MARKET FEED
CURRENCY DERIVATIVES

SNAPSHOT DATA

Version: 1.0

DATE: 20 OCTOBER, 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 1.0	Final Specification Issued	23 April 23, 2015
Version 1.1	1 Minute Snapshot Product introduced	20 October, 2015

Contents

CURRENCY DERIVATIVES MARKET - SNAPSHOT DATA

1. Introduction

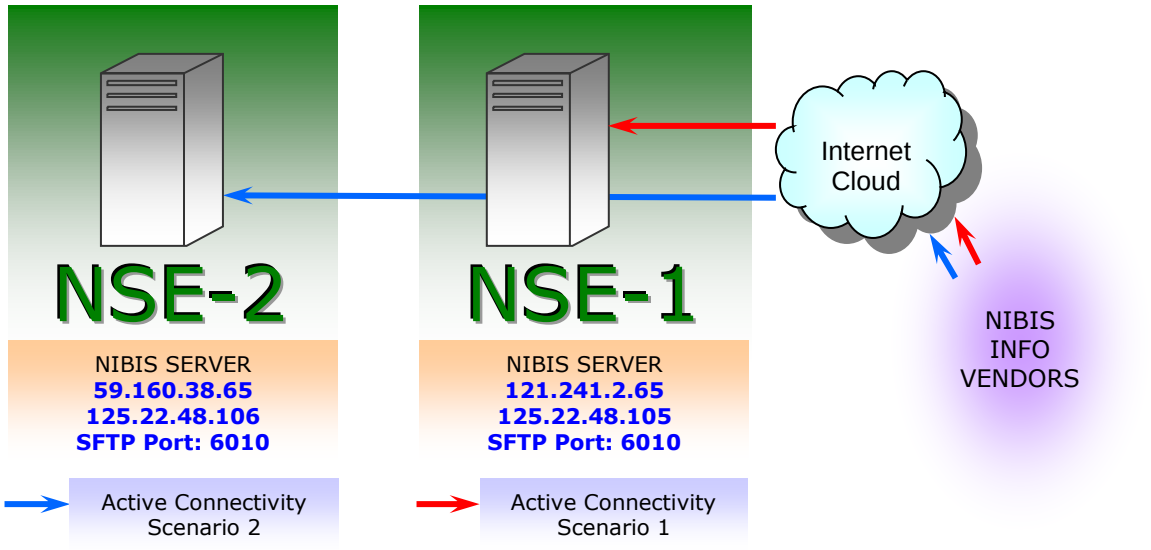
NSE Data & Analytics Ltd. disseminates NSEIL's Real Time Broadcast data to various information agencies. It provides the 3 different types of data to Info Vendors, i.e. Real Time Data, Snapshot Data and End of Day Data. The Real Time Data is a packet broadcast available in TCP/IP packet format, whereas 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 NIBIS (NSE Internet Based Information System) server that caters the NIBIS Info Vendors is available through Internet. All NIBIS Info Vendors connect the server through internet and use SFTP protocol to download the files. The files on this server are generated on regular intervals. The Info Vendors are provided with credentials which are enabled for the agreement period.

2. Connection Details

The Info Vendors connect to NIBIS server over the Internet using SFTP protocol. In NSE premises two NIBIS Production Servers operate in an active-active configuration. Each server can be accessed using two IP addresses, for ISP-level redundancy, 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 Platform notes

1. The SFTP service can be simultaneously accessed through both redundant IP addresses on each server. This is to enable Info Vendors to access the servers in case of link failure.
2. Info Vendors may use both IP addresses of a server during normal course of operations in order to put both available links to use.
3. There may be slight differences between the data disseminated by the two servers because of factors impacting sampling such as CPU clock skew, differences in routing of data, etc.
4. Time stamp on the files on the server is in 24-hour format.
5. Certain files are compressed using ZLIB (gzip). The files may be decompressed using the popular "gunzip" command on Unix/Linux systems. Tools to decompress these files are also freely available for Windows on the World Wide Web, Gzip for Windows and 7-Zip being popular examples.
6. The Exchange does not provide software or support for decompression, SFTP, etc.

3. Overview

3.1 Products and "Product Root"

The files are productized as per the generation frequency and are generated under their designated Product Roots on the server.

Product Root is the name of the top-level directory under which files for a product are generated.

