Market Feed
Corporate Bond Market

Version: 1.4

DATE: 12 February 2024

NSE DATA & ANALYTICS LIMITED
EXCHANGE PLAZA,
PLOT NO. C/1, G BLOCK, BANDRA-KURLA
COMPLEX, BANDRA E, MUMBAI 400 051.
INDIA.

COPYRIGHT NOTICE

Revision History

Version	Description	Date
1.0	Final Specification Issued	28 August 2015
1.1	Trade structure revised	12 October 2015
1.2	Trade structure revised	12 January 2021
1.3	Removal of TCP/IP Format, Session Initialization, Login Request, Login Response.	29 October 2021
1.4	Online Trade structure revised: • "Buyer Deal Type" added • Existing "Deal Type" changed to "Seller Deal Type"	12 February 2024

Table of Contents

CORPORATE BOND MARKET – REAL TIME DATA

1. Introduction

NSE Data & Analytics Ltd. disseminates NSEIL's Real time Broadcast data to various information agencies. It provides the 3 different types of data to Info Vendors, i.e., Real time Data, Snapshot Data and End of Day Data. Real time Data is a packet broadcast available through Multicast protocol, whereas the Snapshot Data and End of day data are available in the form of files. Certain products based on Real time Data are also made available through files.

This document explains the NSE – CBRICS Feed product. Through this product, information NSE's CBRICS (Corporate Bond Reporting Platform - http://www.bricsonline.net) information is disseminated on real time basis.

2. Packet Format

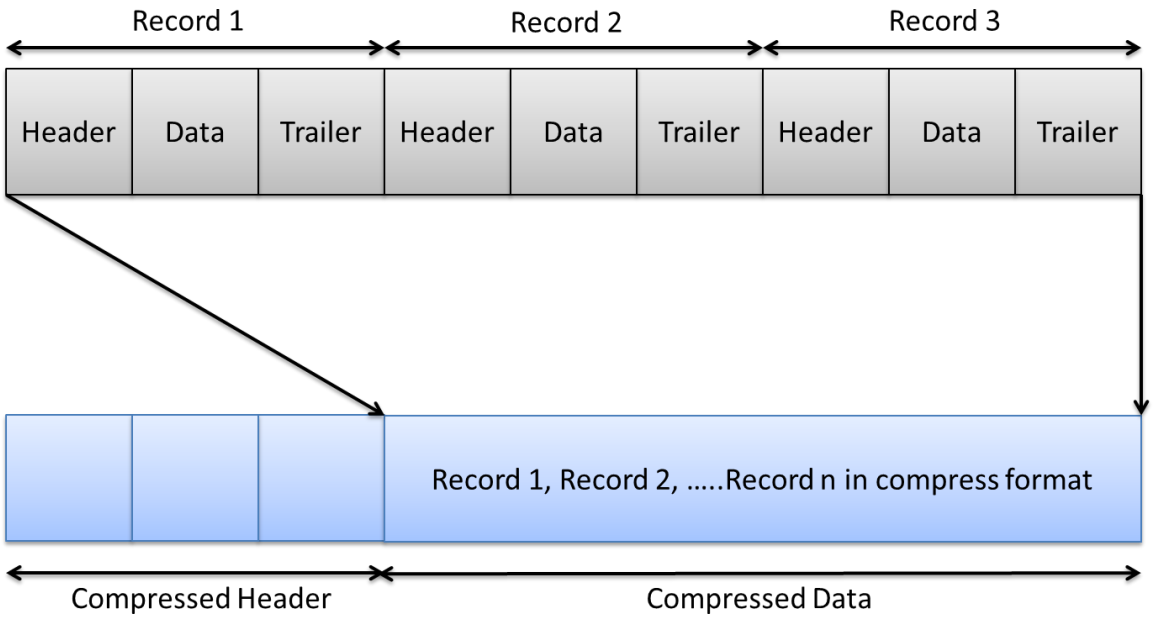
2.1 Data Type

Data Type	Size In Bytes
CHAR	1
SHORT	2
INT	4
LONG	4
DOUBLE	8

2.2 Diagrammatical Representation

1. Compression Header: Also called as Compression Header, this section of the packet describes whether the data packet is compressed, how many messages it carries, etc.
2. Data Payload: This contains the actual data messages or records.

The format is diagrammatically represented as follows:



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.

