S&P Capital IQ Real-Time Solutions

FeedOS™ FAQ

Document Version 3.1

Reference n°: 20150609 – 27213

S&P CAPITAL IQ
McGRAW HILL FINANCIAL

S&P Capital IQ Real-Time Solutions
FeedOS™ FAQ
Reference 20150609 – 27213
June 09, 2015

France
52 Rue de la Victoire
75009 Paris
France
Tel: +33 (0) 1 73 02 32 11

United States
55 Water Street, 44th floor
New York, NY 10041
United States of America
Tel: +1-(212)-438-4346

130 East Randolph
One Prudential Plaza, Suite 2900
Chicago, IL 60601
United States of America
Tel: +1-(312)-233-7129

United Kingdom
20 Canada Square
Canary Wharf
London E14 5LH
United Kingdom
Tel: +44 (0) 203 107 1676

Singapore
12 Marina Boulevard
#23-01 Marina Bay
Financial Centre Tower 3
Singapore 018982
Tel: +65 6530 6546

www.spcapitaliq.com

TABLE OF CONTENTS

FEEDOS™ FREQUENTLY ASKED QUESTIONS

1. Introduction

As part of the S&P Capital IQ Real-Time Solutions FeedOS™ documentation set, this Frequently Asked Questions document provides you with answers to a collection of common questions regarding FeedOS technology and its implementation.

Furthermore, this FAQ also covers particular aspects related to the standards and protocols deployed in real-time electronic exchange and market data stream integration.

The major topics in this document include:

1.1. Who Is This Document For?

This document is primarily intended for use by software engineers, developers and other team members planning to use or using S&P Capital IQ Real-Time Solutions FeedOS middleware technology. This guide also addresses issues and topics of interest to any person planning to develop or developing software that interacts with FeedOS API in particular.

1.2. What Do You Need to Use This Document?

To grasp the concepts and use this document to its full extent, general knowledge of market data and its deployment is mandatory. A good understanding of financial markets and instruments, including standards and protocols, is recommended as well. Additionally, general knowledge of **Application Programming Interface** (API) functioning and specifications is also required.

1.3. What's New in This Document?

The current version of this document includes the following changes:

Table 1 FeedOS™ FAQ – Document History

Release date	Version	Changes
2015-06-09	3.1	Removed references to deprecated documents, updated contact details and Web addresses.
2014-09-09	3.0	Revised and improved answers in sections 2. Financial Terminology, 3. FeedOS Principles and 8. Handling Market Data.
2012-08-06	2.0	Updated sections 3. FeedOS Principles, 6. Exchange Feeds and 8. Handling Market Data. Changed the document structure and layout.
2011-11-18	1.3	Updated answers to accommodate the changes in FeedOS Framework 3.4.0.
2011-02-16	1.2	Added section 9. Troubleshooting Connection Problems.
2010-09-13	1.1	Added questions 6.5. How do you identify Off Book Trades? to 6.8. Why do I receive strange characters in the tags values?.
2010-06-10	1.0	Creation of the document.

1.4. Getting Additional Help and Support

For the latest documentation and product updates, additional support and training, please contact our support services:

E-mail:

rts-support@spcapitaliq.com

Web:

https://support.quanthouse.com

2. Financial Terminology

This section provides you with details about the financial terminology and any specific meaning it may have with regard to S&P Capital IQ Real-Time Solutions products and services.

2.1. What is the CFI Code?

The **CFI Code** is the ISO 10962 standard code that classifies financial instruments.

See also:

- http://en.wikipedia.org/wiki/ISO_10962

2.2. What is the MIC Code?

The **Market Identifier Code (MIC)** is the ISO 10383 standard code that identifies exchanges and financial markets.

See also:

- http://www.iso15022.org/MIC/homepageMIC.htm

2.3. What is Level1 Data? BBO? Top of the Book? Best Limits?

Level1 (L1) Market Data or Level 1 Quotation Data consists of the real-time Best Bid and Best Ask Prices and the Bid and Ask Sizes available at those prices.

The Best Bid and Best Ask Prices are also known as **Best Bid Offer (BBO)** and **Top of the Book**.

