©QuantHouse

FeedOS Toolbox User Guide

2010-09-10

Introduction

This documentation is designed for the FeedOS Toolbox users, it describes the tools and their supported requests.

Document History

Date	Author	Action
2010-07-07	P. Kossaifi	New Document

Contents

1 The Toolbox Overview

The FeedOS Toolbox contains command-line interface tools that allow discovering and querying the available data on a FeedOS server (typically, a FrontEnd Server).
The Toolbox is available on Linux and Windows platforms, under Windows we recommend using Cygwin's window command http://www.cygwin.com
The Toolbox most used tools are :

- feedos_cli : allows sending requests that have the same functionalities, names and parameters as the C++ API requests.
- quotation_export : allows exporting the quotation data to a text file.
- referential_export : allows exporting the referential data to a text file.
- get_histo : allows retrieving the instrument list of values over a period of time.
- ref_varticksize_tables_export : display the tick size tables.
- tradecond_dictionary_export : display the trade conditions dictionary.
- feed_replay : replay data from a FeedOS recording/replay server and print it on a text file.

2 The feedos_cli

The Feedos_cli allows the connection to the FeedOS server and send requests that have the same functionalities, names and parameters as the C++ API requests. It can be used in both interactive and script[1] modes.
To use the Feedos_cli, access to the Toolbox folder on your machine using a command windows and execute ./feedos_cli on a Linux platform or feedos_cli on a Windows platform.

```
$ ./feedos_cli
feedos_cli > help

Usage: connect [-proxy <hostname> <port>] <hostname> <port> <username> <password>
        disconnect
        [exec] [<service>.]<request> [<arg1><arg2>...]
        call <script_file>
        quit
        help [type <type>] [ [exec] [<service> | [<service>.]<request>] ]

help          - online help. Try "help exec" and "help type".
connect       - connect to the specified server.
disconnect    - disconnect from the current server.
exec          - execute the specified FeedOS request.
status        - display internal status.
echo          - (for scripting) print out the given arguments.
msleep        - (for scripting) sleep during N milliseconds.
version       - version info.
quit          - exit the program.
call          - execute commands in the given script file.
date          - print current date & time (UTC).
unset         - clear an internal variable.
set           - assign an internal variable.
read          - assign an internal variable by reading current input stream.
read_console  - assign an internal variable by reading program's stdin.
```

[1] useful for automating tasks

2.1 Basic Commands

1. help : assure on line help and display all the feedos_cli commands
2. connect : allows the connection to a remote FeedOS server, using the server name and port and the user credentials.
   Usage example : connect 84.22.333.222 6040 User pwd
3. exec : is a default command, execute the given API request (must be connected to a server).
4. disconnect : allows the disconnection from the remote FeedOS server
5. quit : quit the feedos_cli

2.2 Advanced Commands

1. status : display the current connection status
2. version : version info.
3. set : assign an internal variable.

   ```
   Example:
   feedos_cli > set
   _date_prints_timet=false
   _dump_request_results=true
   _show_admin_messages=true
   _show_heartbeats=true
   _verbosity=false
   ```

4. unset : assign an internal variable by reading current input stream.

   ```
   Example:
   feedos_cli > unset _date_prints_timet
   ```

5. call : execute commands in the given script file.

   ```
   Usage:
   call script_file
   ```

2.3 Scripting Commands

1. echo : print out the given arguments (used for scripting)
2. read : assign an internal variable by reading current input stream.
3. read_console : assign an internal variable by reading program's stdin.
4. date : print current date & time (UTC).
5. msleep : sleep during N milliseconds (used for scripting)

2.4 The Syntactic Rules

2.4.1 The Command Line

The command line have this form : Command_Name [PARAMETERS] [>OUTPUT_FILE] [<INPUT_FILE]
Where the parameters can be :

- Single word (used for basic strings, numbers, enumerated values, ...)
- Quoted string (used for strings containing spaces)
- Structured values (used to group a predefined list of values)
- Lists (used to group a repetition of values)

2.4.2 Parameters

- Timestamp : can be a simple word or a quoted string.
  - 2009-12-09
  - now
  - null
  - "2009-12-28 12:34:56"
  - "2009-12-28 12:34:56:999"
  - "2009-12-28 12:34:56:999.111"

  N.B.: - Microseconds can only be used on the newest servers.
  - Frequently, dates and timestamps are expressed in Universal Time Coordinates (UTC).

