### 2.2. Online Requirements

- A Router / Switch or a card with TCP/IP capabilities to connect to 2 Mbps transmission lines for receiving NSEIL's Real time information.
- The Information agency should develop applications that initiate TCP/IP calls through 2 Mbps Leased Line.

### 2.3. Steps for Authentication AND Receiving feed

- a) Client applications at vendor end, establish the connection with Infofeed Server application using specified IP address and Port.
- b) After establishing the connection, client application sends the login packet to Infofeed server application.

Packet format of Login packet (LOGIN\_REQ) is as follows,

```
struct LOGIN_REQ
{
    struct INFO_HEADER sHeader;
    struct LOGIN_REQ_DATA sData;
    struct INFO_TRAILER sTrailer;
};
struct LOGIN_REQ_DATA {
    char cUserId[10];
    char cPassword[8];
    char cNewPassword[8];
    char cConfmPassword[8];
};
struct INFO_HEADER
{
    short iCode;
    short iLen;
    LONG ISeqNo;
};
struct INFO_TRAILER
{
    short iChecksum;
    CHAR cEOT;
};
```

- c) Password field is case sensitive, password should be minimum 6 characters long, password and user id should not be same, password should start with alphabet and password should be alphanumeric (No wild characters are allowed).
- d) If user wants to change his password then the user needs to specify new password & confirm password (Both fields should match) otherwise leave it blank.
- e) Based on the above information, user will get Log on response (LOGIN\_RESPONSE) from Infofeed Server.

```
struct LOGIN_RESPONSE
{
    struct COM_HEADER sCOMHeader;
    struct LOGIN_RESPONSE_DETAIL sDetail;
```

```

};
struct LOGIN_RESPONSE_DETAIL
{
    struct INFO_HEADER sHeader;
    struct LOGIN_RESPONSE_DATA sData;
    struct INFO_TRAILER sTrailer;
};
struct COM_HEADER
{
    char cMopOrNot;
    short iPackLen;
    short iNoOfPack;
};
struct INFO_HEADER
{
    short iCode;
    short iLen;
    LONG lSeqNo;
};
struct INFO_TRAILER
{
    short iChecksum;
    CHAR cEOT;
};
struct LOGIN_RESPONSE_DATA
{
    long iErrCode;
    char cMesg[50];
};
};

```

Following Error code will be returned which client needs to interpret as:-

1000- Successful

1001- Password Update Successfully

1002- Wrong UserId-Password Combination

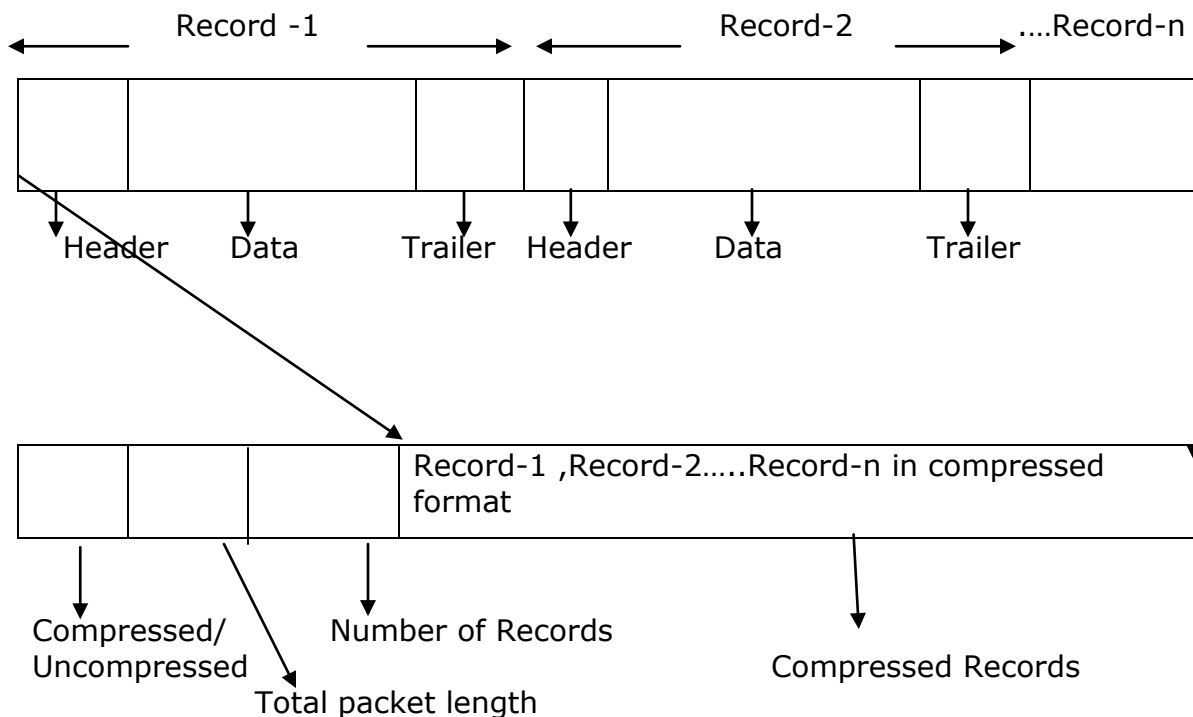
1003- Password is not valid in password change request.

