MARKET FEED
CM, FAO & CD
TICK BY TICK FEED

Version: 5.5
Date: 12 August, 2015

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

Name	Description	Date
Version 5.0	Final Specification Issued	28 November, 2013
Version 5.1	Following sections added: 5.5. Contract Information File Details 5.6. Spread Contract Information File Details	23 December, 2013
Version 5.2	3. Regarding Bucket Functionality 5.7. Bucket Contract Information File Details	20 June, 2014
Version 5.3	11. Tuning guidelines	19 August, 2014
Version 5.4	Capital Market segment related information added.	11 September, 2014
Version 5.5	Following section revised: 10. Miscellaneous, Point 11	12 August, 2015

NSE – NSEMD Feed

1. Introduction

The NSEMD Feed is the Tick By Tick data feed of NSE. It disseminates information about orders and trades on a real time basis. The feed consists of series of sequenced variable length messages. NSEMD Feed data is sent to clients on a connection-less UDP Multicast.

2. Session Details

The Tick Data feed is available as separate Multicast Channels for separate Streams. The contracts are distributed across the streams. For eg. Contracts 1-4 will be on stream 1, contracts 5-8 on stream 2 and so on.

A Data packet consists of a Header which has Stream ID and Sequence number fields. The sequence Number for the first message of the day for the stream will be sent as 1. After that, it will be incremented by 1 for each consecutive message for the Stream.

The feed also consists of Masters Information which is sent only once, in the morning at 8.30 am. The same information is also available on file and can be downloaded any time during the day.
Masters Information provides contract descriptor information about all the contracts available through different Streams and also provides the mapping of a contract to a particular Stream.

3. Exchange Defined Bucket Feature

Exchange Defined Buckets are streams carrying data for grouped contracts.

Each Bucket is differentiated by its own unique Stream ID. The Stream IDs for Buckets are different from the main streams.

The Bucket Contract Information File contains the contracts to bucket association. The Bucket Spread Contract Information File contains the spread contracts to bucket association. The Masters information for bucket contracts is not available over multicast. It is only available through file.

Message formats for Bucket Streams are same as that of normal streams. Online data for Buckets, just like other streams, is transmitted over its own unique Multicast Group.

Recovery for Buckets, just like other streams, is available through the Recovery Server.

It is possible that the same contract / spread contract may be present across multiple buckets.

This feature is currently available in FO and CD segments.

4. Data Type

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

Byte order - Little Endian.
Pragma Pack – 1.

5. Packet Format

Server sends all the packets in following format:

Stream Header:

```
typedef struct
{
        short  msg_len;
        short  stream_id;
        int     seq_no;
}STREAM_HEADER;
```

Stream Data:

```
typedef struct
{
        char   cMsgType;
        .
        .
        .
}STREAM_DATA;
```

Each Data Packet consists of a Header and Data.

The Header (STREAM_HEADER) consists of following fields:

1. Message Length : This is the total size of the packet including Header and Data i.e. sum of length of STREAM_HEADER and STREAM_DATA.
2. Stream Id : This identifies a particular stream
3. Sequence No : This is the sequence number of the packet for a particular Stream Id

Each Data Packet, i.e. STREAM_DATA has Message Type as the first field. Always, the first field should be read first and interpreted. Depending on the value of this field the Data Packet should be mapped to relevant message structure.

6. Masters Data Details

To be able to interpret the data feed, the Client requires Masters Data.

For Masters Information there is a header and trailer packet indicating start and end of transmission

The data contains contract information such as Token, Symbol, Instrument, Expiry Date, Strike Price and Option Type. This data also provides the mapping of Contract Information to Stream ID ie. the Stream in which the contract is available.

Masters Data is sent as a separate Stream only once in the day at 8.30 am in the morning.

All the Order and Trade Messages have only the Token Field for representing the contract. The Client Application must load the Masters Data for the day before receiving Orders and Trades. Otherwise the Client Application will be working on wrong information.