Snapshot Frequency	Generation Frequency	Product Root
1 Minute Snapshot Product	Every 1 Minutes	/CDM01
5 Minute Snapshot Product	Every 5 Minutes	/CDM05

The Product Roots may further contain subdirectories as specified in the relevant sections of this document.

Info Vendors may subscribe to product of their choice depending on their data snapshot frequency requirement.

3.2 Types of files generated

The files are generated in binary format on the servers inside the corresponding type-wise sub-directories as specified in this document and can be broadly classified as follows:

Description	Frequency
Market Information Files	At a specific interval
Contract Information Files	Once a day (EOD)
Bhavcopy Information Files	Once a day (EOD)

For each trading day, files are generated in date-wise sub-directories prefixed with the full month name (MonthDDYYYY) as specified in the relevant sections of this document.

The files generated at fixed frequency are continuously numbered, starting from 1.

3.3 Compression

Certain files are compressed using ZLIB (gzip). The files may be decompressed using the popular "gunzip" command on Unix/Linux systems. Tools to decompress these files are also freely available for Windows on the World Wide Web, Gzip for Windows and 7-Zip being popular examples. It may be noted that the Exchange does not provide software or support for decompression.

4. Data Details

4.1 Market Information

The Market information data files (MBP and OI) are generated on the server at regular intervals.

4.1.1 Market Files

The *.mkt ("*" stands for a number) files contain market statistics and order information of the contracts that are being traded during the last interval, including their open, high, low and close price. The file contains a single record for every contract that is traded during that file interval. These files are generated during normal trading period i.e. 09:00 hrs. To 17:30 hrs. These files are generated in incremental count number on a trading day starting from 1 (for example, 1.mkt, 2.mkt and so on).

4.1.2 OI Files

The *.oi ("*" stands for a number) files contain the details of open interest information of contracts. These files are generated at regular intervals. These files are generated during normal trading period i.e. 09:00 hrs. to 17:30 hrs.

4.2 Contract Information

The CONTRACT.DAT file is the master file that contains the updated information of all contracts traded on the Exchange. The Info Vendors need to download this file and decode it to resolve the "token number" of required contract. The Token number of each contract is unique.

4.3 Bhavcopy Information

The Bhavcopy Information File is generated at around 18:00 hrs on each trading day. This file contains the End of the Day values of the securities that are traded on that trading day.

5. Data Structure Details

5.1 Market Information

Directory Path	/<Product Root>/DATA/<MonthDDYYYY>
File Name	*.mkt
Compression	Compressed (.gz)
Generation Frequency	At fixed intervals

HEADER

Field	Data Type	Length	Description
Timestamp	Long	4 Bytes	Time when the record is updated
Message Length	Short	2 Bytes	Size of DATA packet
Total Length		6 Bytes	

DATA

Field	Data Type	Length	Description
Security Token	Long	4 Bytes	Unique identifier for contract
Market Type	Char	1 Byte	N-Normal Market
Best Buy Price	Char	17 Bytes	Precision up to 7 decimal places
Best Buy Quantity	Char	12 Bytes	In no of contracts
Best Sell Price	Char	17 Bytes	Precision up to 7 decimal places
Best Sell Quantity	Char	12 Bytes	In no of contracts
Last Traded Price	Char	17 Bytes	Precision up to 7 decimal places
Total Traded Quantity	Char	12 Bytes	In no of contracts
Average Traded Price	Char	17 Bytes	Precision up to 7 decimal places
Security Status	Char	1 Byte	Blank for active and „S“ for suspended
Open Price	Char	17 Bytes	Precision up to 7 decimal places
High Price	Char	17 Bytes	Precision up to 7 decimal places
Low Price	Char	17 Bytes	Precision up to 7 decimal places
Close Price	Char	17 Bytes	Precision up to 7 decimal places
Interval High Price	Char	17 Bytes	Precision up to 7 decimal places
Interval Low Price	Char	17 Bytes	Precision up to 7 decimal places
Interval Open Price	Char	17 Bytes	Precision up to 7 decimal places
Interval Close Price	Char	17 Bytes	Precision up to 7

			decimal places
Interval Total Traded Quantity	Char	12 Bytes	In no of contracts
Total Length		258 Bytes	

5.2 Open Interest Information

Directory Path	/<Product Root>/DATA/<MonthDDYYYY>
File Name	*.oi
Compression	Compressed (.gz)
Generation Frequency	At fixed intervals

HEADER

Field	Data Type	Length	Description
Timestamp	Long	4 Bytes	Time when the record is updated
Message Length	Short	2 Bytes	Size of DATA packet
Total Length		6 Bytes	