1004- Login request is not correct.

Error code other than above - Error in receiving logon response

- f) After successful login, Infofeed Application starts sending packets in the below format.

## Packet Format:



Compressed/Uncompressed: This field tells whether packet is compressed or not compressed.

If this Field = 0 then Compressed.

Field = 1 then Uncompressed.

Number of Records: This field tells the number of records present in the compressed packets.

Packet length: this field specifies the total packet length.

```
Structure COM_HEADER
{
    char cMopOrNot;
    short iPackLen;
    short iNoOfPack;
};
```

g) As the data packets are sent in compressed format there is a need to decompress them. The compression algorithm used is LZO.

h) The Decompression algorithm used should be LZO

## 2.4. Steps for Decompressing feed

- LZO Algorithm Details:

- a) LZO is a data compression library which is suitable for data de-/compression in real-time. This means it favors speed over compression ratio.
- b) LZO is written in ANSI C. Both the source code and the compressed data format are designed to be portable across platforms.

LZO implements a number of algorithms with the following feature

- Decompression is simple and *\*very\** fast.
- Requires no memory for decompression.
- Compression is pretty fast.
- Requires 64 KB of memory for compression.
- Allows you to dial up extra compression at a speed cost in the Compressor.
- The speed of the decompressor is not reduced.
- Includes compression levels for generating pre-compressed data which achieve a quite competitive compression ratio.
- There is also a compression level which needs only 8 KB for Compression.
- Algorithm is thread safe.
- Algorithm is lossless.

LZO supports overlapping compression and in-place decompression.

- c) Files required for LZO algorithm.

- Include files, source files (src) provided by LZO
- LZO.lib

**For more information on LZO library and downloads visit:**

[http://www.nseindia.com/content/press/prs\\_whatsnew.htm#2](http://www.nseindia.com/content/press/prs_whatsnew.htm#2)

Specifications for utilizing on-line broadcast information

- **Decompression steps:**

- a) Receive the packet in the temporary buffer i.e. array of characters.
- b) First field will identify whether the packet is compressed or not.
- c) If this field is **0** then Decompress it using LZO algorithm else if **1 don't** decompress it and proceed in normal way as it is being done today.
- d) The second field is packet length.
- e) The third field contains the number of records in the packet.
- f) If compressed use following function of LZO to Decompress.

```
r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength,  
(unsigned char*) cOutputBuf, opLength, NULL);
```

**lzo1z\_decompress:** Function which decompresses the data packet receive

**CInputBuf:** Input buffer in which compressed data is received

**IpLength:** The length of the packet which application has received using Receive ().

**COutputBuf:** The uncompressed output data which is result of decompression.

**OpLength:** Length of uncompressed data

- g) After decompression data will be available in Output Buffer.
- h) Map the outputbuf to existing Header structure according to **CN** or **CX**.
- i) Look for Record size in the length field and Code (i.e. CN or CX).
- j) Steps to recover data from OutputBuf.

**Algorithm:**

```
Length_of_Record = Header->length;  
Sequence_no = Header->Sequence_num;  
For I = 0 to Number of records (obtained in step 4)  
Begin  
    Bytes_to_seek = Length_of_Record * I  
    Seek to number of Bytes_to_seek  
    Map (Length_of_Record) of bytes to proper structure according  
    to iCode (CN or CX) as found in Header part.  
    Do the required processing....  
    ....  
End  
End for Loop.
```

### 3. DATA DETAILS

The real time data is disseminated in the form of TCP/IP packets and each single packet generated by Infofeed system with a definite structure i.e. Header, Data body and Trailer.