6.1. Masters Data Header

Masters Data Header packet is sent first.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'B'	Begin of Masters Data Transmission
Token Count	INT	Numeric	No. of Contract Information Packets

6.2. Masters Data Contract Information Message

Contract Information Messages are sent after the header, containing contract information for all contracts available in the market.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'C'	Contract Information
Stream ID	SHORT	Numeric	Availability of this contract on a particular Stream
Token Number	INT	Numeric	Unique identifier for contract
Instrument	CHAR[6]	Alphanumeric	FO & CD: Security Instrument. The instrument received is right padded with blanks and is not NULL terminated. Eg. FUTIDX CM: Set to "EQUITY".
Symbol	CHAR[10]	Alphanumeric	Security Symbol. The symbol received is right padded with blanks and is not NULL terminated.
Expiry Date	INT	Numeric	FO & CD: Expiry date of contract in seconds from 01-Jan-1980 00:00:00 CM: Not used, set to 0.
Strike Price	INT	Numeric	FO & CD: Strike Price Of Contract (In Paise). For FO segment this should be divided by 100 for converting into Rupees.

			For CD segment this should be divided by 10^7 for converting into Rupees. This will be zero in case of futures contract. CM: Not used, set to 0.
Option Type	CHAR[2]	Alphanumeric	FO & CD: Option Type for the contract CM: Series of the security.

6.3. Masters Data Spread Contract Information Message

FO and CD segments only:

Spread Contract Information Messages are sent, containing spread contract combinations available in the market.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'P'	Spread Contract information
Stream ID	SHORT	Numeric	Availability of this spread contract on a particular Stream
Token 1	INT	Numeric	Token No. of First Spread Contract
Token 2	INT	Numeric	Token No. of Second Spread Contract

6.4. Masters Data Trailer Message

Masters Data Trailer Message is sent after the contract information messages indicating end of transmission of information for all contracts.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'E'	End Of Transmission Message
Token Count	INT	Numeric	No. of Contract Information Packets

6.5. Contract Information File Details

This is a plain-text CSV file containing information for normal contracts for the trading day.

Nomenclature:
FO Segment:
fo_contract_stream_info.csv
CD Segment:
cd_contract_stream_info.csv
CM Segment:
cm_contract_stream_info.csv

File Format:
Plain Text, CSV

Field delimiter:
Comma (,)

Header Record:
This is the first line in the file. It contains the file generation timestamp along with number of contracts in the file, each followed by a comma.

Field Name	Data Type	Value	Remark
Date	INT	Numeric	File generation time in seconds from 01-Jan-1980 00:00:00
No. of contracts	INT	Numeric	Number of contract records

Contract Information Records:
The fields of the subsequent lines should be interpreted as follows. Each field is followed by a comma.

Field Name	Data Type	Value	Remark
Message Type	CHAR	'C'	Contract Information
Stream ID	SHORT	Numeric	Availability of this contract on a particular Stream

Token Number	INT	Numeric	Unique identifier for contract
Instrument	CHAR[6]	Alphanumeric	FO & CD: Security Instrument. Eg. FUTIDX CM: Set to "EQUITY".
Symbol	CHAR[10]	Alphanumeric	Security Symbol
Expiry Date	INT	Numeric	FO & CD: Expiry date of contract in seconds from 01-Jan-1980 00:00:00 CM: Not used, set to 0.
Strike Price	INT	Numeric	FO & CD: Strike Price Of Contract (In Paise). For FO segment this should be divided by 100 for converting into Rupees. For CD segment this should be divided by 10^7 for converting into Rupees. This will be zero in case of futures contract. CM: Not used, set to 0.
Option Type	CHAR[2]	Alphanumeric	FO & CD Option Type for the contract CM: Series of the security.