DATA

Field	Data Type	Length	Description
Security Token	Long	4 Bytes	Unique identifier for contract
Market Type	Char	1 Byte	Market Type
Open Interest	Char	12 Bytes	In no of contracts
Total Length		54 Bytes	

5.3 Contract Information

Directory Path	/<Product Root>/SECURITY/<MonthDDYYYY>
File Name	CONTRACT.DAT
Compression	Not compressed
Generation Frequency	Once (EOD)

DATA

Field	Data Type	Length	Description
Token Number	Long	4 Bytes	Unique identifier for contract
Instrument Name	Char	7 Bytes	Instrument Name
Symbol	Char	11 Bytes	Symbol
Series	Char	3 Bytes	Series
Expiry Date	Long	4 Bytes	Expiry Date
Strike Price	Long	4 Bytes	Strike Price
Option Type	Char	3 Bytes	Option Type
Issue Start Date	Long	4 Bytes	Issue Start Date
Issue Maturity Date	Long	4 Bytes	Issue Maturity Date
Board Lot Quantity	Long	4 Bytes	Board Lot Quantity
Tick Size	Long	4 Bytes	Tick Size

Security Name	Char	26 Bytes	Security Name
Record Date	Long	4 Bytes	Record Date
Ex Date	Long	4 Bytes	Ex Date
No Delivery Start Date	Long	4 Bytes	No Delivery Start Date
No Delivery End Date	Long	4 Bytes	No Delivery End Date
Book Closure Start Date	Long	4 Bytes	Book Closure Start Date
Book Closure End Date	Long	4 Bytes	Book Closure End Date
Remarks	Char	26 Bytes	Remarks
Total Length		128 Bytes	

5.4 Bhavcopy Information

This data file does not contain the Header field.

Directory Path	/<Product Root>/BHAVCOPY/<MonthDDYYYY>
File Name	CDMMKTSTATYYYYMMDD.TXT
Compression	Not compressed
Generation Frequency	Once (EOD)

DATA

Field	Data Type	Length	Description
Instrument Name	Char	6 Bytes	Instrument Name
Symbol	Char	10 Bytes	Symbol
Expiry Date	Char	11 Bytes	Expiry Date (DD-MM-YYYY)
Strike Price	Char	10 Bytes	Precision up to 2 decimal places
Option Type	Char	2 Bytes	Option Type (FF)
Market Type	Char	1 Byte	N - Normal Market
Opening Price	Char	17 Bytes	Precision up to 7 decimal places
Trade High Price	Char	17 Bytes	Precision up to 7 decimal places
Trade Low Price	Char	17 Bytes	Precision up to 7 decimal places
Closing Price	Char	17 Bytes	Precision up to 7 decimal places
Last Traded Price	Char	17 Bytes	Precision up to 7 decimal places
Previous Close Price	Char	17 Bytes	Precision up to 7 decimal places
Settlement Price	Char	17 Bytes	Precision up to 7 decimal places
Total Traded Quantity	Char	12 Bytes	Total Traded Quantity
Total Traded Value	Char	25 Bytes	Precision up to 2 decimal places
Current Open Interest	Char	10 Bytes	In no of contracts
Change in Open Interest	Char	10 Bytes	In no of contracts

Total Length	216 Bytes	

6. Data Field Details

6.1 Security Token Number

The Security Token numbers uniquely identify each contract listed on the National Stock Exchange of India Ltd. The token number, Instrument, Symbol, Expiry Date, Strike Price and Option Type identify a single and unique contract. The Info Vendor will be provided with a binary file i.e. securities.dat giving the combinations of all contracts traded on the Exchange.

6.2 Time Stamp

The time stamp is the number of seconds elapsed from midnight Jan 1, 1970.

6.3 Price Precision

Prices field will have the precision up to 7th decimal places.