#### 3.1. THE HEADER

The header in turn consists of 3 fields – Code, Length and Sequence number. The details of these fields are explained as below.

**Code** – It is a 2-Char field that provides the information about the type of packet or the type of data that each packet contains. The NSEIL real time data contains various types of packets that may or may not have data part. Each packet can be differentiated with the code filed. The various types of packets disseminated as real time data feed are - Heartbeat Signal (CH), Login Request (CQ), Login Response (CR), Market Open (CO), Market Close (CC), New Update (CN), Multiple Index Broadcast (CX), Security Addition (CA) and Modification (CM), End of day Market Status (CS), End of Feed (CE), Security Information (CT), Post Close Session Start(CK), Post Close Session End(CL), Pre-Open Session Start (PO),Pre-Open Session End(PC), Pre-Open New Update (PN) etc.

**Length** – It is a 2-byte hexadecimal field that provides the length of record within the each packet of NSEIL real time data. This includes the length of Header, Data, Trailer and the Carriage return.

**Sequence Number** – It is a 4-byte ASCII field that provides the sequence number of each packet that is disseminated in NSEIL real time data. The very first packet sequence number is initiated with number 1 (One) for a each market and each successive packet is incremented by one thereafter.

#### 3.2. DATA BODY:

The following information is provided in data block -

- a. Market Open
- b. Market Close
- c. Trade and Order information with Index (New update)
- d. Broadcast Messages
- e. Multiple Indices Broadcast
- f. Descriptor Addition, Modification and Deletion
- g. Market Status at the end of the day
- h. Index Information at the end of the Day
- i. Additional Index Information
- j. Corporate Actions Update
- k. End of the feed
- l. Heart Beat Signal
- m. Security Information
- n. Post Close Session Start

- o. Post Close Session End
- p. Pre-Open Session Start
- q. Pre-Open Session End
- r. Pre-Open New Update

### **3.3. TRAILER**

The trailer is a two-byte checksum. A CR will terminate the block of data.

## 4. DATA STRUCTURE DETAILS

### 4.1. LOGIN REQUEST (send by client application)

Login Request packet will be send by the client application for login into the Infofeed application. If user wants to change his password he will specify the new password and confirm password field. Password is case sensitive. Format of this packet is as follows.

#### 4.1.1 INFO HEADER:

- |                    |                         |
|--------------------|-------------------------|
| a) Code            | 'CQ'                    |
| b) Length          | Short Integer (2 Bytes) |
| c) Sequence Number | 00                      |

#### 4.1.2 ASCII DATA:

- |                     |          |
|---------------------|----------|
| d) User Id          | 10 Chars |
| e) Password         | 8 Chars  |
| f) New Password     | 8 Chars  |
| g) Confirm Password | 8 Chars  |

#### 4.1.3 INFO TRAILER:

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

### 4.2 LOGON RESPONSE (send by Infofeed application)

Logon response packet will be send by the Infofeed server application after receiving the Login Request packet from the client application.

#### COM HEADER:

- |                      |                         |
|----------------------|-------------------------|
| a) Compressed or Not | 1 Char                  |
| b) Packet Length     | Short Integer (2 Bytes) |
| c) No of Packets     | Short Integer (2 Bytes) |

#### INFO HEADER:

- |                    |                         |
|--------------------|-------------------------|
| a) Code            | 'CR'                    |
| b) Length          | Short Integer (2 Bytes) |
| c) Sequence Number | Long (4 Bytes)          |

#### ASCII DATA:

- |               |                |
|---------------|----------------|
| a) Error Code | Long (4 Bytes) |
| b) Message    | 50 Chars       |

#### INFO TRAILER:

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

### 4.3. MARKET OPEN

The Market Open message is received whenever any particular market opens for the day. Market Open message will be received separately for different markets.

Pre Open Session start information will be sent through same packet with code field as **'PO'**.

#### HEADER:

- |                    |  |
|--------------------|--|
| a) Code            | <b>'CO'</b> For Normal Market Open<br><b>'PO'</b> For Pre-Open Session Start |
| b) Length          | 0x0C   |
| c) Sequence Number | XXXX   |

#### ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Market Type	1 Char ( <b>'N'</b> -Normal, <b>'S'</b> - Spot, <b>'O'</b> - Odd Lot, <b>'A'</b> -Auction, <b>'L'</b> -All Market)

#### TRAILER:

Checksum is not computed.