In FeedOS, Level1 Market Data contains the BBO data, the latest trade prices, trade-related details, the trading status and other values, such as Open, Close, High, Low etc.

2.4. What is Level2 Data? MBO? MBL? Depth of the Market?

Level2 (L2) Market Data or Level 2 Quotation Data is the list of all Bid and Ask Prices with sizes and order originators. This is also known as the **Depth of Market** and **Market by Order (MBO)**.

Market by Limit (MBL) is an abbreviated version of the Depth of Market which provides the top 5 or 10 price levels on each side (Bid and Ask) with the number of orders and total volume at each level.

2.5. What is Price Step? Tick Size? Price Increment?

Price Step, **Tick Size** and **Price Increment** represent the minimum amount by which a quote for a financial instrument can change. Where this information is provided by an exchange, it is available in the Referential Data, via the following tags: `PriceIncrement_static` and `PriceIncrement_dynamic_TableId` (in the Dynamic tableID).

2.6. What is Contract Size?

Contract Size is the quantity of an underlying security that the holder of a derivative has the right to buy or sell. For most equity options, the contract size is 100 shares (unless a split or other special circumstances occur). Where this information is provided by the exchange, the data is available in the `ContractMultiplier` referential tag.

2.7. To what currencies do the symbols GBp and GBX correspond?

The codes `GBp` and `GBX` are both used by to signify *British Pence*. The code `GBP` stands for *British Pound*.

2.8. To what currency does the symbol USX correspond?

The code `USX` represents the *Dollar Cent*. The code `USD` represents the *US Dollar*.

2.9. To what currency does the USR symbol correspond?

The code `USR` is used by some exchanges, such as RTS, to indicate that the Price Step Value is calculated in Russian Rubles using the US Dollar exchange rate. This is based on MICEX USD/RUB rate when MICEX is open, and on Reuters rate when USD/RUB currency pair trading session on MICEX is closed.

3. FeedOS Principles

This section gives you an overview of the principles underlying the FeedOS technology.

3.1. What is the Internal Code?

The **Internal Code** is a numeric value generated by S&P Capital IQ Real-Time Solutions to uniquely identify an instrument.

The Internal Code consists of two parts:

- the parent Market ID and
- the unique identifier within this market.

For instance, the Internal Code `295/750273` is composed of `295` – the Market ID in internal numeric format, and `750273` – the numeric code for the Vodafone Ordinary Share listed on the London Stock Exchange (ISIN `GB00BH4HKS39`).

3.2. What is the Local Code?

The **Local Code** is a proprietary, market-dependent string, computed from a subset of the instrument's characteristics, such as Symbol, Expiry Date, etc. The Local Code can also be used to uniquely identify an instrument.

The Local Code consists of two parts, separated by the "`@`" symbol:

- the parent Market ID and
- the `LocalCodeStr`, which is the instrument listing identifier on the given market.

For instance, the Local Code `XLON@GB_GB00BH4HKS39` is composed of `XLON` – the Market ID in ISO format, and `GB_GB00BH4HKS39` – which is the unique identifier for the Vodafone Ordinary Share listed on the London Stock Exchange.

3.3. What are the pros and cons of the two identifiers?

We recommend you implement in your application algorithms that derive from the referential data a list of instruments based on their fundamental characteristics rather than referencing instrument identifiers. By doing so, your application will correctly manage any delisted and new instruments from one day to the next.

The **Internal Code** has the following advantages:

- **Consistency** – it is not affected by changes that may occur in an instrument's characteristics
- **Performance** – it is easier to process
- **Precision** – unlike symbol-based codes, the Internal Code helps you easily and unambiguously reference an instrument

while the **Local Code** has these advantages:

- **Usability** – it is human readable and easier to understand
- **Similarity** – it resembles (and sometimes duplicates) the system of symbols that exchanges use.

Thus, if you are looking for performance, the list of instruments provided as an input parameter for your subscription requests should use the **Internal Codes**.

However, if you need to store instrument-related data on a disk or in a database, we strongly recommend you use both representations. This way it is easier to identify instruments if either code changes following an exchange or technical migration or when the exchange delists the security.