- Structured values : are closed in curly brackets, the number and the type of the enclosed elements are predetermined.
  - {"this is a string" now}
  - {1 2 3 4 "string" 5 6 7.1 "2009-12-01" 8.999}

  N.B.:
  - The context defines the structure format.

- Lists : are delimited with round brackets and the enclosed values must have the same type.
  - ("this" is a "list of strings")
  - (1 2 3 4 55)

  N.B.:
  - A list can be empty.
  - If a list contains only one values the parentheses can be omitted.

- Lists and structures : they can be combined.
  - {1 2 "three" (4 5 6 7)"eight" {9 10}}({a b}{c d}{"e" "f g h"})

  N.B.:
  - The context always defines the expected format

2.4.3 Structured Values

1. The values list are delimited by parenthesis : (”str1” ”str2”)
2. The structures are delimited by braces : { 10 ”str1” Label }

2.5 The Interactive Usage

The feedos_cli can be use in an interactive mode, it reacts as soon as the requests are seized. In the session sample bellow contains:

1. Sending the command Referential.Lookup to locate only two instruments matching the pattern DAX

```
feedos_cli (User@SRV:PORT) > ref.lookup DAX 2 narrow
instr # 89/7528 = 186654056
    Symbol                      string{DAX}
    Description                 string{DAX (PERFORMANCE-INDEX)}
    SecurityType                string{NONE}
    FOSMarketId                 XETR
    CFICode                     string{MRIXXX}
    InternalCreationDate        Timestamp{2009-10-27 08:00:24:008}
    InternalModificationDate    Timestamp{2009-11-30 17:31:23:876}
    InternalSourceId            uint16{51}
    LocalCodeStr                string{DE0008469008}
    ISIN                        string{DE0008469008}
    WertpapierKennNummer        string{846900}
instr # 81/757263 = 170626575
    PriceCurrency               string{EUR}
    Symbol                      string{DAX}
    Description                 string{LYXOR ETF DAX}
    SecurityType                string{NONE}
    FOSMarketId                 XPAR
    CFICode                     string{EUXXXX}
    RoundLot                    float64{1}
    SecuritySubType             string{263}
    InternalCreationDate        Timestamp{2009-09-28 04:15:00:802}
    InternalModificationDate    Timestamp{2009-12-01 05:15:55:881}
    InternalSourceId            uint16{95}
    LocalCodeStr                string{LU0252633754}
    ISIN                        string{LU0252633754}
    MARKET_EURONEXT_InstrumentGroupCode string{18}
    MARKET_EURONEXT_TypeOfUnitOfExpressionForInstrumentPrice    string{1}
    MARKET_EURONEXT_NominalMarketValueOfTheSecurity     float64{0}
    MARKET_EURONEXT_QuantityNotation    string{UNT}
    MARKET_EURONEXT_IndicatorOfUnderlyingSecurityOnLending      string{2}
```

2. Sending the command Quotation.SnapshotInsturmentsL1 to retrieve the Last Price for 2 instruments using their internal code.

```
feedos_cli (User@SRV:PORT) > snapshotinstrumentsL1 ( 186654056 170626575 ) LastPri
InstrumentStatusL1
-- 89/7528
        BID: 0   0      *NO ORDER*
        ASK: 0   0      *NO ORDER*
        LastPrice                       float64{5743.75}
InstrumentStatusL1
-- 81/757263
        BID: 56.54      15014   @1
```

```
ASK: 56.585     15014   @1
LastPrice                       float64{56.57}
```

2.6 The Script Usage

In order to execute several commands in a row, a script file can be created it must contains a feedos_cli command per line.
To run the script, you have to pipe it into the command's standard input, in case of error the return code is not null.

```
Usage:
1 $ ./feedos_cli SERVER PORT User pwd < my_script
2 $ ./feedos_cli SERVER PORT User pwd call my_script
3 $ ./feedos_cli SERVER PORT User pwd
    user@SERVER:PORT> call my_script
```

2.7 The Available Requests

All requests are grouped into "services", according to their role. The main services are REFERENTIAL and QUOTATION.
The full syntax for the request is " exec Service.Command *arguments ...* ". Both " exec " and " Service " can be omitted, for example : all commands below can be used to issue request "DumpStructure" of the "Referential" :

- exec REFERENTIAL.DumpStructure
- exec ref.dumpS
- ref.dumpS
- dumpS

To avoid ambiguity, it is recommended to use the verbose syntax.

