

TECHNICAL DOCUMENT

REAL TIME DATA

SECURITIES LENDING AND BORROWING (SLB) MARKET (LEVEL 1)

19 MAY 2010



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

COPYRIGHT NOTICE

All rights reserved. No part of this document may be reproduced or transmitted in any form and by any means without the prior permission of DotEx Ltd.

CONTENTS

Chapter	Contents	Page No.
1.	INTRODUCTION	4
2.	CONNECTION DETAILS	5
2.1.	STRUCTURAL DIAGRAM	5
2.2.	ONLINE REQUIREMENTS	5
2.3.	STEPS FOR AUTHENTICATION AND RECEIVING FEED	6
2.4.	STEPS FOR DECOMPRESSING FEED	8
3.	DATA DETAILS	11
3.1	THE HEADER	11
3.1.1	CODE	11
3.1.2	LENGTH	11
3.1.3.	SEQUENCE NUMBER	11
3.2	DATA BODY	11
3.3	TRAILER	11
4.	DATA STRUCTURE DETAILS	12
4.1.	LOGIN REQUEST	12
4.1.1	INFO HEADER	12
4.1.2	ASCII DATA	12
4.1.3	INFO TRAILER	12
4.2.	LOGON RESPONSE	12
4.2.1	COM HEADER	12
4.2.2	INFO HEADER	12
4.2.3	ASCII DATA	12
4.2.4	INFO TRAILER	12
4.3.	MARKET OPEN	12
4.4.	MARKET CLOSE	13
4.5.	NEW UPDATE	13
4.6.	BROADCAST MESSAGE	14
4.7.	SECURITY DESCRIPTOR	15
4.8.	END OF THE DAY STATUS	16
4.9.	END OF THE FEED	17
4.10.	HEARTBEAT SIGNAL	17
4.11.	SECURITY INFORMATION	18
5.	CONTACT INFO	29
6.	NOTE	20
6.1.	NORMAL MARKET	20
6.2.	AUCTION MARKET	20
7.	CHECKSUM	21
8.	EXAMPLE: FUNCTION FOR DECOMPRESSION	22

