TECHNICAL DOCUMENT

Market Data Platform – Real Time

REAL TIME DATA
Corporate Bond Market

(STANDARD PRODUCT)

Version 1.1

12 OCT 2015

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

COPYRIGHT NOTICE

Revision History

Name	Description	Date
Version 1.0	Final Specification Issued	August 28, 2015
Version 1.1	Trade structure revised	October 12, 2015

Contents

CORPORATE BOND MARKET – REAL TIME DATA (STANDARD PRODUCT)

1. Introduction

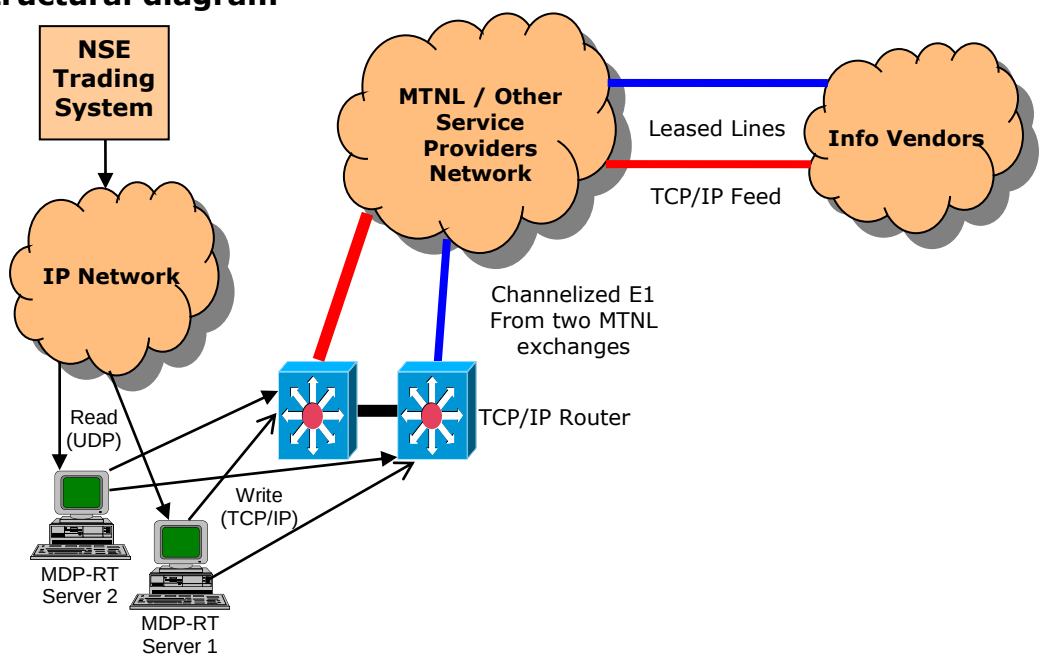
DotEx International Ltd. disseminates NSEIL's Real Time Broadcast data to various information agencies. It provides the 3 different types of data to Info Vendors, i.e. Real Time Data, Snapshot Data and End of Day Data. The Real Time Data is a packet broadcast available in TCP/IP packet format, whereas the Snapshot Data and End of day data are available in the form of files. Certain products based on the Real Time Data are also made available through files.

The information agencies connect to the MDP-RT Server through Leased Lines. These leased lines are terminated on MDP-RT Router and their data specific pneumonic calls are forwarded to MDP-RT server. The MDP-RT server accepts these pneumonic calls and creates a socket connection. The TCP/IP data flows to the information agencies through these socket connections.

2. Connection Details

The Info Vendors connect to Real Time server over Leased Lines using the proprietary protocol specified in this document. In NSE premises two Real Time Production Servers operate in an active-active configuration. Each server can be accessed using their IP addresses as shown in the Structural Diagram. In case a server becomes inaccessible through both its IP addresses, the Info Vendor software requires to fail-over to the other server.

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 MDP-RT 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 MDP-RT servers. Connectivity through internet is available for some products only.

2.3 Platform notes

1. The service can be simultaneously accessed through IP addresses on each server. This is to enable Info Vendors to access the servers in case of failure.
2. There may be slight differences between the data disseminated by the two servers because of differences in routing of data, etc.
3. The feed consist of series of sequenced and un-sequenced variable length compressed messages.
4. The compression algorithm used over here is LZO – Compression.
5. Following are sizes of Data Types referred throughout this document:

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

6. Byte order (Endian-ness): All values are Little Endian.
7. Byte alignment (Structure Packing): All structures are aligned to 1 byte boundary.

3. Product Overview

This document explains about 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.

4. Session Initialization

The Feed is built on TCP/IP socket connection. This feed consists of sequenced and un-sequenced messages. Un-sequenced messages provides the login and connection related messages such as login and heartbeats messages. Un-sequenced 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.