N.B. :

1. The greater part of requests have default parameters, like the "referential.lookup" that can be invoked only with a pattern string as its first parameter.
2. Some requests are blocking and can be stopped only when the user hits the ENTER key, these requests display the incoming data as soon as they arrive. For example the subscription requests are designed to receive continuously real-time data updates from the server.

2.7.1 Requests related to Referential Data

Aka static or reference data.

```
feedos_cli > help exec Referential

available requests for 'REFERENTIAL' service:
     DumpContent
     DumpStructure
     FindDerivatives
     GetInstruments
     GetMarkets
     Lookup
     ResolveCodes
```

1. DumpStructure : Retrieves the available Markets list, and its instruments numbers (sorted by category).
   Usage : this command does not have arguments.
2. FindDerivatives : Find the instrument derivatives.
   N.B. : this command is not supported, the derivatives market and the market which contains the underlying are managed differently and in the different pipes, and the connection between both markets could only be done manually and not automated.
3. GetInstruments : Retrieves the characteristics of given instruments.

```
 Usage:
 exec REFERENTIAL.GetInstruments <Arguments>

      <Arguments>:
                   ( <codes:PolymorphicInstrumentCode> )
                   ( <format:ReferentialTagNumber> )
                   <ignoreinvalidcodes:bool>
```

```
feedos_cli (User@SRV:PORT) >
GetInstruments ( XCME@USDU0 XCME@USDZ9 ) ( ISIN LocalCodeStr )
instr # 309/310122 = 648330090
     LocalCodeStr                string{USDU0}
instr # 309/134405 = 648154373
     LocalCodeStr                string{USDZ9}
```

4. GetMarkets : Retrieves the characteristics of a given Market.

```
Usage:
exec REFERENTIAL.GetMarkets <Arguments>

     <Arguments>:
                  ( <marketids:FOSMarketId> )


feedos_cli (User@SRV:PORT) > GetMarkets XCME
market # 309    CME
```

```
    MIC = XCME
    TimeZone =
    Country =
    NbMaxInstruments = 1000000
```

5. Lookup : Search for instruments, using a pattern and optional filtering/formatting.

```
Usage:
exec REFERENTIAL.Lookup <Arguments>

     <Arguments>:
                   <pattern:String>
                   <maxhits:uint16>
                   <mode:LookupMode>
                   ( <filter:ReferentialAttribute> )
                                               { <num:TagNumber>
                           <value:Any>
                         }
                   ( <format:ReferentialTagNumber> )


feedos_cli (User@SRV:PORT) >
lookup DAX 5 narrow ({PriceCurrency CHF}) (CFICode FOSMarketId ISIN)
instr # 498/546385 = 1044928081
    CFICode                     string{RWXXXX}
    FOSMarketId                 XQMH
    ISIN                        string{CH0107782275}
instr # 498/546384 = 1044928080
    CFICode                     string{RWXXXX}
    FOSMarketId                 XQMH
    ISIN                        string{CH0107782267}
instr # 498/546383 = 1044928079
    CFICode                     string{RWXXXX}
    FOSMarketId                 XQMH
    ISIN                        string{CH0107782259}
instr # 498/546382 = 1044928078
    CFICode                     string{RWXXXX}
    FOSMarketId                 XQMH
    ISIN                        string{CH0107782242}
instr # 498/546381 = 1044928077
    CFICode                     string{RWXXXX}
    FOSMarketId                 XQMH
    ISIN                        string{CH0107782234}
```