6.6. Spread Contract Information File Details

FO and CD segments only:

This is a plain-text CSV file containing information for spread contracts for the trading day.

Nomenclature:
FO Segment:
fo_spd_contract_stream_info.csv
CD Segment:
cd_spd_contract_stream_info.csv

File Format:
Plain Text, CSV

Field delimiter:
Comma (,)

Header Record:
This is the first line in the file. It contains the file generation timestamp along with number of contracts in the file, each followed by a comma.

Field Name	Data Type	Value	Remark
Date	INT	Numeric	File generation time in seconds from 01-Jan-1980 00:00:00
No. of spread contracts	INT	Numeric	Number of spread contract records

Spread Contract Information Records:
The fields of the subsequent lines should be interpreted as follows. Each field is followed by a comma.

Field Name	Data Type	Value	Remark
Message Type	CHAR	'P'	Spread Contract information
Stream ID	SHORT	Numeric	Availability of this spread contract on a

			particular Stream
Token 1	INT	Numeric	Token No. of First Spread Contract
Token 2	INT	Numeric	Token No. of Second Spread Contract

6.7. Bucket Contract Information File Details

FO and CD segments only:

This is a plain-text CSV file containing information for normal contracts for the trading day for various bucket streams.

Nomenclature:
FO Segment:
fo_bkt_contract_stream_info.csv
CD Segment:
cd_bkt_contract_stream_info.csv

File Format:
Plain Text, CSV

Field delimiter:
Comma (,)

Header Record:
This is same as Contract Information File.
The field "Stream ID" contains the Stream ID for the bucket.

Contract Information Records
This is same as Contract Information File.
The same contract token may be repeated in the file, as a contract may be present in more than one bucket.

6.8. Bucket Spread Contract Information File Details

FO and CD segments only:

This is a plain-text CSV file containing information for spread contracts for the trading day for various bucket streams.

Nomenclature:
FO Segment:
fo_bkt_spd_contract_stream_info.csv
CD Segment:
cd_bkt_spd_contract_stream_info.csv

File Format:
Plain Text, CSV

Field delimiter:
Comma (,)

Header Record:
This is same as Spread Contract Information File.
The field "Stream ID" contains the Stream ID for the bucket.

Contract Information Records
This is same as Spread Contract Information File.
The same spread contract token pair may be repeated in the file, as a spread contract may be present in more than one bucket.

7. Sequenced Data Messages (Market Data)

Sequenced data messages will be sent by server on UDP Multicast and will contain market data. These messages will contain normal market, regular lot and spread data.

7.1. Order Message

For every new order request, modification request, cancellation request, this message is sent. All new order requests will be sent including the active orders.
Modifications and cancellations must be handled at the client end on basis of Order Id and not on Price.

Currently Stop Loss Orders are not sent to clients. However when Stop Loss Order gets modified to Regular Lot Order or Regular Lot Order gets modified to Stop Loss Order it leads to following scenarios:
It is possible that for an Order Cancellation Message the original Order Id is not found by the Client Application. In that case the particular Order Cancel message should be ignored by the Client application
It is possible that for an Order Modification Message the original Order Id is not found by the Client Application. In that case the particular Order Modification Message should be treated as a new Order and handled accordingly.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'N' or 'X' or 'M'	'N' - New Order 'M' - Order Modification 'X' - Order Cancellation
Timestamp	LONG	Numeric	Time in milliseconds from 01-Jan-1980 00:00:00
Order ID	DOUBLE	Numeric	Day Unique Order Reference Number
Token	INT	Numeric	Unique Contract Identifier
Order Type	CHAR	Character	'B' - Buy Order

			'S' - Sell Order
Price	INT	Numeric	Price of the order (In Paise) This field contains the price at which the order is placed. The price is in multiples of the tick size. For FO and CM segments this should be divided by 100 for converting into Rupees. For CD segment this should be divided by 10^7 for converting into Rupees.
Quantity	INT	Numeric	Quantity of the order