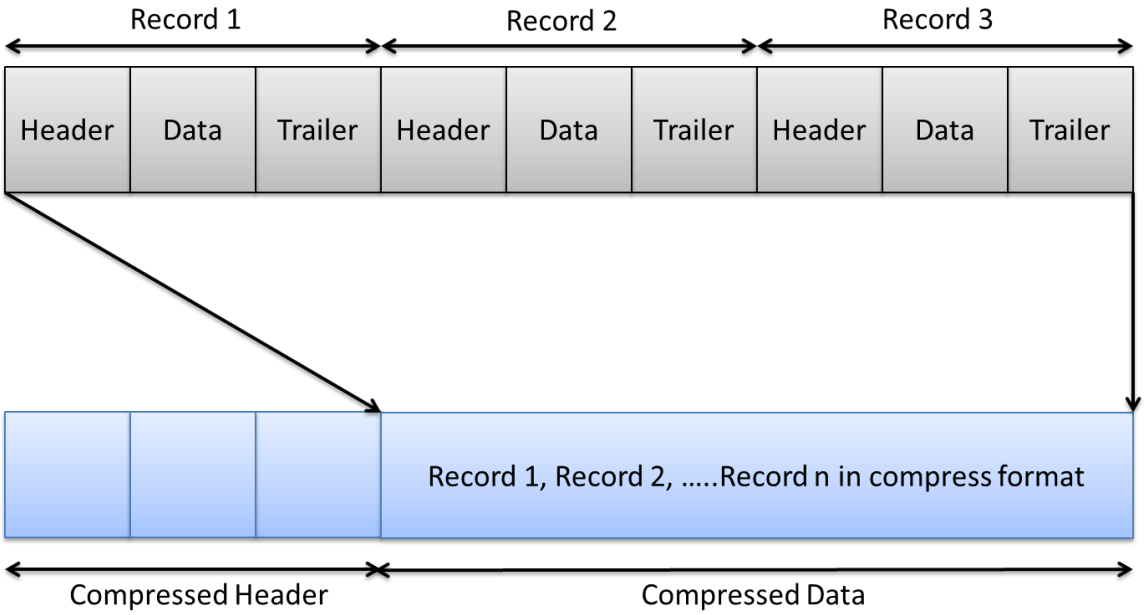
Data packet consist of sequence number as one field. The first sequenced message of the day will send the sequence number as 1 and after that it will be incremented by 1 for each sequenced message.

5. Packet Format

The data is sent out over the TCP/IP stream in batches. Each batch received on the TCP stream is organized as follows:

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.

6. 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.

6.1 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	'CQ'	
Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0 (Zero)
INFO DATA			
User Id	CHAR[10]	Alphanumeric	Exchange provided user id (Null terminated)
Password	CHAR[8]	Alphanumeric	Password (Null terminated)
New Password	CHAR[8]	Alphanumeric	New password (Null terminated)
Confirm Password	CHAR[8]	Alphanumeric	Confirm password (Null terminated)
INFO TRAILER			
Check Sum	SHORT	Numeric	Refer Section on "Checksum Calculation Algorithm"
End Of Trailer	CHAR	'\r'	Carriage Return

6.2 Login Response (Sent by server)

Login response packet will be sent by server after receiving the login request packet. This packet does 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	'CR'	

Length	SHORT	Numeric	Size of (INFO HEADER + INFO DATA + INFO TRAILER)
Sequence Number	LONG	Numeric	0 (Zero)
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. 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 Section on "Checksum Calculation Algorithm"
End Of Trailer	CHAR	'\r'	Carriage Return

6.3 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			
Check Sum	SHORT	Numeric	0 (Zero)
End Of Trailer	CHAR	'\r'	Carriage Return

6.4 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			
Check Sum	SHORT	Numeric	0 (Zero)
End Of Trailer	CHAR	''\r'	Carriage Return

7. Sequenced Data Messages (Sent by server)

Sequenced data messages will be sent by server and will contain the actual market data. These messages have a sequence number assigned for each data message

7.1 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	Application sequence number
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 in Lacs)	CHAR[24]	Character	Total Trade Value in Rupees in Lacs
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
INFO TRAILER			
Check Sum	SHORT	Numeric	Refer Section on "Checksum Calculation Algorithm"
End Of Trailer	CHAR	'\r'	Carriage Return

8. Steps for decompressing the data packets

8.1 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.

8.2 LZO Algorithm Details

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

8.3 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_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;
          }
   }
}
```

9. 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);
}
```

10. Contact Information

Name	Email	Contact Number
DOTEX Business	dotex@nse.co.in	91-22-26598385
Technical Support	infofeed_support@nse.co.in	91-22-26598270