6. ResolveCodes : Translates the instruments codes from local code into internal code.

```
Usage:
exec REFERENTIAL.ResolveCodes <Arguments>

     <Arguments>:
                   ( <codes:PolymorphicInstrumentCode> )
                   <ignoreinvalidcodes:bool>
```

```
feedos_cli (User@SRV:PORT) > ResolveCodes ( XCME@USDU0 XCME@USDZ9 ) false
    309/310122
    309/134405
```

2.7.2 Requests related to Quotation Data

Prices and other values: status and timestamps.

```
feedos_cli > help exec quotation

available requests for 'QUOTATION' service:
     GetHistoryDaily
     GetHistoryIntraday
     SnapshotBranch_OBSOLETE
     SnapshotInstruments
     SnapshotInstrumentsL1
     SnapshotInstrumentsL2
     SnapshotMarkets
     SnapshotOneInstrumentMarketSheet_fast
     SubscribeAllStatus
     SubscribeInstruments
     SubscribeInstrumentsL1
     SubscribeInstrumentsL2
     SubscribeMarketsStatus
     SubscribeOneInstrumentMarketSheet
     SubscribeOneInstrumentOrderBook2
```

1. SnapshotMarkets : Retrieves the markets status.

```
Usage:
exec QUOTATION.SnapshotMarkets <Arguments>

     <Arguments>:
                   ( <markets:FOSMarketId> )


feedos_cli (User@SRV:PORT) > snapshotmarkets FNSE
MarketStatus for market # 443
    SessionId   = 0
    TradSesStatusEnum   = Unknown
    TradSesStartTime    = null
    TradSesOpenTime     = null
    TradSesPreCloseTime = null
    TradSesCloseTime    = null
    TradSesEndTime      = null
    TotalVolumeTraded   = 0
    LastUpdateTimestamp = null
MarketUTCTime2MarketLocalTime_minutes   =0
MarketUTCTime2ServerUTCTime_milliseconds        =0
```

2. SnapshotInstrumentsL1 : Retrieves current values (trading status, last price, volumes, best ask / bid limits, etc).

```
Usage:
exec QUOTATION.SnapshotInstrumentsL1 <Arguments>

     <Arguments>:
```

```
                     ( <codes:PolymorphicInstrumentCode> )
                     ( <values:QuotationTagNumber> )
                     <ignoreinvalidcodes:bool>


feedos_cli (User@SRV:PORT) > snapshotinstrumentsL1 XCME@USDU0
InstrumentStatusL1
-- 309/310122
        BID: 0  0       *NO ORDER*
        ASK: 0  0       *NO ORDER*
        LastPrice                       float64{79.83}
        InternalPriceActivityTimestamp  Timestamp{2009-11-30 22:48:18:598}
        TradingStatus                   17/ReadyToTrade
        DailySettlementPrice            float64{79.83}
```

3. SnapshotInstrumentsL2 : Retrieves the top best prices (usually 5) in the markets with all the orders at the same price being merged.

```
Usage:
exec QUOTATION.SnapshotInstrumentsL2 <Arguments>

     <Arguments>:
                     ( <codes:PolymorphicInstrumentCode> )
                     <orderbookdepth:int8>
                     <ignoreinvalidcodes:bool>
```

4. SubscribeInstrumentsL1 : Subscription to trade events for one or more instruments : same as Market Snapshot + continuous updating.

```
Usage:
exec QUOTATION.SubscribeInstrumentsL1 <Arguments>

     <Arguments>:
                     ( <codes:PolymorphicInstrumentCode> )
                     ( <othervaluestolookfor:QuotationTagNumber> )
                     <contentmask:QuotationContentMask>
                     <ignoreinvalidcodes:bool>


feedos_cli (User@SRV:PORT) > subscribeinstrumentsL1 XCME@USDZ9
SubscribeInstrumentL1_Started
ticket = 313868
InstrumentStatusL1
-- 309/134405
        BID: 60         1       *NO ORDER*
        ASK: 80.86      10      *NO ORDER*
        LastPrice                       float64{79.68}
        InternalPriceActivityTimestamp  Timestamp{2009-11-30 22:48:18:598}
        TradingStatus                   17/ReadyToTrade
        DailySettlementPrice            float64{79.68}
```