### 4.4. MARKET CLOSE

The Market Close message will be received whenever any particular market closes for the day. Market Close message will be received separately for different markets.

Pre Open Session end information will be sent through same packet with code field as **'PC'**.

#### HEADER:

- |                    |  |
|--------------------|--|
| a) Code            | <b>'CC'</b> For Normal Market Open<br><b>'PC'</b> For For Pre-Open Session Ended |
| b) Length          | 0x0C   |
| c) Sequence Number | XXXX   |

#### ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Market Type	1 Char ( <b>'N'</b> -Normal, <b>'S'</b> - Spot, <b>'O'</b> - Odd Lot,

'A'-Auction,  
'L'-All Markets)

**TRAILER:**

Checksum is not computed.

**4.5 NEW UPDATE**

These packets will be received only during the trading period of the particular markets and signify the trades occurring in that market. The Online Index field in this packet indicates the value of the S&P CNX Nifty when the particular trade occurred.

Pre-Open session new updates will be sent through same packet with code field as 'PN'.

In Pre open Session only 4 best buy and sell side order information will be sent. In 5<sup>th</sup> price and quantity field the buy and sell side ATO (i.e. At The Opening) order information will be sent. For ATO orders the price will always be zero. During Pre-Open session the indicative open price (IOP) for the security will be sent in Open Price field.

**HEADER:**

- |                    |   |
|--------------------|---|
| a) Code            | 'CN' Normal Market New Update<br>'PN' Pre-Open Session New Update |
| b) Length          | 0x182   |
| c) Sequence Number | XXXX  |

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Symbol	10 Chars
2) Series	2 Chars
3) Market Type	1 Char ('N'-Normal, 'S'- Spot, 'O'- Odd Lot, 'A'-Auction)
4) Best Buy-Order Price1	10 Chars
5) Best Buy-Order Quantity1	12 Chars
6) Best Buy-Order Price2	10 Chars
7) Best Buy-Order Quantity2	12 Chars
8) Best Buy-Order Price3	10 Chars
9) Best Buy-Order Quantity3	12 Chars
10)Best Buy-Order Price4	10 Chars
11)Best Buy-Order Quantity4	12 Chars
12)Best Buy-Order Price5	10 Chars (Buy ATO order price information for 'PN' packet)
13)Best Buy-Order Quantity5	12 Chars (Buy ATO order quantity information for 'PN' packet)

14)Best Sell-Order Price1	10 Chars
15)Best Sell-Order Quantity1	12 Chars
16)Best Sell-Order Price2	10 Chars
17)Best Sell-Order Quantity2	12 Chars
18)Best Sell-Order Price3	10 Chars
19)Best Sell-Order Quantity3	12 Chars
20)Best Sell-Order Price4	10 Chars
21)Best Sell-Order Quantity4	12 Chars
22)Best Sell-Order Price5	10 Chars (Sell ATO order price information for ' <b>PN</b> ' packet)
23)Best Sell-Order Quantity5	12 Chars (Sell ATO order quantity information for ' <b>PN</b> ' packet)
24)Last Traded Price	10 Chars
25)Last Traded Quantity	12 Chars
26)Total Traded Quantity	12 Chars
27)Security Status	1 Char (Suspended ( <b>S</b> ) or Blank)
28)Opening Price	10 Chars (IOP for ' <b>PN</b> ' packet)
29)High Price	10 Chars
30)Low Price	10 Chars
31)Close Price	10 Chars
32)Average Traded Price	10 Chars
33)Total Buy Quantity	12 Chars
34)Total Sell Quantity	12 Chars
35)Total Turnover	25 Chars
36)Online Index	8 Chars

**TRAILER:**

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

**4.6 BROADCAST MESSAGE**

These packets consist of the messages broadcast during the Trading time containing information like changes in the price bands of particular scrips and market-related information.

