NSE-OFFLINE DATA FEED

Version: 1.2
Date: September 09, 2015

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

Revision History

Name	Description	Date
Version 1.0	New Specification Issued	October 16, 2012
Version 1.1	Correction in ST_COMP_BATCH_HEADER Point no 3.	November 30, 2012
Version 1.2	Added Information for FO segment.	September 09, 2015

Table of Contents

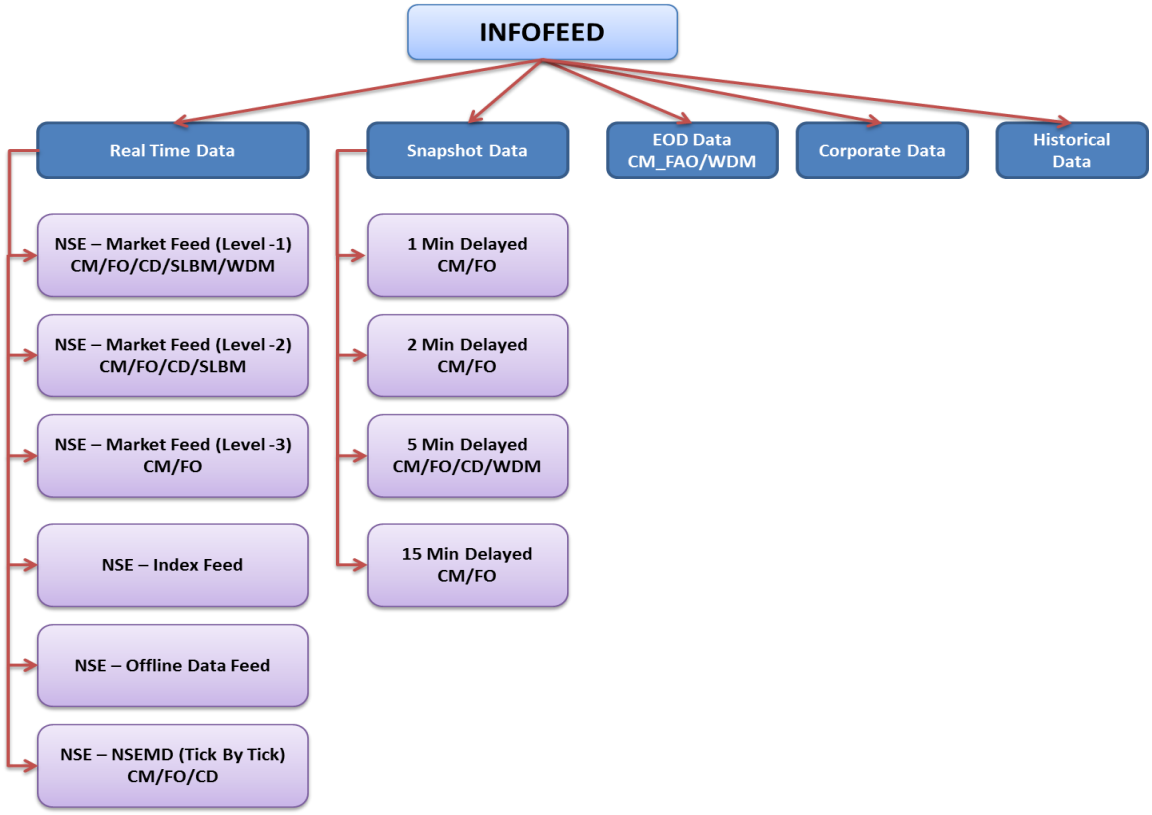
NSE – Offline Data Feed

1. Introduction

DotEx International Ltd. disseminates NSEIL's real time broadcast data to various information agencies. It provides the 5 different types of data products viz.

A. Real Time Data
B. Snapshot Data
C. End of Day Data
D. Corporate Data
E. Historical Data

The real time data and corporate data is a packet broadcast available in TCP/IP format, whereas the snapshot data, end of day data and historical data is available in the form of files. All these data products come under in Infofeed application.



In Infofeed's Real Time Data product following sub-products are available

a. NSE - Market Feed (CM/FO/CD/SLBM/WDM Level 1)
b. NSE - Market Feed (CM/FO/CD/SLBM Level 2)
c. NSE - Market Feed (CM/FO Level 3)
d. NSE - Index Feed
e. NSE - Offline Data Feed
f. NSE - NSEMD (CM/FO/CD Tick By Tick)

This document explains about the NSE – Offline Data Feed product. Through this product clients can recover the missed out data from the Infofeed server. Through offline data server following three options are available to recover the data.