5. SubscribeInstrumentsL2 : Subscription to the order book depth (aka Market By Level) for one or more instrument (initial snapshot + continuous updating).

```
Usage:
exec QUOTATION.SubscribeInstrumentsL2 <Arguments>

     <Arguments>:
                   ( <codes:PolymorphicInstrumentCode> )
                   <ignoreinvalidcodes:bool>
```

6. SubscribeOneInstrumentMarketSheet : Subscription to Market Sheet (Market By Order) for one instrument: (initial snapshot + continuous updates).

```
Usage:
exec QUOTATION.SubscribeOneInstrumentMarketSheet <Arguments>

     <Arguments>:
                   <code:PolymorphicInstrumentCode>
```

2.7.3 Others Requests

CONNECTION.SubscribeToFeedStatus : Display the feed name and status and others functionalities.
The feed state can have multiple values :

- ProbablyNormal : the feed is probably normal
- Active : the feed is normal with activity
- ProbablyDisrupted : the feed is probably disrupted
- Disrupted_TechnicalLevel :the feed not usable for technical reasons
- Disrupted_ExchangeLevel :the feed not usable (N/A at exchange level)

```
feedos_cli (User@SRV:PORT) > SubscribeToFeedStatus

*** Initial Snapshot
feed name: XETRA
internal source IDs: 44
    feed state          : ProbablyNormal
    latency penalty     : false
    out of date values : false
    bad data quality    : false
feed name: VIENNA
internal source IDs: 28
    feed state          : ProbablyNormal
    latency penalty     : false
    out of date values : false
    bad data quality    : false
```

3 The quotation_export

This tool allows exporting the quotation data to a text file using some filtering options are available.

```
$ ./quotation_export
syntax:  quotation_export [OPTIONS] -server SERVER PORT LOGIN PASSWORD [MARKETS [SECTYP
[CFICODES]] [-quotdir path[-merge]]

examples:

example1: (filtering with ISO Market IDs + SecurityType)

      quotation_export -server SRV 6020 User pwd  XEUR,XLIF  FUT,OPT  > quot.txt
      ==> dump all FUTures and OPTions belonging to markets EUREX and LIFFE

example2: (filtering with CFICode)

      quotation_export -server SRV 6020 User pwd  all  all  E,MRI > quot.txt
      ==> dump all Equities and Indices belonging to all markets

example3: (filtering with ISO Market IDs + SecurityType + CFICode)

quotation_export -server SRV 6020 User pwd  XEUR  OPT  O.E > quot.txt
      ==> dump all the European style OPTions belonging to the EUREX market

      quotation_export -server SRV 6020 User pwd  XCME  FUT  FF > quot.txt
      ==> dump all the Financial Futures belonging to the XCME market

      quotation_export -server SRV 6020 User pwd  XCME  MLEG all > quot.txt
      ==> dump all the multilegs belonging to the XCME market

      MARKETS  is a comma-separated list of MIC to use as a filter, or "all"
SECTYPES is a comma-separated list of SecurityType to use as a filter, or "all"
CFICODES is a comma-separated list of CFICode to use as a filter, or "all"
GENERAL OPTIONS :
    -v                            (verbose)
    -f value1,value2...           (QUOTATION Format -- set the list of requested values)
    +f value1,value2...           (QUOTATION Format -- add some values to the default li
    +bestlimits                   (QUOTATION Format -- add best bid/ask price&qty)

file OPTIONS (this is the default if no quotdir) :
     -printff                     (use printf %f to print numbers)
     -mode MODE                   (all/visible_only/hidden_only)
     +F value1,value2...          (REFERENTIAL Format -- add some Ref. Tags)
     -o OUTFILE                   (Output file, default is stdout)
     -a APPENDFILE                (Output file, default is stdout)
     -nh                          (NoHeader, default is false unless -a)
quotdir OPTIONS:
  (Output in binary format, in given directory)
     -quotdir QUOTDATADIR         (Output in binary format, in given directory)
     -merge                       (merge result with existing data)
```