**HEADER:**

- a) Code **'CB'**
- b) Length 0xXX (Variable depending on the length of the Message string)
- c) Sequence Number XXXX

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Message Code	3 Chars ( <b>NSE</b> or <b>AUC</b> )

- |                   |                               |
|-------------------|-------------------------------|
| 2) Message Length | 3 Chars                       |
| 3) Message String | (Message Length) No. Of Chars |

**TRAILER:**

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

**4.7 MULTIPLE INDICES BROADCAST**

The Multiple Indices broadcast is timer driven event and this information is broadcasted at predefined time interval. This includes the latest information of all 24 NSE Indices viz. S&P CNX Nifty, CNX IT Sector Index, CNX Nifty Junior, S&P CNX Defty, CNX Bank Index, CNX Midcap, S&P CNX 500, CNX 100, Nifty Midcap 50, CNX Reality, CNX Infrastructure, India VIX, CNX Energy, CNX FMCG, CNX MNC, CNX Pharma, CNX PSE, CNX PSU Bank, CNX Service, CNX Smallcap, CNX 200, CNX Auto, CNX Media, CNX Meta & CNX Dividend Opptl. This Multiple Indices broadcast will continue throughout the day viz. before and after the trading hours also.

During Pre-Open Session till the final opening index is not computed the Current Index Value Field will contain the indicative index.

**HEADER:**

- |                    |      |
|--------------------|------|
| a) Code            | 'CX' |
| b) Length          | 0x5C |
| c) Sequence Number | XXXX |

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Index Name	17 Chars
2) Current Index Value	8 Chars
3) Open Index Value	8 Chars
4) Close Index Value	8 Chars
5) High Index Value	8 Chars
6) Low Index Value	8 Chars
7) Percent Change	8 Chars
8) Yearly High Index Value	8 Chars
9) Yearly Low Index Value	8 Chars

**TRAILER:**

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

#### 4.8 SECURITY DESCRIPTOR ADDITION / MODIFICATION / DELETION

This packet consists of information about addition, modification or deletion any of the Security (Descriptor). This is sent as a part of the End of the Day feed.

##### HEADER:

a) Code	`CA' or `CM' or `CD'
b) Length	0x6B
c) Sequence Number	XXXX

##### ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Symbol	10 Chars
2) Series	2 Chars
3) Security Description	30 Chars
4) Regular Lot	5 Chars
5) Market Type	1 Char (`N'- Normal, `S' - Spot, `O' - Odd Lot, `A'- Auction)
6) Tick Size	6 Chars
7) Face Value	9 Chars
8) Issue Capital	12 Chars
9) Market Index Participation	1 Char (`Y'-Yes, `N' -No)
10) Last Update Date & Time	20 Chars (Format: DD-MON-YYYY HH: MM: SS)

##### TRAILER:

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

#### 4.9 END OF THE DAY MARKET STATUS

At the end of the day, market status will be sent for all the descriptors in the system. The feed will be sent during all trading days.

##### HEADER:

a) Code	`CS'
b) Length	0x79
c) Sequence Number	XXXX

##### ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Symbol	10 Chars
2) Series	2 Chars
3) Market Type	1 Char (`N'-Normal, `S'- Spot, `O'- Odd Lot, `A'-Auction)

4) Trade High Price	10 Chars
5) Trade Low Price	10 Chars
6) Opening Price	10 Chars
7) Closing Price	10 Chars
8) Last Traded Price	10 Chars
9) Previous Close Price	10 Chars
10) Total Traded Quantity	12 Chars
11) Total Traded Value	25 Chars

**TRAILER:**

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

**4.10 INDEX INFORMATION**

This packet will be sent as a part of the End of the Day feed on the trading days.

**HEADER:**

a) Code	'CI'
b) Length	0x4F
c) Sequence Number	XXXX

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Date	11 Chars (DD-MON-YYYY)
2) Index Name	17 Chars
3) Opening Index	8 Chars
4) Closing Index	8 Chars
5) High Index	8 Chars
6) Low Index	8 Chars
7) Previous Closing Index	8 Chars