REAL TIME DATA TECHNICAL SPECIFICATION

SLB MARKET

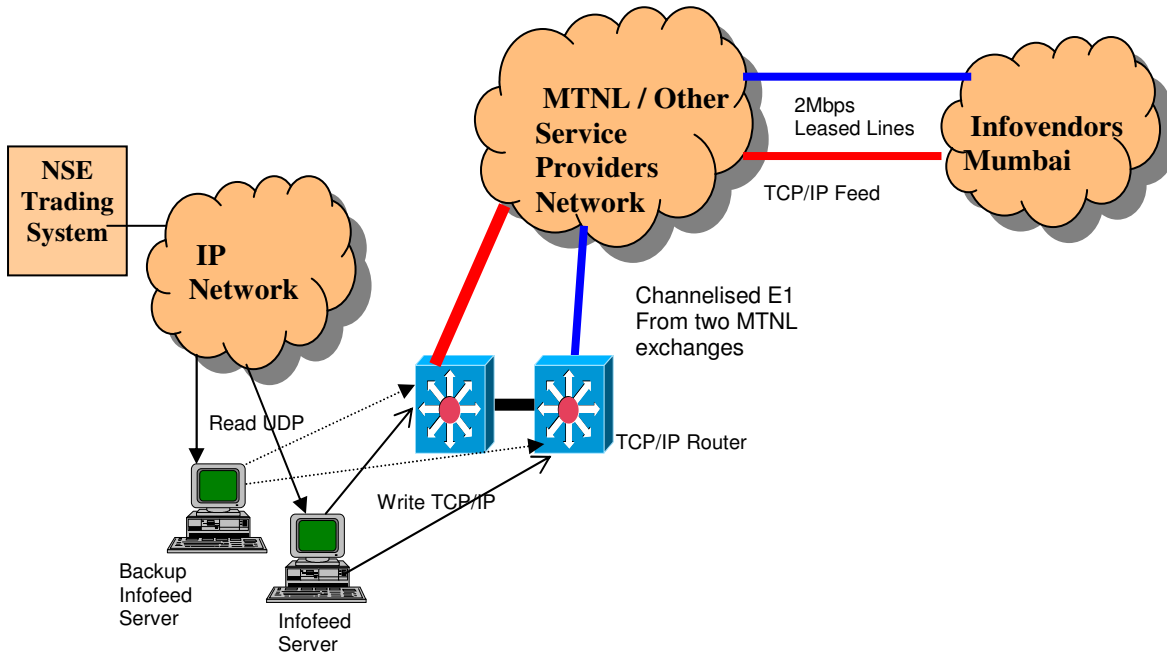
1. INTRODUCTION

DotEx International Ltd. disseminates NSEIL's (National Stock Exchange of India Ltd.) real time broadcast data to various information agencies. It provides the 3 different types of data to 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 format, where as the snapshot data and End of day data is available in the form of files. The Infofeed server provides NSEIL real time broadcast data. The information agencies connect to the Infofeed Server through 2 Mbps Leased Lines. These leased lines are terminated on Infofeed Router and their data specific mnemonic 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.

2. CONNECTION DETAILS

Structural Diagram

The structural diagram of Real Time data connection has been explained below -



Online Requirements

- A Router / Switch or a card with TCP/IP capabilities to connect to 2 Mbps transmission lines for receiving NSEIL's Real time information.
- The Information agency should develop applications that initiate TCP/IP calls through 2 Mbps Leased Line.

Steps for Authentication AND Receiving feed

- a) Client applications at vendor end, establish the connection with Infofeed Server application using specified IP address and Port.
- b) After establishing the connection, client application sends the login packet to Infofeed server application.

Packet format of Login packet is as follows,

```
struct LOGIN_REQ
{
    struct INFO_HEADER sHeader;
    struct LOGIN_REQ_DATA sData;
    struct INFO_TRAILER sTrailer;
};
struct LOGIN_REQ_DATA {
    char cUserId[10];
    char cPassword[8];
    char cNewPassword[8];
    char cConfmPassword[8];
};
struct INFO_HEADER
{
    short iCode;
    short iLen;
    LONG lSeqNo;
};
struct INFO_TRAILER
{
    short iChecksum;
    CHAR cEOT;
};
```

- c) Password field is case sensitive, password should be minimum 6 characters long, password and user id should not be same, password should start with alphabet and password should be alphanumeric (No wild characters are allowed).
- d) If user wants to change his password then the user needs to specify new password & confirm password (Both fields should match) otherwise leave it blank.
- e) Based on the above information, user will get Log on response from Infofeed Server.

```
struct LOGIN_RESPONSE
{
    struct COM_HEADER sCOMHeader;
    struct LOGIN_RESPONSE_DETAIL sDetail;
};
```

```

struct COM_HEADER
{
    char cCmopOrNot;
    short iPackLen;
    short iNoOfPack;
};
struct LOGIN_RESPONSE_DETAIL
{
    struct INFO_HEADER sHeader;
    struct LOGIN_RESPONSE_DATA sData;
    struct INFO_TRAILER sTrailer;
};
struct LOGIN_RESPONSE_DATA
{
    long iErrCode;
    char cMesg[50];
};