7.2. Trade Message

This message is sent whenever an order in the order book gets executed fully or partially.
It is possible that either Buy Order ID or Sell Order ID can have value as Zero. In such a case that Order ID should be ignored and there should not be any follow up action based on that Order ID. Both the Order Ids will not have the value zero at the same time.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'T'	'T' = Trade Message
Timestamp	LONG	Numeric	Time in milliseconds from 01-Jan-1980 00:00:00
Buy Order ID	DOUBLE	Numeric	Day Unique Order Reference Number for Buy-Side Order
Sell Order	DOUBLE	Numeric	Day Unique Order

ID			Reference Number for Sell-Side Order
Token	INT	Numeric	Unique Contract Identifier
Trade Price	INT	Numeric	Trade Price (In Paise) This field contains the price at which the trade took place. The price is in multiples of the tick size. For FO and CM segments this should be divided by 100 for converting into Rupees. For CD segment this should be divided by 10^7 for converting into Rupees.
Trade Quantity	INT	Numeric	Trade Quantity

7.3. Spread Order Message

FO and CD segments only:

For every new spread order request, modification request, cancellation request, this message is sent. All new spread order requests will be sent including the active orders.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'G' or 'H' or 'J'	'G' - New Spread Order 'H' – Spread Order Modification 'J' – Spread Order Cancellation
Timestamp	LONG	Numeric	Time in milliseconds from 01-Jan-1980 00:00:00
Order ID	DOUBLE	Numeric	Day Unique Order

			Reference Number
Token	INT	Numeric	Unique Contract Identifier
Order Type	CHAR	Character	'B' - Buy Order 'S' - Sell Order
Price	INT	Numeric	Price of the order (In Paise) This field contains the price difference for this spread. The price is in multiples of the tick size. For FO segment this should be divided by 100 for converting into Rupees. For CD segment this should be divided by 10^7 for converting into Rupees.
Quantity	INT	Numeric	Quantity of the spread order

7.4. Spread Trade Message

FO and CD segments only:

This message is sent whenever a spread order in the order book gets executed fully or partially. It is possible that Buy Order ID or Sell Order ID can have value as Zero. In such a case that Order ID should be ignored and there should not be any follow up action based on that Order ID.

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'K'	'K' = Spread Trade Message
Timestamp	LONG	Numeric	Time in milliseconds from 01-Jan-1980 00:00:00

Buy Order ID	DOUBLE	Numeric	Day Unique Order Reference Number for Buy-Side Order
Sell Order ID	DOUBLE	Numeric	Day Unique Order Reference Number for Sell-Side Order
Token	INT	Numeric	Unique Contract Identifier
Trade Price	INT	Numeric	Trade Price (In Paise) This field contains the price at which the trade took place. The price is in multiples of the tick size. For FO segment this should be divided by 100 for converting into Rupees. For CD segment this should be divided by 10^7 for converting into Rupees.
Quantity	INT	Numeric	Quantity of the spread trade

8. Recovery Details

A Tick Agent Application is provided which is to be installed at Client End Server for receiving the Multicast Tick Data. Orders and Trades Tick Data is forwarded to end client CTCL applications on Multicast by the Tick Agent.

The Tick Data feed is available on two separate Multicast Channels for each Stream on active-active basis. For eg. if there are four streams of multicast feeds, then no. of multicast channels are eight (two for each stream)

Separate instances of Tick Agent application is to be executed for each multicast channel.

Tick Agent also provides Tick Data recovery and it is the sole contact point for recovery of missed Tick Data for a particular multicast stream.

If the CTCL application misses any tick data it can recover the ticks in the following manner

1. It can receive the ticks from both the Multicast Channels in Active-Active manner. If tick is missed in one channel it can receive the tick from the other channel.

2. If the ticks are missed from both the channels it can request the Tick Agent for the missed tick. This request is on TCP. The Tick Agent will provide the missed tick data on TCP.