4 The referential_export

This tool allows exporting the referential data to a text file using some filtering options are available.

```
$ ./referential_export
syntax:  referential_export [OPTIONS] -server SERVER PORT LOGIN PASSWORD [MARKETS
[SECTYPES [CFICODES]]]

MARKETS  is a comma-separated list of MIC to use as a filter, or "all"
SECTYPES is a comma-separated list of SecurityType to use as a filter, or "all"
CFICODES is a comma-separated list of CFICode to use as a filter, or "all"

example1: (filtering with ISO Market IDs + SecurityType)

      referential_export -server SRV 6020 User pwd  XEUR,XLIF  FUT,OPT  > ref.txt
      ==> dump all FUTures and OPTions belonging to markets EUREX and LIFFE

example2: (filtering with CFICode)

      referential_export -server SRV 6020 User pwd  all  all  E,MRI > ref.txt
      ==> dump all Equities and Indices belonging to all markets

example3: (filtering with ISO Market IDs + SecurityType + CFICode)

referential_export -server SRV 6020 User pwd  XEUR  OPT  O.E > ref.txt
      ==> dump all the European style OPTions belonging to the EUREX market

      referential_export -server SRV 6020 User pwd  XCME  FUT  FF > ref.txt
      ==> dump all the Financial Futures belonging to the XCME market

      referential_export -server SRV 6020 User pwd  XCME  MLEG all > ref.txt
      ==> dump all the multilegs belonging to the XCME market


GENERAL OPTIONS:

     -v                        (verbose)
     -since DATE               (only instruments created/modified after this date)
     -mode MODE                (all/visible_only/hidden_only)
     -f attrib1,attrib2...     (Format -- override the list of attributes retrieved)
     -/                        (Instrument code formatted as MarketId/InstrumentId)

file OPTIONS (this is the default if no -refdir):
     -printff                  (use printf %f to print numbers)
     -o OUTFILE                (Output file, default is stdout)
     -a APPENDFILE             (Output file, default is stdout)
     -nh                       (NoHeader, default is false unless -a)
     +f attrib1,attrib2...     (Format -- add some attributes to the default list)

refdir OPTIONS:
     -refdir REFDATADIR        (Output in binary format, in given directory)
     -merge                    (same as refidr + merge with existing data)
```

5 The get_histo

This tool retrieves the instrument list of values over a period of time, if the first argument is fixed to -daily it retrieves the daily open, high, low and close price over a specified period of time, if it is -intraday retrieves the intraday trades over a specified period of time.

```
syntax:  get_histo [-daily|-intraday] OPTIONS -server SERVER PORT LOGIN PASSWORD {-|INS

example: get_histo -daily -begin 2009-01-01 -server server port user pwd XEUR@FDAX1206
    ==>  get OCHL daily values for DAX Future 2009-12, starting 2009-01-01

OPTIONS can be any combination of :

     -nb_points NB            (intraday only: NB last ticks)
     +nb_points NB            (intraday only: NB ticks sampled over the period)
     -begin BEGIN_DATE        (begin date, if it is not specify the older date available
                               the FeedOS server is the BEGIN_DATE)
     -end END_DATE            (end date, if it is not specify the date of the moment
                               is the END_DATE)
     -o OUTFILE               (Output file, default is stdout)
     -a APPENDFILE            (Output file, default is stdout)
     -nh                      (NoHeader, default is false unless -a)
     -v                       (Verbose)
```

6 The ref_varticksize_tables_export

Display the tick size tables that contain informations on the market instruments liquidity.

```
Usage:
syntax:  ref_varticksize_tables_export [OPTIONS] -server SERVER PORT LOGIN PASSWORD

OPTIONS:

     -o OUTFILE               (Output file, default is stdout)
     -a APPENDFILE            (Output file, default is stdout)


Example:
$ ./ref_varticksize_tables_export -server 84.233.192.228 6041 User pwd
TABLE # 3342336 / normal_trading
>= 0.26 : 0.01
>= 0    : 0.001
TABLE # 3342337 / high_liquidity
>= 0.26 : 0.005
>= 0    : 0.001
```