See also:

- questions 3.9. What are the Fundamental Characteristics of a Financial Instrument?, 3.14. What is Referential Data?, 8.3. How do I manually search for instruments?, 8.5. How do I export the referential data?
- *FeedOS API Guide*

3.4. Can I translate an instrument's Local Code into its Internal Code?

Yes, you can translate a list of Local Codes into Internal Codes by using the **FeedOS API request** `REFERENTIAL_ResolveCodes` **or** the equivalent command `ResolveCodes` of the **tool FEEDOS_CLI, available in FeedOS Toolbox**.

See also:

- section 7. FeedOS API and FeedOS Toolbox – Features and Options
- *FeedOS API Guide*
- *FeedOS Toolbox User's Guide*

3.5. What is the InternalSourceId?

The **`InternalSourceId`** is a tag in an instruments' Referential State and the Feed Status Update that indicates the Feed Internal ID.

3.6. What is the relationship between a MIC and an InternalSourceId?

A **MIC can be associated to multiple** `InternalSourceId`s. For instance, the feed LIFFE is split into several sources.

Conversely, an `InternalSourceId` **can be related to multiple MICs**. For example, Euronext source `95` consists of several MICs, including `XPAR`, `XBRU`, `XAMS` etc.

3.7. What is the relationship between an Exchange and a MIC?

An **Exchange can be associated with multiple MICs**. For instance:

- CME has `XCME`, `XCBT`, `XCEC` and `XNYM`

- Euronext has XPAR, XBRU, XLIS, XAMS, ALXB, ALXP, XMLI, etc.
- Borsa Italiana has MTAA and ETFP.

3.8. Can I match up an InternalSourceId with an instrument?

The best way to match up an `InternalSourceId` with an instrument is to use the **FeedOS API function** `CONNECTION_SubscribeToFeedStatus_delegate`.

You can also use the **request** `feed_status` **of the tool FEEDOS_CLI, available in FeedOS Toolbox**, as shown in the example below:

```
FEEDOS_CLI (username@serverIP:port) > feed_status

*** Initial Snapshot
feed name: CME
internal source IDs: 17
    feed state         : Active
    latency penalty    : false
    out of date values : false
    bad data quality   : false
feed name: EUREX_FUTURES
internal source IDs: 52
    feed state         : Active
    latency penalty    : false
    out of date values : false
    bad data quality   : false
feed name: LIFFE_interest_rates
internal source IDs: 174
    feed state         : Active
    latency penalty    : false
    out of date values : false
    bad data quality   : false
feed name: LIFFE_equities
internal source IDs: 178
    feed state         : Active
    latency penalty    : false
    out of date values : false
    bad data quality   : false

[...]
```

See also:

- *FeedOS API Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

3.9. What are the Fundamental Characteristics of a Financial Instrument?

The **Fundamental Characteristics** of an instrument are the values that describe the instrument, such as:

- `FOSMarketId` (see question 3.13. What is the FOSMarketId?)
- `Symbol`
- `SecurityType`
- `Currency`
- `CFICode` (see question 2.1. What is the CFI Code?)
- `StdMaturity`, etc.

They may vary according to the asset class and source.

3.10. Why does FeedOS not use ISIN to identify Financial Instruments?

The ISIN is a security level identifier which is shared by all listings for that security. Therefore **it does not uniquely identify** an instrument at the listing level. For example, the ISIN US0003752047 is shared by ordinary share listings of ABB Ltd trading on NYSE, Xetra, Deutsche Borse and Bolsa Mexicana de Valores.

3.11. What if I already have my own instrument identification system? Is there a way to match it up with the referential data that FeedOS provides?

Yes, to match up your own instrument identification system with the referential data provided by FeedOS, **use a combination of normalized tags that serve as unique and stable identifiers**.

Depending on the instrument type, use the following combinations:

- **Equities:** `Exchange` + (`ISIN` or `Symbol` or `CUSIP` or `SEDOL`) + `PriceCurrency`
- **Derivatives:** `Exchange` + (`SecurityType` and/or `CFICode`) + `Symbol` + `StdMaturity` [+ `StrikePrice` + `OptNumVersionContract` for Options].