8.1. Tick Data Recovery Request Message

Whenever tick data is missed this message needs to be sent to Tick Agent on TCP. No header is required for this message. In case only one tick is missed Start Sequence No. and End Sequence No. should be same.

A Tick Data Recovery Response Message is sent first by the Tick Agent to indicate whether this request will be processed successfully by Tick Agent.
If the Tick Recovery is successful then rest of the response messages are provided on TCP and format is same as mentioned in section 4.

If the Tick Recovery Request is unsuccessful, Member Application can send the request again.

Field Name	Data Type	Value	Remark
Message Type	CHAR	'R'	Recovery Request
Stream ID	SHORT	Numeric	Unique Stream Reference
Start Sequence Number	INT	Numeric	Requested Message Sequence Number
End Sequence Number	INT	Numeric	Requested Message Sequence Number

8.2. Tick Data Recovery Response Message

This is the first message sent as response by Tick Agent on TCP to indicate whether Tick Data recovery request has been processed successfully by Tick Agent.
For this message, the sequence number field in Global Header will have the value 0 (zero).

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'Y'	Recovery Response
Request Status	CHAR	'S' or 'E'	'S' – Success 'E' – Error

9. Heartbeat Message

Heartbeat message will be sent when there is no data available for a few seconds. These messages will be available on all Streams.

In case of Heartbeat messages, the sequence number field in Global Header will have the value 0 (zero).

Field Name	Data Type	Value	Remark
Global Header	STREAM_HEADER	Header Data	Refer "Global Header" definition in Section 4.
Message Type	CHAR	'Z'	Heartbeat message
Last Sequence No.	INT	Numeric	Last sent data sequence no. of the Stream

10. Miscellaneous

1. What happens to existing TBT Feed On TCP?

The existing TCP TBT Feed will be available.

2. Why is TBT Feed being provided on Multicast?

Multicast is another type of feed that is being made available as per member requirements, international standards and it provides an additional option for members to use it if they are ready and find it useful.

3. Multicast can have problems of data packet loss or non sequential delivery. How are these problems addressed?

Exchange shall publish multicast data feed for each market engine.
There will be 2 streams for each engine, and members can subscribe to both the streams if they so desire so as to enable them to be able to receive data on a live basis. Both the streams are same and follow the same sequence. This is the first point of failover that members can rely on, i.e. by subscribing to 2 streams for each engine.
Exchange provides a TBT Agent which is to be installed at member end to listen and receive feed. From this TBT Agent the data can be fed to

member internal systems for further processing and use.
If there are any missing ticks the request can be sent to this TBT Agent and data retrieved. Members can configure their buffer size so as to retain data in the TBT Agent.

A modified TCP TBT similar to the multicast will also be made available. All of these are in addition to the existing TCP TBT data feed.
Members will have a whole suite of TBT data options that they can choose to access and use based on their readiness.

4. Old Price and Quantity is not being provided as part of Order Modification and Cancellation. How to handle Order Modification and Cancellation ?

All Order Messages are provided with Order Id data. Tracking of Orders need to be done by Member Applications based on Order id. Whenever a Order Modification message is received the previous Order Image needs to be looked up based on Order Id and accordingly adjusted with the Modified Order Message.
Same behaviour for Order Cancellation Message.

5. Are active or aggressive Orders being sent as part of Order Messages ?

Yes, active or aggressive orders are also being sent as part of Order Messages.

6. Is there any specific handling required for DQ functionality ?

DQ functionality will be handled at Exchange End. For Member End Applications there is no specific handling required.

7. The multicast feed specifications are applicable for which market segments ?

The specifications are applicable for all NSE market segments.
Deployment will be done in CD segment first followed by FO and then CM

8. Is Order Id same as interactive Order Number ?

Yes, Order id is same as interactive Order Number.