```

Following Error code will be returned which client needs to interpret as:-

1000- Successful

1001- Password Update Successfully

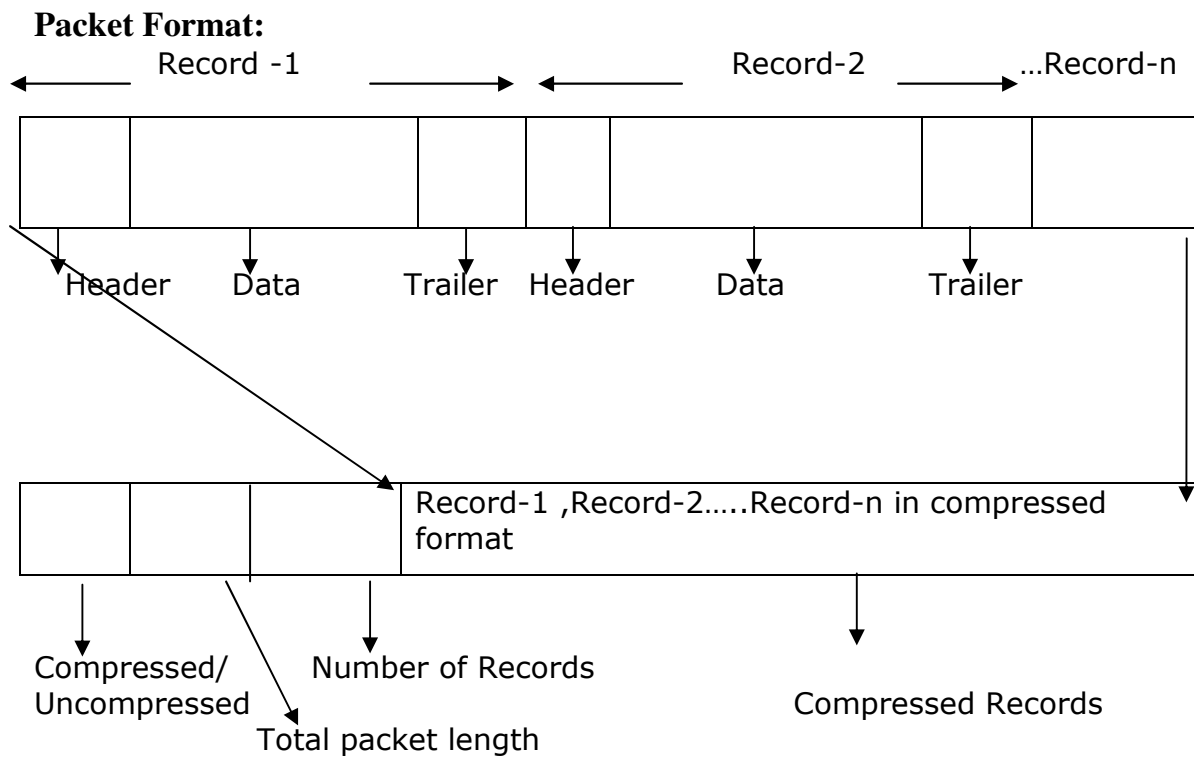
1002- Wrong UserId-Password Combination

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

- f) After successful login, Infofeed Application starts sending packets in the below format.



Compressed/Uncompressed: This field tells whether packet is compressed or not compressed.
 If this Field = 0 then Compressed.
 Field = 1 then Uncompressed.

Number of Records: This field tells the number of records present in the compressed packets.

Packet length: this field specifies the total packet length.

```

Structure COM_HEADER
{
    char cCmpOrNot;
    short iPackLen;
    short iNoOfPack;
};

```

- g) As the data packets are sent in compressed format there is a need to decompress them. The compression algorithm used is LZ0.
- h) The Decompression algorithm used should be LZ0

Steps for Decompressing feed

- LZO Algorithm Details:
 - a) LZO is a data compression library which is suitable for data de-/compression in real-time. This means it favors speed over compression ratio.
 - b) 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 less memory for decompression.
- Compression is pretty fast.
- 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 decompressor 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.

- c) Files required for LZO algorithm.
 - Include files, source files (src) provided by LZO
 - LZO.lib

For more information on LZO library and downloads visit:

http://www.nseindia.com/content/press/prs_whatsnew.htm#2

Specifications for utilizing on-line broadcast information

- **Decompression steps:**

- a) Receive the packet in the temporary buffer i.e. array of characters.
- b) First field will identify whether the packet is compressed or not.
- c) If this field is **0** then Decompress it using LZO algorithm else if **1** **don't** decompress it and proceed in normal way as it is being done today.
- d) The second field is packet length.
- e) The third field contains the number of records in the packet.
- f) If compressed use following function of LZO to Decompress.
r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength, (unsigned char*) cOutputBuf, opLength, NULL);

lzo1z_decompress: Function which decompresses the data packet receive

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

- g) After decompression data will be available in Output Buffer.
- h) Map the outputbuf to existing Header structure according to **SN**.
- i) Look for Record size in the length field and Code (i.e. SN).
- j) Steps to recover data from OutputBuf.

Algorithm:

```
Length_of_Record = Header->length;
Sequence_no = Header->Sequence_num;
For I = 0 to Number of records (obtained in step 4)
Begin
  Bytes_to_seek = Length_of_Record * I
  Seek to number of Bytes_to_seek
  Map (Length_of_Record) of bytes to proper structure according
  to iCode (SN) as found in Header part.
  Do the required processing....
  ....
End
End for Loop.
```

3. DATA DETAILS

The real time data is disseminated in the form of TCP/IP packets and each single packet generated by Infofeed system with a definite structure i.e. Header, Data body and Trailer.

THE HEADER

The header in turn consists of 3 fields – Code, Length and Sequence number. The details of these fields are explained as below.