1. Master Information (i.e. Data sent before market open i.e. BOD data)
2. End Of Day Information (i.e. Data sent after market close i.e. EOD data)
3. Sequence number based recovery of market information (i.e. Complete Data sent during the day. BOD,ONLINE & EOD data)

The information agencies connect to the Offline Data Feed Server through leased lines. These leased lines are terminated on Infofeed Router and their data specific pneumonic calls are forwarded to Infofeed server. The Infofeed server accepts these pneumonic calls and creates a socket connection. The TCP/IP data flows to the information agencies through these socket connections.

The feed consist of series of sequenced variable length compressed messages. The compression algorithm used over here is LZO – Compression.

This document can be used for all the market segments. This document explains how to connect and what all different types of recovery options available.

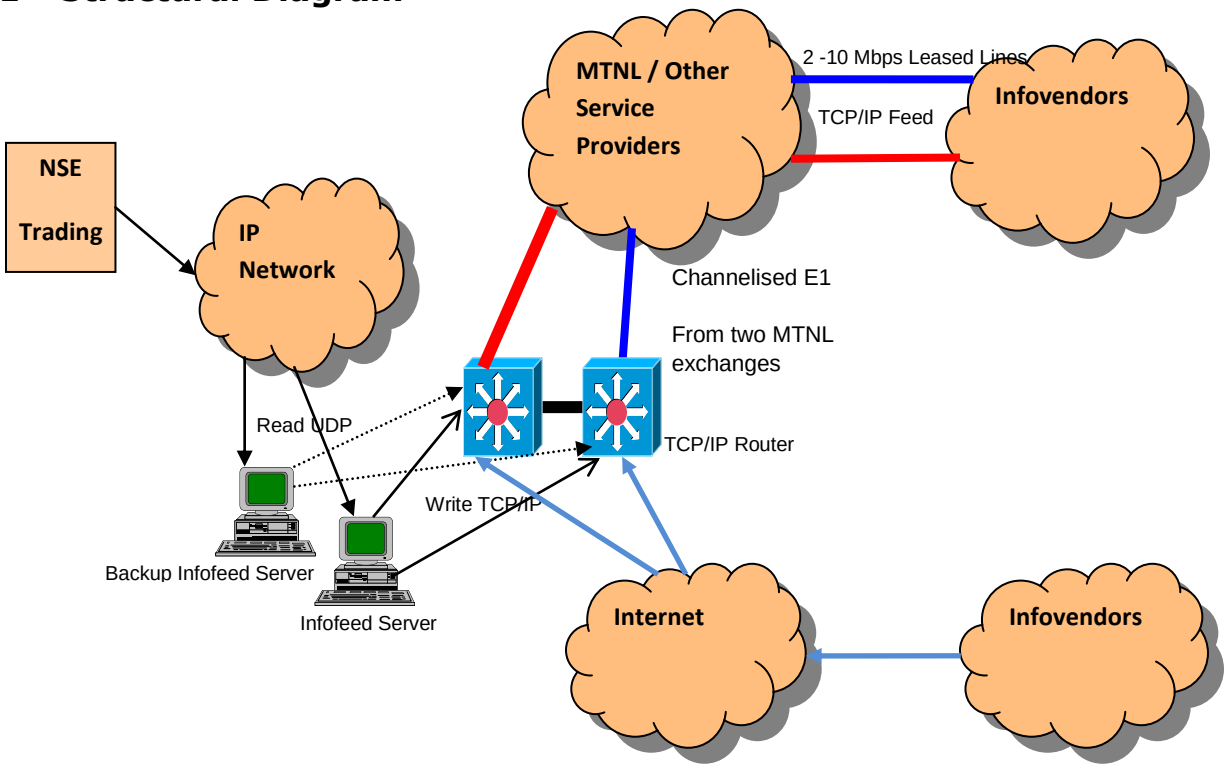
2. Session Initialization

NSE Offline Data Feed is built on TCP/IP socket connection. This feed consists of sequenced and unsequenced messages. Unsequenced messages provides the login and connection related messages such as login and heartbeats messages. Unsequenced messages are not part of the data. The sequenced data contains the actual market data and are reliable and recoverable.

A session begins with client establishing a TCP connection and sending the login request packet. Once the login request received the server authenticate it and send the login response. If the login is successful server will begin to send the sequenced data, or reject the login and terminate the connection.

Data packet consist of sequence number as one field. The first sequenced message of the day will send the sequence number as 1 and after that it will be incremented by 1 for each sequenced message. Client can recover the missed out data from separate NSE Offline Data system.

2.1 Structural Diagram



2.2 Online Requirements

a) A Router / Switch or a card with TCP/IP capabilities to connect to 2 Mbps transmission lines for receiving NSEIL's Real time information.
b) The Information agency should develop applications that initiate TCP/IP calls through 2 Mbps Leased Line.
c) Information agency can connect to the Infofeed servers through the internet also. For IP validation at application level, information agencies has to provide the public static IP from which they will connect to Infofeed servers. Connectivity through internet is available for some products only.