For instance, to map European equities, use a combination of `PriceCurrency` + `ISIN`. To map the MTFs, the tag `ForeignFOSMarketId` could help you identify the primary exchange.

When mapping FeedOS referential data to your instrument identification system, please keep in mind that an exchange can contain multiple MICs (see question 3.7. What is the relationship between an Exchange and a MIC?).

See also:

- *FeedOS API Guide*

3.12. How does FeedOS manage the identification of financial markets?

To identify financial markets, FeedOS uses the **MIC Code**, which is internally processed from the ISO 10383 standard code into a numeric identifier.

See also:

- question 2.2. What is the MIC Code?

3.13. What is the FOSMarketId?

The `FOSMarketId` is a Referential Tag that stores a numeric value for market identification.

3.14. What is Referential Data?

Referential Data (also called "Static Data" or "Dictionary") provides constant or static details about financial instruments and market characteristics, i.e. data that does not change during a trading session. In the FeedOS API, these details are disseminated through values in the **Referential Attributes** tags.

3.15. What is Quotation Data?

Quotation Data provides dynamic details about the status and price of an instrument, i.e. data that changes during a trading session. In the FeedOS API, these details are disseminated through values in the **Quotation Variables** tags.

See also:

- *FeedOS Quotation Tags Guide*

3.16. What is the difference between Referential and Quotation Data?

As explained in questions 3.14. What is Referential Data? and 3.15. What is Quotation Data?, Referential Data does not change during a trading session, while Quotation Data changes during a trading session.

3.17. To what extent do you normalize the Referential Data?

S&P Capital IQ Real-Time Solutions **normalizes the Referential Data** as much as possible. FeedOS has implemented several general international standards and protocols (such as *ISO 10383 MIC* – which identifies the markets, *ISO 10962* – which indicates the CFI Code or the type of security, and *FIX Protocol*), as well as programing conventions, relevant to each development environment.

3.18. To what extent do you normalize the Quotation Data?

S&P Capital IQ Real-Time Solutions **converts timestamps to UTC and non-decimal prices** (such as those received from CME or CBOT) **to decimal**. However, trade conditions and market specific tags are not changed.

3.19. What is a Referential Data Branch?

A **Referential Data Branch** is a set of instruments grouped by Market ID, `SecurityType` and `CFICode`.

3.20. What are Trade Conditions?

Trade Conditions provide details about the type of trades (such as Electronic, OTC, etc.).

See also:

- *FeedOS API Guide*
- *FeedOS Quotation Tags Guide*

4. FeedOS System Architecture

This section answers questions concerning the architecture of FeedOS.

4.1. What is a Subscription Server?

A **Subscription Server** aggregates feeds from several Feed Handlers.

4.2. What is a Feed Handler?

A **Feed Handler** is a system that normalizes the data it receives from incoming market feed sources.

4.3. What is a Publication Server?

A **Publication Server** is a specific type of system that publishes an entire feed or enriches the available market data.

4.4. How do I connect to a Subscription Server?

You can connect to a Subscription Server by using the **FeedOS API** or **FeedOSToolbox**.

See also:

- section 7. FeedOS API and FeedOS Toolbox – Features and Options
- *FeedOS API Guide*
- *FeedOS Toolbox User's Guide*.

4.5. What is the first thing I should do after connecting to a Subscription Server?

After you connect to a Subscription Server, you can **download the Referential Data and subscribe to different instruments**.

4.6. What type of protocol do you use between a Subscription Server and a Feed Handler? And between the user's API and a Subscription Server?

The Subscription Server and the Feed Handler communicate via a **multicast protocol**.

The Subscription Server and the FeedOS API communicate via a **TCP or IPC protocol**.

See also:

- *FeedOS API Guide*

4.7. Do you redistribute all the Level2 Market Data you receive from an Exchange?

S&P Capital IQ Real-Time Solutions sends **all the Level2 Market Data** it receives from the exchanges. Depending on your entitlements, you may receive all the Level 2 Market Data or the top 5 / 10 levels or no Level 2 Data.