9. Trade Message has an Order Id which is not present in client Order Book. Is this scenario possible ?

Yes, this scenario is possible as market orders are not being sent. In such scenario the particular Order Id should be ignored by the client application.

10. Order Cancel message has a price or quantity which is different from the original order. Is this possible ?

Yes, this scenario is possible. For Order Cancel messages the client application should consider only the Order Id field and accordingly remove the original order from the order book.

11. Is there a limit to the maximum number of messages which can be recovered in a single session from the Exchange Recovery Servers ?

Yes. This is currently 3,00,000 messages. More messages can be recovered through subsequent requests.

12. Is there any handling required for pre-open orders in CM segment?

No specific handling is required for pre-open orders at the Member end. Pre-open orders will not be sent out on the MTBT Feed in the Order Collection phase of Pre-open Session. The Trades occurring in the Matching Phase of the Pre-open Session will be sent out on the feed. The Pending Orders which will be carried forward to the Normal Market Session will be sent out as normal orders after the Pre-open Session ends.

11. Tuning guidelines

11.1. NIC Recommendations

The minimum value of the Max Receive Buffer size on the NICs which are going to be used for receiving the TBT Multicast should be 4 KB. NICs with the specified value below the recommended value may not be able to handle the incoming data bursts, resulting in UDP packet drops. In case of teamed NICs (NIC Bonds), this is applicable for both the underlying physical NICs.

Where: This is meant for the machine which runs Tick Agent AND the machine which runs Member application which subscribes to the TBT Multicast.

11.2. Increase the Kernel Receiver Backlog

To find out the current value, as root, execute the following command:

```
sysctl -a | grep netdev_max_backlog
```

```
/tbtdv1/users/tbtdv> sysctl -a | grep netdev_max_backlog
net.core.netdev_max_backlog = 1000
```

Make a note of the value of this parameter.

If it is lower than 50000, as root, execute the following command to increase it as follows:

```
/sbin/sysctl -w sys.net.core.netdev_max_backlog=50000
```

Verify whether the parameter has been set as desired:

```
/tbtdv1/users/tbtdv> sysctl -a | grep netdev_max_backlog
net.core.netdev_max_backlog = 50000
```

Where: This should be done both on the machine which runs Tick Agent AND the one which runs Member application which subscribes to the TBT Multicast.

11.3. Increase the NIC Ring Buffer Receive size on NICs:

As root, execute the following command for all of the NICs on the system which are used for receiving TBT Multicast data:

```
ethtool -g ethX
```

.. where X corresponds to the NIC number (run the "ifconfig" command to list NICs on your system).

```
[idxmgr@indexdev1 ~]$ ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:             4078
RX Mini:        0
RX Jumbo:       0
TX:             511
Current hardware settings:
RX:             200
RX Mini:        0
RX Jumbo:       0
TX:             511
```

Make a note of the following values:
a -> "RX" parameter displayed in the "Pre-set maximums" (current receive buffer size).
b -> "RX" parameter displayed in the "Current hardware settings" (maximum receive buffer size which can be set).

If "b" is lower than "a", as root, execute the following command to increase such that it is set to the max:

```
ethtool -G ethX rx a
```

.. where X corresponds to the NIC number and 'a' corresponds to maximum value that can be set for the RX parameter.

In case of teamed NICs (NIC Bonds), this tuning should be performed for the underlying physical NICs.

Verify whether the parameter has been set as desired:

```
[idxmgr@indexdev1 ~]$ ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:             4078
RX Mini:        0
RX Jumbo:       0
TX:             511
Current hardware settings:
RX:             4078
RX Mini:        0
RX Jumbo:       0
TX:             511
```

Where: This should be done both on the machine which runs Tick Agent AND the one which runs Member application which subscribes to the TBT Multicast.