2.3 Data Types

Data types used in feed,

Data Type	Size In Bytes
CHAR	1
INT	4
LONG	4
DOUBLE	8

Byte order - Big Endean.

2.4 Acronyms Used

BOD	Begin Of Day Information
EOD	End Of Day Information
ONLINE	Information Sent During Market Timing
CM	Cash Market
F&O	Future & Options Market
CD	Currency Derivatives Market
SLBM	Securities Lending & Borrowing Market
WDM	Wholesale Debt Market

3. Packet Format

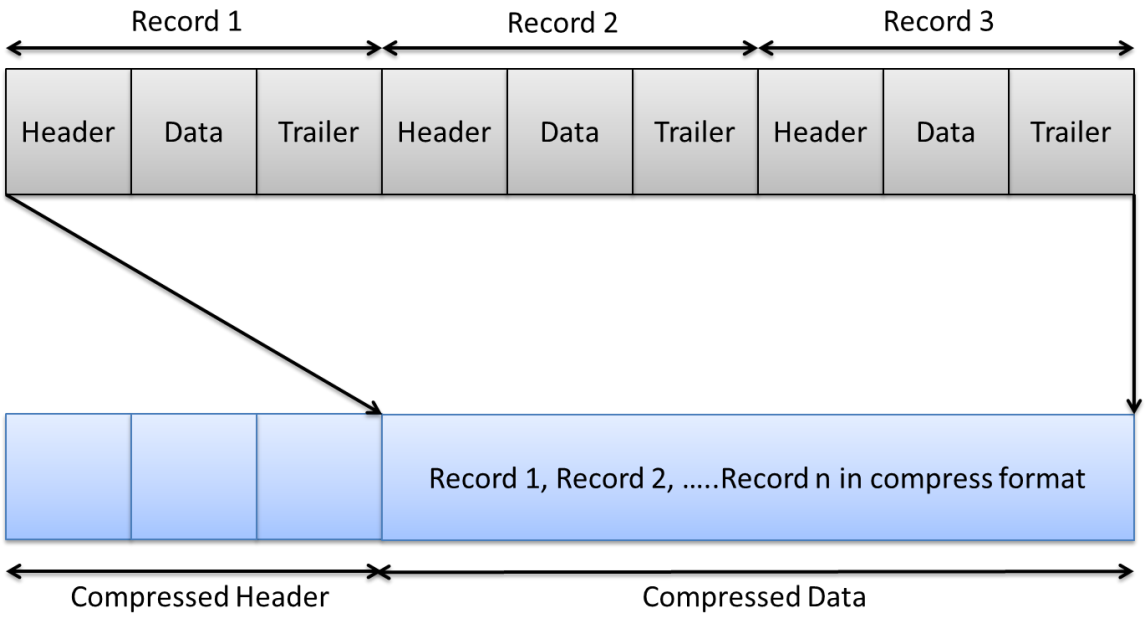
Server sends all the packets in following format

```
typedef struct
{
        CHAR        cCompOrNot
        SHORT       nDataSize;
        SHORT       iNoOfPackets;
}ST_COMP_BATCH_HEADER

typedef struct
{
        SHORT       iCode;
        SHORT       iLen;
        LONG        lSeqNo;
} ST_INFO_HEADER;

typedef struct
{
        .
        .
}ST_DATA_INFO;

typedef struct
{
        SHORT       iCheckSum;
        CHAR        cEOT;
} ST_INFO_TRAILER;

typedef struct
{
        ST_INFO_HEADER  stInfoHdr;
        ST_DATA_INFO    stDataInfo;
        ST_INFO_TRAILER stInfoTrailer;
        .
}ST_DATA_PACKET
```

All the packets received from server consist of compress batch header. Compress batch header gives the information about the data packet compressed or not, number of packets in the following data packet and the

total size of data packet. Client needs to decompress the data packet using LZO decompression algorithm. After decompression each data packet consists of ST_INFO_HEADER, which has the iCode field to identify the type of the packet. Using iCode field, data info packet is mapped to the respective data packet.

3.1 Diagrammatic Representation of Packet Format:



Compressed Header

1. Compressed/ Uncompressed = 0 then compressed/ 1 uncompressed
2. Number of packets = Number of records in compressed data
3. Data Size = Compressed data size

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

4. Session Messages

Session messages are not considered as market data messages. These messages provide the connection and login related messages such as login, and heartbeat messages.

4.1 Login Request (Sent by client)

Login request packet is sent by the client immediately after connecting to the server. This packet doesn't contain the compress batch header. If the client wants to change his default password then he needs to send "New Password" and "Confirm Password" in the request otherwise it should be kept blank. Password is case sensitive.