All the packets received from server consist of Compression Header. Compression Header indicates whether the data is compressed or not. It also specifies the number of packets in the batch and the size of data payload. In case the data is compressed, the client is required to decompress the data payload using LZO decompression algorithm.

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. Session Messages

3.1 Heartbeat Message (Sent by server)

Heartbeat message will be sent if data is not available.

Field Name	Data Type	Value	Remark
INFO HEADER			
Code	SHORT	'CH'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0 (Zero)
INFO DATA			
Not associated with any data			
INFO TRAILER			
Checksum	SHORT	Numeric	0 (Zero)
End Of Trailer	CHAR[1]	'\r'	Carriage Return

3.2 End of Feed Message (Sent by server)

End of Feed message will be sent to indicate feed termination.

Field Name	Data Type	Value	Remark
INFO HEADER			
Code	SHORT	'CE'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0 (Zero)
INFO DATA			
Not associated with any data			
INFO TRAILER			
Checksum	SHORT	Numeric	0 (Zero)
End Of Trailer	CHAR[1]	'\r'	Carriage Return

4. Sequenced Data Messages (Sent by server)

Sequenced data messages will be sent by server and will contain the actual market data.

4.1 Online Trade Information

NSE-CBRICS Trade information is sent through this message.

Field Name	Data Type	Value	Remark
INFO HEADER			
Code	SHORT	'CX'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0 (Zero)
INFO DATA			
Time Stamp	CHAR[11]	Character	No of seconds from 01-01-1970 00:00:00 (DD-MM-YYYY HH:MM:SS)
Message Code	CHAR[1]	Character	'L' = OTC listed Bonds 'U' = OTC unlisted Bonds
ISIN	CHAR[12]	Character	ISIN Code of the Bond
Descriptor	CHAR[128]	Character	Descriptor of the Bond
Weighted Average Price (Rupees)	CHAR[24]	Character	Weighted Average Price in Rupees
Weighted Average Yield (YTM %)	CHAR[24]	Character	Weighted Average Yield (YTM) - Percent
No. of Trades	CHAR[24]	Character	No. of Trades
Total Trade Value (Rupees)	CHAR[24]	Character	Total Trade Value in Rupees
Last Trade Price (Rupees)	CHAR[24]	Character	Last Trade Price in Rupees
Last Trade Yield (YTM, Annualized, %)	CHAR[24]	Character	Last Trade Yield (YTM, Annualized) - Percent
Source	CHAR[1]	Character	'1' = NSE CBRICS Reporting '5' = RFQ

Seller Deal Type	CHAR[1]	Character	'D' = Direct 'B' = Brokered 'I' = Inter scheme Transfer
ISIN Category	CHAR[2]	Character	'CB' = Corporate Bond 'CP' = Commercial Paper 'CD' = Certificate of Deposit 'SD' = Securitized Debt 'GS' = GSEC / TBILL / SL
Buyer Deal Type	CHAR [1]	Character	'D' = Direct 'B' = Brokered 'I' = Inter scheme Transfer
INFO TRAILER			
Checksum	SHORT	Numeric	Refer to section checksum calculation
End Of Trailer	CHAR[1]	'\r'	Carriage Return

5. Steps for decompressing the data packets.

5.1 LZO Algorithm Details

The LZO stands for Lempel Ziv Oberhaumer. It is a data compression library which is suitable for data Decompression 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. This algorithm is freely available on the internet (URL: https://www.oberhumer.com/opensource/lzo/). It is made available by free software foundation. The algorithm is tested on various operating systems like UNIX and Red Hat Linux.

LZO implements several algorithms with the following features

- 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 LZO Algorithm Details

- 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 De-compressed.
The second field is the number of packets 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_INFO_HEADER *pstInfoHeader = NULL;

for (i=0; i < iNoOfPackets; i++)    // iNoOfPackets received in
// Compressed data header
{
   pstInfoHeader = (ST_INFO_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 is 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. FAQs

1) For Sequenced Data Messages, why do fields contain datatype as short, but contain value is specified as character?

   Data sent by server contains number, which is the ASCII value of the field and at client's end it needs to be converted from ASCII value into character.

2) How to differentiate between numeric and non-numeric values?

   Numeric values are always right aligned and non-numeric values are left aligned. For instance, even though LTP has a datatype as character, it is distinguished by the alignment as numeric value is always right aligned.

8. Contact Information

Name	Email	Contact Number
Business & Technical Support	marketdata@nse.co.in	+91-22-26598385