11.4. Increase the OS Send and Receive Buffers

As root, execute the following commands:

```
sysctl -a | grep rmem_max
```

```
/tbtdv1/users/tbtdv> sysctl -a | grep rmem_max
net.core.rmem_max = 4669986
```

```
sysctl -a | grep tcp_mem
```

```
/tbtdv1/users/tbtdv> sysctl -a | grep tcp_mem
net.ipv4.tcp_mem = 196608  262144  4669986
```

Make a note of the value of these parameters.

If the parameter rmem_max and third parameter of tcp_mem are lower than 134217728, as root, execute the following command to increase them to 134217728 accordingly:

```
/sbin/sysctl -w net.core.rmem_max=134217728
/sbin/sysctl -w net.ipv4.tcp_mem=10240 87380 134217728
```

Verify whether the parameter has been set as desired:

```
/tbtdv1/users/tbtdv> sysctl -a | grep rmem_max
net.core.rmem_max = 134217728
/tbtdv1/users/tbtdv> sysctl -a | grep tcp_mem
net.ipv4.tcp_mem = 196608  262144  134217728
```

(The first two values of tcp_mem shown above may differ with Member's machine.)

System-specific consideration:

The network read buffer of OS is being set to a significantly high value here (134 MB). This will become applicable system-wide, which means that all applications running on the system will be able to request this amount of OS network read buffer size from OS. Tune as per your available RAM.

Where: This should be done both on the machine which runs Tick Agent AND the one which runs Member application which subscribes to the TBT Multicast.

11.5. Increase the Application Send and Receive Buffers

Ensure that large network Receive Buffer is requested from the OS through the Member application code on all the sockets used to receive UDP traffic.

```
int nTCPRecvBufSize = 134217728;
if(0 != setsockopt(nFD, SOL_SOCKET, SO_RCVBUF, &nTCPRecvBufSize,
sizeof(nTCPRecvBufSize))
```

```
{
      // This means that the call did not go through.
      // Log this error and take necessary action.
}
```

Where: This must be done in the Member application just after creation of the socket(s) used to receive the TBT Multicast.

11.6. Buffered Message Count in Tick Agent

For Tick Agent, the "Buffered Message Count" value must be set to 134217728 (Refer the Tick Agent User Manual for details on how to set this parameter).

Where: This should be done on the machine which runs Tick Agent.

11.7. Ensure that Multicast IP is also provided to "bind"

In Member software, when populating the "sockaddr_in" structure which is passed to "bind" system call, ensure that the Multicast IP Address is populated as well when populating the port. On systems which subscribe to multicast from both markets on same machine, this will result in multicast from both the markets to be received on the same socket, resulting in data issues.

```
int                     nRetVal = 0;
struct sockaddr_in      stLocalAddr;
memset(&stLocalAddr, 0, sizeof(stLocalAddr));

stLocalAddr.sin_family          = AF_INET;
stLocalAddr.sin_port            = htons(pstStreamAttr->nMulticastPort);
stLocalAddr.sin_addr.s_addr     = inet_addr(pstStreamAttr->szMulticastIP);

nRetVal = bind(pstStreamAttr->nRecvSockFD, (struct sockaddr *)&stLocalAddr,
                sizeof(stLocalAddr));
if(nRetVal != 0)
{
        nRetVal = errno;
        LOG_FATAL("Error in creating client socket <%s>", strerror(nRetVal));
        return FAILURE;
}
```

Where: This must be done in Member application when it binds to the Multicast Port before adding the Membership Request for receiving the TBT Multicast.

11.8. Disclaimer

The methods provided in the Tuning Guidelines are in an effort to try and help Members tune their systems. Members must take into careful consideration the load on their systems, availability of system resources and their specific use-cases before embarking on any system tuning exercise. The Exchange does not officially mandate the use of any of the tuning methods mentioned here and is not responsible for the results of any tuning exercise carried out by the Member at their end. Currently these guidelines are available for Linux systems only.

Support Information

For any further assistance please find below the support details.

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