Field Name	Data Type	Value	Remark
INFO HEADER			
Code	SHORT	CM: 'CQ' FO: 'FQ'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0(Zero) for login request
INFO DATA			
User Id	CHAR[10]	Alphanumeric	Exchange provided user id
Password	CHAR[8]	Alphanumeric	Password
New Password	CHAR[8]	Alphanumeric	New password
Confirm Password	CHAR[8]	Alphanumeric	Confirm password
Offline Data Code	SHORT	Numeric	1=Master Information 2=EOD Information 3=Sequence Number Based Recovery
Start Sequence Number	LONG	Numeric	Missed out data's start sequence number
End Sequence Number	LONG	Numeric	Missed out data's end sequence number
INFO TRAILER			
Check Sum	SHORT	Numeric	Refer point no. 6
End Of Trailer	CHAR	'\r'	Carriage Return

In Offline Data download login request "Offline Data Code" field is used to identify the type of offline data download request

For "Offline Data Code" field possible values are
1 = BOD Data only
2 = EOD Data only
3 = BOD, ONLINE & EOD data

For BOD & EOD offline data download request (i.e. "Offline Data Code" = 1 /2) the start and end sequence number will be ignored in login request. The data packets sent in this data download request will have the sequence number as zero (0).

For BOD, ONLINE & EOD data offline download request (i.e. "Offline Data Code" = 3) client has to send the Start Sequence Number and End Sequence Number

- Start Sequence Number = Missed out data's start application sequence number.
- End Sequence Number = Missed out data's end application sequence number

The data sent through infofeed data products has the application sequence number assigned to each sequenced data packet. Recovery of that sequenced data is done through Offline data product. To map the data packet received in offline data feed client has to refer the respective market segment's **NSE - Market Feed (Level 1)/ NSE - Market Feed (Level 2) and NSE – Index Feed** data feed specification.

The maximum data that can be downloaded through this request is restricted to 500000 records (i.e. (End Sequence Number – Start Sequence Number) <= 500000). If Client missed out data is more than the 500000 records then he has to send multiple times this request with proper sequence number in start and end sequence number field.

4.2 Login Response (Sent by server)

Login response packet will be sent by server after receiving the login request packet. This packet doesn't contain the compress batch header.
This packet contains the error code from which the client can identify the status of the login.

Field Name	Data Type	Value	Remark
INFO HEADER			
Code	SHORT	CM: 'CR' FO: 'FR'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0(Zero) for login response
INFO DATA			
Error Code	LONG		1000- Successful Login 1001- Password Update Successfully 1002- Wrong UserId-Password Combination 1003- Password is not valid in password change request. 1004- Login request is not correct. 1006- Client Disabled 1007- Password update failed 1008- Client Already Connected 1011- Invalid Start End Sequence Number 1012- Invalid Offline Data Code Error code other than above - Error in receiving logon response
Error Message	CHAR[50]		Description about the error code
INFO TRAILER			
Check Sum	SHORT	Numeric	Refer point no. 6
End Of Trailer	CHAR	'\r'	Carriage Return

5. Steps For Decompressing The Data Packets

5.1 LZO Algorithm Details

LZO is a data compression library which is suitable for data de-/compression in real-time. This means it favors speed over compression ratio.

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.
- 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 decompression 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.

5.2 Files required for LZO algorithm.

- Include files, source files (src) provided by LZO
- LZO.lib
- LZO library version used is 1.0.7

5.3 Decompression steps

Receive the packet in the temporary buffer i.e. array of characters.

The first field is compressed or not compresses?
The second field is the number of packet in the following data packet.
The third field is data packet length.

Use the following function of LZO to Decompress.

r = lzo1z_decompress ((lzo_byte*)cInputBuf, ipLength, (lzo_byte*)cOutputBuf, (lzo_uint*)&opLength, NULL);

lzo1z_decompress: Function which decompresses the data packet received
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
After decompression data will be available in Output Buffer.

Each output data packet contains the INFO HEADER, after mapping the output decompressed buffer to INFO HEADER find out the data packet and the according to it map the output buffer to respective data packet.

Algorithm:

```
ST_NIFO_HEADER  *pstInfoHeader;
for (i=0; i < iNoOfPackets; i++) // iNoOfPackets received in
                                 // compressed data header
{
      pstInfoHeader = (ST_NIFO_HEADER   *) cOutputBuf
            switch (pstInfoHeader->iCode)
      {
                  case CX:     //Indices Information
                  {
            ST_INDEX_DATA     *stIndexData     =     (ST_INDEX_DATA
            *)cOutputBuf;
                  .
                  .
                  cOutputBuf = cOutputBuf + sizeof(ST_INDEX_DATA);
                  break;
                  }
            }
}
```

6. Checksum Calculation 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);
}
```

7. Support Information

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