**TRAILER:**

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

**4.11 CORPORATE ACTIONS UPDATE**

The Corporate Actions data is sent as part of the End of the Day feed

**HEADER:**

a) Code	'CU'
b) Length	0x95
c) Sequence Number	XXXX

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Symbol	10 Chars
2) Series	2 Chars
3) Instrument Type	1 Char (Valid Values: '0'-Equities, '1'- Preference Shares '2' - Debentures, '3' - Warrants '4' - Miscellaneous, '5'- Others)
4) Issue Capital	12 Chars
5) Face Value	9 Chars
6) Market Lot	5 Chars
7) Dividend/Interest Rate	6 Chars
8) Record Date	10 Chars (Format: YYYY-MM-DD)
9) Book Closure Start Date	10 Chars (Format: YYYY-MM-DD)
10)Book Closure End Date	10 Chars (Format: YYYY-MM-DD)
11)Ex Date	10 Chars (Format: YYYY-MM-DD)
12)No Delivery Start Date	10 Chars (Format: YYYY-MM-DD)
13)No Delivery End Date	10 Chars (Format: YYYY-MM-DD)
14)Dividend	1 Char ('D' or Blank)
15)Rights Flag	1 Char ('R' or Blank)
16)Bonus Flag	1 Char ('B' or Blank)
17)Interest Flag	1 Char ('I' or Blank)
18)AGM Flag	1 Char ('A' or Blank)
19)EGM Flag	1 Char ('E' or Blank)
20)Others Flag	1 Char ('O' or Blank)
21)Corp Data Type	1 Char ('B' -Book Closure 'R'-Record Date, 'N'-None)
22)Corp Action Description	25 Chars

**TRAILER:**

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

**4.12 END OF THE FEED**

This end of the packet indicates that all the parts of End of the Day feed have been completed.

**HEADER:**

a) Code	'CE'
b) Length	0x0B
c) Sequence Number	XXXX

**ASCII DATA:**

Not associated with any ASCII data.

**TRAILER:**

Checksum is not computed.

#### 4.13 HEARTBEAT SIGNAL

The Heartbeat packets are sent throughout the day from 8:00 a.m. to 08:30 p.m. This packet indicates to the Info-Vendors that data packets are received from the Infofeed server.

##### HEADER:

a) Code	<b>`CH`</b>
b) Length	0x0B
c) Sequence Number	XXXX

##### ASCII DATA:

Not associated with any ASCII data.

##### TRAILER:

Checksum is not computed.

#### 4.14 SECURITY INFORMATION

These packets are sent at the beginning of the each trading day by approximately **08:45 AM**. This feed contains the information about the securities valid in the CM Market for trading.

##### HEADER:

d) Code	<b>`CT`</b>
e) Length	0x2E
f) Sequence Number	XXXX

##### ASCII DATA:

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
1) Token Number	10 Chars
2) Symbol	10 Chars
3) Series	2 Chars
4) ISIN Number	12 Chars
5) Is Deleted	1 Char

##### TRAILER:

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

#### 4.15 POST CLOSE SESSION START

This message is received whenever the post close session starts for the day.

**HEADER:**

d) Code	'CK'
e) Length	0x0C
f) Sequence Number	XXXX

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
Market Type	1 Char ('N'-Normal)

**TRAILER:**

Checksum is not computed.

#### 4.16 15 POST CLOSE SESSION END

This message is received whenever the post close session ends for the day.

**HEADER:**

g) Code	'CL'
h) Length	0x0C
i) Sequence Number	XXXX

**ASCII DATA:**

The format of the ASCII data sent is as follows:

<b>FIELD TYPE</b>	<b>FIELD WIDTH</b>
Market Type	1 Char ('N'-Normal)

**TRAILER:**

Checksum is not computed.

## 5. CONTACT INFO

Following are the contact details for business assistance:

<b>Name</b>	<b>Email Address</b>	<b>Contact Numbers</b>
Mr. Ved Malla	vmalla@nse.co.in	91-22-26598385
Ms. Prachee Chavan	pracheec@nse.co.in	91-22-26598385
Mr. Pankaj Agarwal	pagarwal@nse.co.in	91-22-26598385

You can also email us on **dotex@nse.co.in**

For technical assistance email us on **infofeed\_support@nse.co.in**.

## **6. NOTE**

### **6.1 Normal Market**

In Pre Open session the indicative index will be disseminated. After Pre open session ends and before the normal market opens the final open index value will be disseminated.

In Pre Open New Update message the open price field will contain the indicative opening price and the last traded price field will be zero ('0'). After pre open session ends and before the normal market opens trades will result due to pre-open session order matching. During this period open price field will contain the final open price for the security and last traded price field will be updated with the last traded price.

### **6.2 Auction Market:**

In the auction market the Open price and the Last Traded Price would be zero till the auction ends and the auction price is calculated by the system. Since Auction in any particular scrip is done at a fixed price the High Price, Low Price, Closing Price and Index values is zero for all scrips traded in the Auction Market.

The auction market timings are from 12:00 hrs to 13:00 hrs.

## 7. 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;
    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);
}
```

## 8. 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;
    }
}
```