Code – It is a 2-Char field that provides the information about the type of packet or the type of data that each packet contains. The NSEIL real time data contains various types of packets that may or may not have data part. Each packet can be differentiated with the code filed. The various types of packets disseminated as real time data feed are – Login Request (SQ), Logon Response (SR), Heartbeat Signal (SH), Market Open (SO), Market Close (SC), New Update (SN), Security Addition (SA) and Modification (SM), End of day Market Status (SS), End of Feed (SE), Security Information (ST) etc.

Length – It is a 2-byte hexadecimal field that provides the length of record within the each packet of NSEIL real time data. This includes the length of Header, Data, Trailer and the Carriage return.

Sequence Number – It is a 4-byte ASCII field that provides the sequence number of each packet that is disseminated in NSEIL real time data. The very first packet sequence number is initiated with number 1 (One) for a each market and each successive packet is incremented by one thereafter.

DATA BODY:

The following information is provided in data block -

- a. Market Open
- b. Market Close
- c. Trade and Order information
- d. Broadcast Messages
- e. Descriptor Addition, Modification and Deletion
- f. Market Status at the end of the day
- g. End of the feed
- h. Heart Beat Signal
- i. Security Information

TRAILER

The trailer is a two-byte checksum. A CR will terminate the block of data.

4. DATA STRUCTURE DETAILS

4.1 LOGIN REQUEST (send by client application)

Login Request packet will be send by the client application for login into the Infofeed application. If user wants to change his password he will specify the new password and confirm password field. Password is case sensitive. Format of this packet is as follows.

INFO HEADER:

- | | |
|--------------------|-------------------------|
| a) Code | 'SQ' |
| b) Length | Short Integer (2 Bytes) |
| c) Sequence Number | 00 |

ASCII DATA:

- | | |
|---------------------|----------|
| d) User Id | 10 Chars |
| e) Password | 8 Chars |
| f) New Password | 8 Chars |
| g) Confirm Password | 8 Chars |

INFO TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.2 LOGON RESPONSE (send by Infofeed application)

Logon response packet will be send by the Infofeed server application after receiving the Login Request packet from the client application.

COM HEADER:

- | | |
|----------------------|-------------------------|
| a) Compressed or Not | 1 Char |
| b) Packet Length | Short Integer (2 Bytes) |
| c) No of Packets | Short Integer (2 Bytes) |

INFO HEADER:

- | | |
|--------------------|-------------------------|
| a) Code | 'SR' |
| b) Length | Short Integer (2 Bytes) |
| c) Sequence Number | Long (4 Bytes) |

ASCII DATA:

- | | |
|---------------|----------------|
| a) Error Code | Long (4 Bytes) |
| b) Message | 50 Chars |

INFO TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.3 MARKET OPEN:

The Market Open message is received whenever any particular market opens for the day. Market Open message will be received separately for different markets.

HEADER:

- | | |
|--------------------|------|
| a) Code | 'SO' |
| b) Length | 0x0C |
| c) Sequence Number | XXXX |

ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Market Type	1 Char ('N'-Normal, 'A'-Auction, 'L'-All Markets)

TRAILER:

Checksum is not computed.

4.4 MARKET CLOSE

The Market Close message will be received whenever any particular market closes for the day. Market Close message will be received separately for different markets.

HEADER:

- | | |
|--------------------|------|
| a) Code | 'SC' |
| b) Length | 0x0C |
| c) Sequence Number | XXXX |

ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Market Type	1 Char ('N'-Normal, 'A'-Auction, 'L'-All Markets)

TRAILER:

Checksum is not computed.

4.5 NEW UPDATE

These packets will be received only during the trading period of the particular markets and signify the trades occurring in that market. The Online Index field in this packet will be always zero.

HEADER:

- | | |
|--------------------|------|
| a) Code | 'SN' |
| b) Length | 0xB9 |
| c) Sequence Number | XXXX |

ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Symbol	10 Chars
2) Series	2 Chars
3) Market Type	1 Char ('N'-Normal, 'A'-Auction)
4) Best Buy-Order Price	10 Chars
5) Best Buy-Order Quantity	12 Chars
6) Best Sell-Order Price	10 Chars
7) Best Sell-Order Quantity	12 Chars
8) Last Traded Price	10 Chars
9) Total Traded Quantity	12 Chars
10) Security Status	1 Char (Suspended (S) or Blank)
11) Opening Price	10 Chars
12) High Price	10 Chars
13) Low Price	10 Chars
14) Close Price	10 Chars
15) Average Traded Price	10 Chars
16) Total Turnover	25 Chars
17) Online Index	8 Chars (Value will be always zero)
18) Settlement Date	11 Chars (Format DD-MON-YYYY)

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.6 BROADCAST MESSAGE

These packets consist of the messages broadcast during the Trading time containing information like changes in the price bands of particular scrips and market-related information.