7 The tradecond_dictionary_export

Display the trade conditions dictionary that contains the market trade conditions values.

```
Usage:
syntax:  tradecond_dictionary_export [OPTIONS] -server SERVER PORT LOGIN PASSWORD

OPTIONS:

file OPTIONS (this is the default if no -refdir):
     -o OUTFILE              (Output file, default is stdout)
     -a APPENDFILE           (Output file, default is stdout)
refdir OPTIONS:
     -refdir REFDATADIR      (Output in binary format, in given directory)
     -merge                  (same as refdir + merge with existing data)

Example:
$ ./tradecond_dictionary_export -server SRV 6024 User PWD
4/100   MARKET_LSE_TradeTypeIndicator=NT
4/101   MARKET_LSE_TradeTypeIndicator=NT,MARKET_LSE_BargainConditionIndicator=Y
4/102   MARKET_LSE_TradeTypeIndicator=OK
4/103   MARKET_LSE_TradeTypeIndicator=O
4/104   MARKET_LSE_TradeTypeIndicator=OT,MARKET_LSE_BargainConditionIndicator=Y
4/105   MARKET_LSE_TradeTypeIndicator=NK
4/106   MARKET_LSE_TradeTypeIndicator=NT,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_TradeTimeIndicator=L
4/107   MARKET_LSE_TradeTypeIndicator=OT
4/108   MARKET_LSE_TradeTypeIndicator=UT
4/109   MARKET_LSE_TradeTypeIndicator=IF
4/110   MARKET_LSE_TradeTypeIndicator=NT,MARKET_LSE_TradeTimeIndicator=L
4/111   MARKET_LSE_TradeTypeIndicator=O,MARKET_LSE_TradeTimeIndicator=L
4/112   MARKET_LSE_TradeTypeIndicator=OK,MARKET_LSE_TradeTimeIndicator=L
4/113   MARKET_LSE_TradeTypeIndicator=O,MARKET_LSE_BargainConditionIndicator=Y
4/114   MARKET_LSE_TradeTypeIndicator=O,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_ConvertedPriceIndicator=Y
4/115   MARKET_LSE_TradeTypeIndicator=NK,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_TradeTimeIndicator=L
4/116   MARKET_LSE_TradeTypeIndicator=NK,MARKET_LSE_BargainConditionIndicator=Y
4/117   MARKET_LSE_TradeTypeIndicator=NK,MARKET_LSE_TradeTimeIndicator=L
4/118   MARKET_LSE_TradeTypeIndicator=O,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_TradeTimeIndicator=L
4/119   MARKET_LSE_TradeTypeIndicator=LC,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_TradeTimeIndicator=L
4/120   MARKET_LSE_TradeTypeIndicator=OT,MARKET_LSE_BargainConditionIndicator=Y,
        MARKET_LSE_TradeTimeIndicator=L
```

8 The feed_replay

Replay data from a FeedOS recording/replay server and print it on a text file.

```
Usage:
$ ./feed_replay [-v] SERVER PORT LOGIN PASSWORD SRC_ID {L1|L2|L12|IBAR|MBO|STATUS|MUXED
    BEGIN_DATE END_DATE [parameters]

arguments:
REPLAY_SERVER PORT LOGIN PASSWORD SRC_ID {L1|L2|L12|IBAR} BEGIN_DATE END_DATE [optional
     L1    Trades, best bid/ask and status/values updates
     L2    Order book updates aka Market By Limit
     L12   Mixed stream of L1 and L2
     MBO   Market Sheet Data aka Market By Order
     MUXED Replay the L1 and L2 (it depends on the stored data format on the server)
     IBAR  Intraday Bars

OPTIONAL PARAMETERS:
     -trades_only           display only the trades without Bid/Ask (for L1)
     -cache_book            display the entire order book (for L2)
     -nh                    no header
     -cache_best_limits     always print latest (Bid/Ask)
     -nic                   display instrument codes as raw numerical values
     -mic                   filter on these MICs
     -instr_filter_args     filter on these instruments (read from arguments)

example1:

  ./feed_replay.exe SRV PORT User pwd 50 L1 "2006-12-01 08:15:00" "2007-01-30 18:59:00"
  => replay L1 stream (trades+best limits) from source 50 (Eurex) on the given period

example2: (filtering -- available only on L1)

  ./feed_replay.exe SRV PORT User pwd 50 L1 "2006-12-01 08:15:00" "2007-01-30 18:59:00"
    -trades_only XEUR@FDAX0608 XCME@FGBL1208
  => get only trades for the given future contracts
```