7. Date and Time Conversion

7.1 Sample Program 1

Sample program for converting long date into the DD MM YYYY format.

```
/*********************************************************
Routine Name: DateConv
Synopsis: This routine is responsible for processing a
   Date (LONG) which constitutes a six digit integer.
   The integer is then converted into the following:
         A string DD MMM YYYY
   If the incoming number is zero it will return blank date.
Parameter descriptions:
   lNoInput - Incoming number
   pDateStr - Date output in string format
Return value: NONE
*********************************************************/

#define JAN 0
#define FEB 1
#define MAR 2
#define APR 3
#define MAY 4
#define JUN 5
#define JUL 6
#define AUG 7
#define SEP 8
#define OCT 9
#define NOV 10
#define DEC 11
#define JAN_STR "JAN"
#define FEB_STR "FEB"
#define MAR_STR "MAR"
#define APR_STR "APR"
```

```
#define MAY_STR "MAY"
#define JUN_STR "JUN"
#define JUL_STR "JUL"
#define AUG_STR "AUG"
#define SEP_STR "SEP"
#define OCT_STR "OCT"
#define NOV_STR "NOV"
#define DEC_STR "DEC"

CHAR m_cMonth[12][4];
strcpy(m_cMonth[JAN],JAN_STR);
strcpy(m_cMonth[FEB],FEB_STR);
strcpy(m_cMonth[MAR],MAR_STR);
strcpy(m_cMonth[APR],APR_STR);
strcpy(m_cMonth[MAY],MAY_STR);
strcpy(m_cMonth[JUN],JUN_STR);
strcpy(m_cMonth[JUL],JUL_STR);
strcpy(m_cMonth[AUG],AUG_STR);
strcpy(m_cMonth[SEP],SEP_STR);
strcpy(m_cMonth[OCT],OCT_STR);
strcpy(m_cMonth[NOV],NOV_STR);
strcpy(m_cMonth[DEC],DEC_STR);

BOOL DateConv (long lNoInput ,char * pDateStr )
{
   struct tm *pDate ;
   static char cConvertedMonth [ 4 ];
   char s1[]="19";
   char s2[]="20";
   int j;
   CString cTempMonth;
   cTempMonth = ' ';
   if ( lNoInput == 0L )
   {
         strcpy(pDateStr,"");
   }
   else
   {
         // Convert the incoming number...
         lNoInput += 315513000L ;
         pDate = localtime ( ( time_t * ) &lNoInput ) ;
         if(pDate == NULL)
         {
               return FALSE;
         }
         if(pDate->tm_mon>=0 && pDate -> tm_mon<12)
         {
               strcpy(cConvertedMonth,m_cMonth[pDate->tm_mon]);
         }

         //NOW THE STRING FORMAT
         j =sprintf ( pDateStr , "%02d%03s" , pDate -> tm_mday ,
               cConvertedMonth);
         if (pDate->tm_year > 99)
         {
               // if year after two thousand bring year back to two
               digits
```

```
                pDate->tm_year -= 100 ;
                sprintf ( pDateStr +j,"%02s%02d",s2, pDate ->
                      tm_year);
          }
          else
          {
                sprintf ( pDateStr +j,"%02s%02d",s1, pDate ->
                      tm_year);
          }
     }
     return TRUE;
  }
```

7.2 Sample Program 2

Sample program for convert long time in to HH:MM:SS format

```
/*************************************************************
Routine Name: TimeConv
Synopsis: This routine is responsible for processing a time
      Value (LONG) which constitutes a six digit integer.
      The integer is then converted into the following:
      (takes into account decade 70 )
      A string - 99:99:99
      If the incoming number is zero it will return blank time
Parameter descriptions:
      lNoInput - Incoming number
      cTimeStr - Time output in string format
Return value: NONE
*************************************************************/
VOID TimeConv(LONG lNoInput,CHAR* cTimeStr)
{
      struct tm * pTimeStruct ;
      CString szTime;

      if ( lNoInput == 0 )
      {
            strcpy(cTimeStr,"");
      }
      else
      {
            pTimeStruct = localtime ( ( time_t * ) &lNoInput ) ;
            if ( pTimeStruct->tm_isdst == 1 )
                  --pTimeStruct->tm_hour ;
            if (pTimeStruct->tm_hour == -1 )
            {
                  // Make sure that from midnight returns to 11 pm.
                  pTimeStruct->tm_hour = 23;
            }
            sprintf ( cTimeStr, "%02d:%02d:%02d",
                  pTimeStruct->tm_hour, pTimeStruct->tm_min,
                  pTimeStruct->tm_sec);
      }
}
```

8. About SFTP (Secure File Transfer Protocol)

The file transfer takes place over SFTP (Secure FTP) protocol over the Internet.

The Info Vendor requires to provide the Exchange with the SSH RSA Public Key of their machine for receiving login details form the Exchange.

The following details will be provided once the request is processed by the Exchange:

- Server IP
- SSH Service Port
- User ID
- File Path

General information on SFTP has been provided in the following sections for popular OS platforms.