HEADER:

a) Code	'SB'
b) Length	0xXX (Variable depending on the length of the Message string)
c) Sequence Number	XXXX

ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Message Code	3 Chars (NSE or AUC)
2) Message Length	3 Chars
3) Message String	(Message Length) No. Of Chars

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

8) Last Traded Price	10 Chars
9) Previous Close Price	10 Chars
10) Total Traded Quantity	12 Chars
11) Total Traded Value	25 Chars
12) Settlement Date	11 Chars (Format : DD-MON-YYYY)

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

4.9 END OF THE FEED

This end of the packet indicates that all the parts of End of the Day feed have been completed.

HEADER:

a) Code	'SE'
b) Length	0x0B
c) Sequence Number	XXXX

ASCII DATA:

Not associated with any ASCII data.

TRAILER:

Checksum is not computed.

4.10 HEARTBEAT SIGNAL

The Heartbeat packets are sent throughout the day from 8:00 a.m. to 08:30 p.m. This packet indicates to the Info-Vendors that data packets are received from the Infofeed server.

HEADER:

a) Code	'SH'
b) Length	0x0B
c) Sequence Number	XXXX

ASCII DATA:

Not associated with any ASCII data.

TRAILER:

Checksum is not computed.

4.11 SECURITY INFORMATION

These packets are sent at the beginning of the each trading day by approximately **08:45 AM**. This feed contains the information about the securities valid in the CM Market for trading.

HEADER:

a) Code	'ST'
b) Length	0x3B
c) Sequence Number	XXXX

ASCII DATA:

The format of the ASCII data sent is as follows:

FIELD TYPE	FIELD WIDTH
1) Token Number	10 Chars
2) Symbol	10 Chars
3) Series	2 Chars
4) ISIN Number	12 Chars
5) Is Deleted	1 Char
6) Settlement Date	11 Chars (DD-MON-YYYY)
7) Allow Recall	1 Char ('1' = Yes, '2' = No)
8) Allow Repay	1 Char ('1' = Yes, '2' = No)

TRAILER:

The Trailer is a 2-byte checksum. A CR will terminate the block of data.

5. CONTACT INFO

Following are the contact details for business assistance:

Name	Email Address	Contact Numbers
Mr. Ved Malla	vmalla@nse.co.in	91-22-26598385
Ms. Prachee Chavan	pracheec@nse.co.in	91-22-26598385
Mr. Pankaj Agarwal	pagarwal@nse.co.in	91-22-26598385

You can also email us on **dotex@nse.co.in**

For technical assistance email us on **infofeed_support@nse.co.in**.

6. NOTE

6.1 Normal Market

In the Pre-Open phase of the market, the Order Price can be 0.00. This indicates that the order placed is an At Open Order (ATO) and the actual Order Price will be calculated by the system when the market opens. Even though no trade takes place in the Pre-Open phase the Last Traded Price and the Open Price will change as the orders are placed in the system. The final open price for the security is calculated after Pre-Open period has ended. The Total Traded Quantity will be zero in the Pre-Open phase. In the Market Open phase also the Total Traded Quantity will be zero when orders are placed but no trades take place. The Last Traded Price will never be zero in the Normal Market.

6.2 Auction Market:

In the auction market the Open price and the Last Traded Price would be zero till the auction ends and the auction price is calculated by the system. Since Auction in any particular scrip is done at a fixed price the High Price, Low Price, Closing Price and Index values is zero for all scrips traded in the Auction Market.

The auction market timings are from 10:30 hrs to 11:00 hrs.

7. CHECKSUM

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);
}
```

8. EXAMPLE: FUNCTION FOR DECOMPRESSION.

```
lzo_decomp (char cInputBuf [], unsigned int ipLength, char cOutputBuf [],
unsigned *opLength, unsigned short *lzo_errorcode)
{
    int r;
    Char mess [50];
    r = lzo1z_decompress ((unsigned char *) cInputBuf, ipLength,
                        (unsigned char *) cOutputBuf, opLength,
                        NULL);

    If ( r == LZO_E_OK)
    {
        Print (mess," Decompressed %lu bytes back into %lu bytes\n",
              (long) ipLength, (long) *opLength);

        Return true;
    }

    Else
    {
        OutputDebug ("Internal error - decompression failed");
        Return false;
    }
}
```