Parser instructions

Deutsche Börse Data Shop

How to use parsers provided by the Deutsche Börse Data Shop

1. Introduction

Files captured from streaming data can be purchased in the Deutsche Börse's Data Shop. Two parsers are available for customer's convenience, facilitating parsing of these files without extra development effort.

In the customer's download area, you can find:

- emdda<version number>. tgz → it decodes EMDI, RDI, EMDS and ENBS files
- EOBI_Parser.zip → it decodes EOBI files

Both archives contain Linux 64-bit executables that run in a shell environment. This document provides instructions for their use. The reader should be familiar with protocols and message formats EMDI, RDI, ENBS, and EOBI as far as necessary for understanding the parsing process and interpretation of the parsing result.

Documentation for the used protocols is available on the Data Shop's public website.

2. Parser for EMDI, RDI, EMDS: "emdda"

Prerequisites

- emdda was compiled for an Intel/AMD 64-bit architecture
- As emdda is a statically linked executable, there are no dependencies on special libraries
- When parsing pcaps, enough RAM for loading the entire input file is required; if pcap files are too large for the memory available, they need to be split into parts by a tool such as editcap

Please note that the archive comprises a manual with comprehensive descriptions. Hence, only some tips for usage "out of the box" follow.

Usage

Within a shell (Linux terminal), invoke emdda by simply typing

```
<path_to_executable>/emdda
```

Without additional parameters, emdda will show all parameters/options available.
The manual gives detailed descriptions of these options.

emdda can read a number of input file formats and produce various output file formats. Convenient for parsing the streaming data files purchased at Deutsche Börse Data Shop is parsing pcap (Packet CAPture) files with output to FIX:

```
emdda  -fix c -t EMDIFastTemplates-1.1.xml -pcap pcap_file_name >
output_file_name.fix
```

where
-fix: FIX output
c: (Next to "-fix") requests output both of FIX tags and field names ("combined")
Common alternatives would be "-fix n" for FIX tags ("numeric") or "-fix h" for field names ("human readable")
-t: Name of template file follows. The template file needs to belong to the same system version as the streaming data file
-pcap: Name of pcap file follows

Special considerations

Emdda for EnBS
emdda can parse ENBS files (pcap or non-pcap) with the following limitations:

- Default FIX output (single FIX message per line) will lack field IDs because there aren't any defined in ENBS.XML templates; one should always be using the -fmt or -fix h option:

```
emdda  -fix h -t ENBS.XML -pcap pcap_file_name > output_file_name
```

- Timestamps are shown only as a raw number - human readable will be wrong because ENBS uses microseconds since midnight instead of nanoseconds since 1970
- Gap check will not work
- Reference data cycle detection will not work

Non-pcap input
If the streaming data was captured and written to a log without encapsulation into pcap, emdda will decode it if it is invoked using the "-file" option instead of "-pcap":

```
emdda  -fix c -t EMDIFastTemplates-1.1.xml -file input_file_name >
output_file_name.fix
```

Note: The standard case in the Data Shop is delivery of streaming data in pcap format. Above option is described for exceptional cases only.

Reference data
Whenever a file archive delivered by the Data Shop comprises a dedicated file with reference data, this needs to be parsed for obtaining the correct instrument identifiers' (SecurityID for T7 interfaces or isix for ENBS) mapping to instruments.

Reference data file names are:

```
ENBS: 1308_*_YYYYMMDD.pcap
RDI: 2302_*_YYYYMMDD.pcap
```

3. Parser for EOBI: "deobi"

Prerequisites

- The executable was compiled for an Intel/AMD 64-bit architecture
- deobi makes use of a number of libraries that are listed by ldd when applied to deobi

```
$ ldd deobi
        linux-vdso.so.1 =>  (0x00007ffdfbd87000)
        libpcap.so.1 => /lib64/libpcap.so.1 (0x00007fae6c0b8000)
        libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007fae6bdb1000)
        libm.so.6 => /lib64/libm.so.6 (0x00007fae6baaf000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007fae6b899000)
        libc.so.6 => /lib64/libc.so.6 (0x00007fae6b4cc000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fae6c2f9000)
```

The package EOBI_Parser.zip contains the library libcap.so.1 that can be installed if not available on the target system.

Usage

Invocation of deobi without additional parameters gives a list of all parameters/options available. The file archive containing deobi comes with a shell script for invocation and a sample output configuration file. The script consists of only this line for starting deobi:

```
./deobi -s eobi.xml -c output.conf --input-file input_file_name
```

where
-s: XML spec file name
-c: CSV mappings file name
--input-file: input file(s), containing the binary messages

Special considerations

Output configuration file

Parser output is written to csv files – one per EOBI message type. The fields populated in these files need to be listed in the configuration file given to the parser as argument to option "-c". The parser package provided by Deutsche Börse contains a sample configuration file named output.conf (where the name doesn't have any meaning to the parser).

A frequent misunderstanding is to consider the sample configuration a "ready to use production set-up" – which is not the case:

In order to extract the correct messages/fields, one needs to edit the parser's configuration file accordingly, by looking up in the EOBI manual which fields in which message types are relevant. The first string in each line of the configuration file is the name of an EOBI message type (cf. EOBI documentation), represented by one csv file.

Note: The PARENT_ID and ID fields are not in the configuration file but hard coded. They are meant to allow reassembling the original technical message. They should not be required for understanding the business contents.

Orphaned messages
In case the parser complains about orphaned messages like this…

```
Received orphaned order for triple (224.0.114.48, 59000, 675) for timestamp
2017/11/01 12:42:07.914698811 and application sequence number 961130

Received too few orders for quadriple (224.0.114.42, 59000, 731, 2318335) for
timestamp 2017/11/01 12:42:59.384803981, message sequence number 1543

Received too few orders for quadriple (224.0.114.32, 59000, 691, 2307758) for
timestamp 2017/11/01 12:42:59.112293391, message sequence number 988
```

… this is due to how the snapshot cycles are spanned over multiple UDP datagrams and the parser doesn't read the snapshot stream from the beginning of the snapshot cycle.

If one reads the snapshot from the beginning of the snapshot cycle, the parser should not give any warnings.

Reference data
The EOBI file does not contain reference data. The RDI file packaged with the EOBI files delivers reference data and SecurityID for mapping to EOBI messages.

Important: For parsing the RDI file, use the emdda parser – deobi is not able to parse RDI.

Distinguish files:

```
EOBI: 2350_*_YYYYMMDD.pcap
RDI: 2302_*_YYYYMMDD.pcap
```

4. References

Manuals for the streaming data file protocols and formats are available on the Deutsche Börse Data Shop website at https://datashop.deutsche-boerse.com/market-data-stream-files.

Eurex provides additional information at: https://www.eurexchange.com/exchange-en/technology/t7/system-documentation/

Published by
Deutsche Börse Group
Data Services
60485 Frankfurt/Main
www.data.services.deutsche-boerse.com

January 2020