8.1 SFTP on Linux platform

The OpenSSH suite, which comes pre-installed in most Linux distributions, can be used for transferring files securely using SFTP.

The SSH key-pair is generally generated in the ".ssh" directory in the user's home directory.

It is highly recommended that you consult your systems administrator to generate/locate the key-pair and set up SFTP for you.

Continue reading for information on how to generate the key-pair.

8.1.1 Generation of the SSH RSA key-pair on Linux

- Generate the new key-pair with following command:

```
ssh-keygen -t rsa -C "user@host"
```

- You will receive the following prompt:

```
Generating public/private rsa key pair.
"Enter file in which to save the key".
```

  Press the Enter to continue with the defaults.

  You will receive the following prompt:

```
Enter file in which to save the key
(/host/users/user/.ssh/id_rsa):
```

  Press the Enter to continue with the defaults.

- If a file already exists with the same name, then you will receive the following prompt:

```
/host/users/user/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

Type "y" and press Enter to overwrite.

- You will be prompted to enter a passphrase as follows:
  `Enter passphrase (empty for no passphrase):`
  Press Enter to continue without a passphrase.

  Your will be prompted to re-enter the passphrase:
  `Enter same passphrase again:`
  Press Enter again to continue without a passphrase.

- After you enter a passphrase, you will be presented with the "Fingerprint" (or ID) of your SSH key.

  It will look something like this:

```
Your identification has been saved in
/host/users/user/.ssh/id_rsa.
Your public key has been saved in
/host/users/user/.ssh/id_rsa.pub.
The key fingerprint is:
87:c4:85:90:91:16:39:de:c2:26:49:4a:b3:38:80:97
user@host
```

After generating public key, user needs to share the Public Key file with exchange for requesting the credentials.

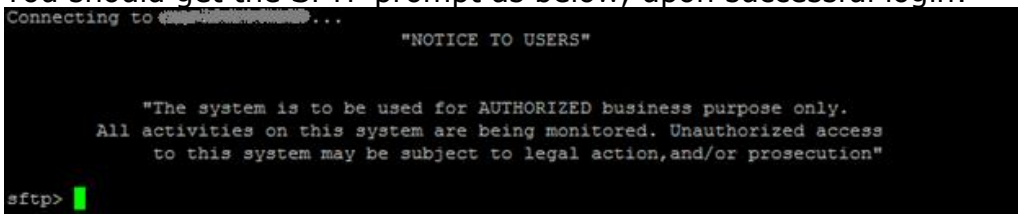
NOTE: In above steps the words "host" and "user" are used to represent the host name and user name of the machine. This is used for demo purpose only. The same will differ as per your server and user names.

8.1.2 SFTP Login

Login to the Exchange Server over SFTP using the following command:
`sftp -o PORT=6010 remote_user@remote_host`

Where remote_user is the User ID provided to you by the Exchange upon sharing your Public Key and remote_host is the Exchange Server IP.

You should get the SFTP prompt as below, upon successful login:



8.1.3 Fetching files over SFTP

The SFTP "get" command may be used at the SFTP prompt for fetching the files while logged into the host over SFTP.

8.1.4 Ending the SFTP session

The SFTP "bye" command may be used for terminating the session

8.1.5 SFTP commands help

Help may be obtained with SFTP commands by typing the "help" command at the SFTP prompt.

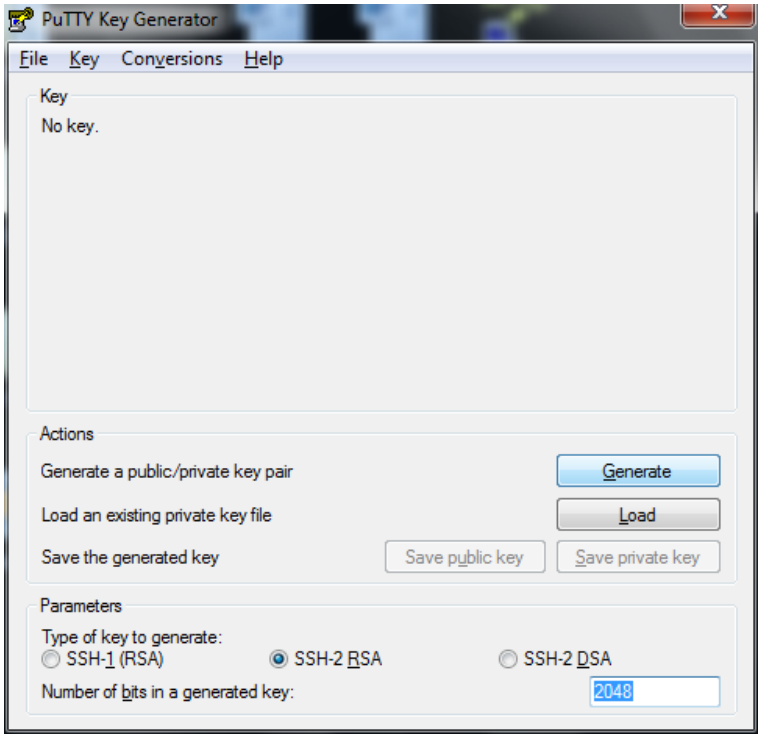
8.2 SFTP on Windows platform

8.2.1 Generation of the SSH RSA key-pair on Windows

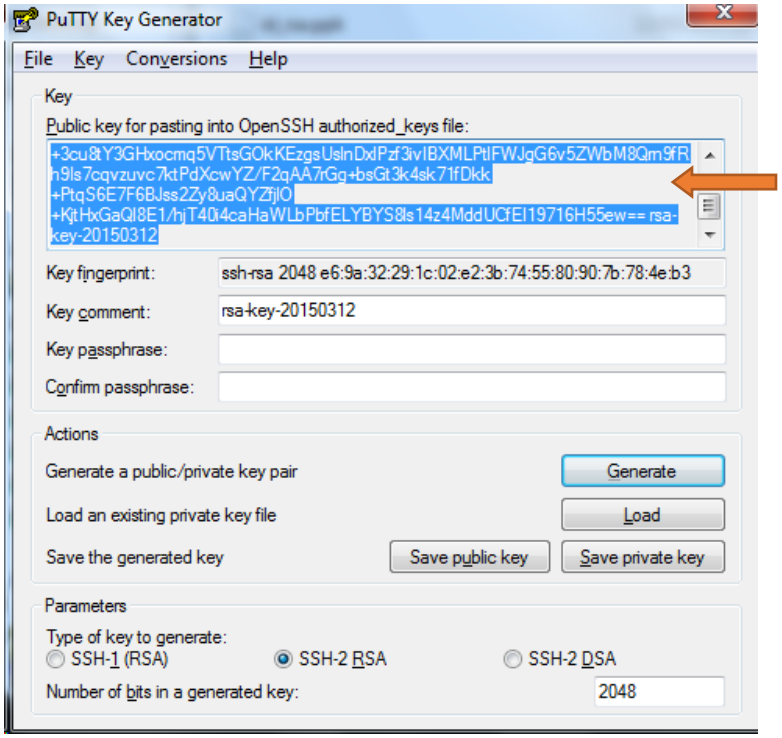
This guide explains how to generate the SSH RSA key-pair using the PuttyGen application.

Download the PuttyGen application (freely available on the Internet). Then follow these steps to generate the key-pair:

- Start the PuttyGen application.
  You will be presented with a dialog which looks something like this:



- Select "SSH2RSA" with 2048 bit size or greater.
- Press the "Generate" button.
- After generating the key, you will be shown the screen below.
  Keep the "Key passphrase" and "Confirm passphrase" as blank.



SSH RSA Public Key should be copied from here after generation

- Create a blank file with the name "id_rsa.pub".
  This will be the public key file which will be populated with your Public Key and shared with the Exchange.

- Copy the public key content as presented on the screen (selected area in the below screenshot) and paste into newly created public key file (id_rsa.pub) and save the file.

- Share this Public Key File (id_rsa.pub) with the Exchange when requesting for SFTP credentials.

8.2.2 SFTP Client Software on Windows

There are multiple SFTP Client Programs (paid for and free) available for transferring files over SFTP.

One such software is WinSCP, available for free from the WinSCP website. This program is intuitive, user friendly and can be used in interactive mode (GUI) as well as from the command line (for automation/batch processing).

Information on using WinSCP can be found on the WinSCP website.

8.3 Further support

Apart from the above guide, many of the online resources can be referred on the World Wide Web for more information on how to set up and use SFTP at the Client's site on various OS platforms.

Note:

This "About SFTP" section is intended as a guide used to understand and become familiarized with this transfer protocol.

It may be noted that the Exchange does not provide SFTP software or support for configuring and using SFTP at Client site.

9. Contact Information

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