4.8. Do you aggregate the Level2 Market Data you receive from an Exchange?

S&P Capital IQ Real-Time Solutions **does not aggregate the Level2 Market Data** it receives, but it does limit the depth of the book, to the top 5 or 10 best prices dependent on individual exchange rules.

See also:

- question 2.4. What is Level2 Data? MBO? MBL? Depth of the Market?

4.9. Do you keep a Reference Database?

Yes, S&P Capital IQ Real-Time Solutions keeps a Reference Database. It is called the **Referential Database**.

4.10. How often do you update the Reference Database?

The update frequency of the Referential Database depends on the exchange. To receive the market notifications in real-time, use **FeedOS API function** `REFERENTIAL_Download and Subscribe`.

See also:

- section 8. Handling Market Data
- *FeedOS API Guide*

5. FIX Protocol Usage

This section details the implementation of the FIX Protocol in the FeedOS API.

5.1. What is the Standard FIX Protocol?

The **Financial Information eXchange (FIX) Protocol** is a messaging standard developed specifically for the real-time electronic exchange of securities transactions. For consistency purposes, S&P Capital IQ Real-Time Solutions has adopted and implemented some of the FIX Protocol definitions in the FeedOS API.

However, please note that the FIX Protocol is text-oriented and therefore longer, whereas the FeedOS API Protocol is binary and optimized, which means it has a variable-length encoding, implicit values, etc. Hence, subscribing through FIX Protocol requires more network bandwidth and processing power on both sides (server and API) than subscribing via FeedOS API.

5.2. Which parts of the FIX Protocol do you support?

FeedOS Subscription Servers can download the **referential data, subscribe to trades, MBL orders and real-time events**.

See also:

- *FeedOS FIX Protocol Guide*
- the examples in the archive *Fix v4.4 Sample Messages*, available at https://mypage.quanthouse.com/

5.3. What version of the FIX Protocol do you support?

The FeedOS API currently supports **versions 4.3 and 4.4** of FIX.

5.4. Do changes occur in your implementation of the FIX Protocol?

Although S&P Capital IQ Real-Time Solutions endeavours to implement FIX as strictly as possible, **minor deviations may occur**. However, changes will only ever be minor enough not to affect the client-server communication.

See also:

- *FeedOS FIX Protocol Guide*
- the examples in the archive *Fix v4.4 Sample Messages*, available at https://mypage.quanthouse.com/

6. Exchange Feeds

The following section provides you with answers related to the S&P Capital IQ Real-Time Solutions market data feeds.

6.1. Is there any way of telling by looking at a feed what session a message is from (such as pre-opening indicative trades differentiated from the main trades)?

Yes, the **Open/Close signals and the trading status indicate the session**. Moreover, where these are available from the exchange, S&P Capital IQ Real-Time Solutions sends the Auction Price and Volume Price via the quotation tags `LastAuctionPrice` and `LastAuctionVolume`.

See also:

- *FeedOS Quotation Tags Guide*

6.2. Are session transitions distinctly marked in the feed?

Yes, where available from the exchange, the session transitions are indicated **by the quotation tags** `TradingStatus` and `CurrentBusinessDay`.

See also:

- *FeedOS Quotation Tags Guide*

6.3. Are trading halts/unexpected closures signaled through the feed?

Yes, where available from the exchange, trading halts and unexpected closures are indicated **by the quotation tag** `TradingStatus` and other market specific tags.

See also:

- *FeedOS Quotation Tags Guide*

6.4. Does the feed indicate non-trading days and holidays?

No, it does not.

6.5. How do you identify Off Book Trades?

Depending on the exchange, OTC trades are identified **by using market-specific tags** and the bit `OffBookTrade` of the **content mask**.

See also:

- *FeedOS Quotation Tags Guide*

6.6. How can I identify an Auction phase?

Depending on the feed handler, an auction phase is identified through **simple or normalized quotation tags**, such as: `TradingStatus`, `LastAuctionPrice` and `LastAuctionVolume`.

See also:

- *FeedOS Quotation Tags Guide*

6.7. Is Market News available in the feeds? If so, how do I retrieve it?

Yes, where available from the exchange, FeedOS feeds disseminate Market News.

6.8. Why do I receive strange characters in the tags values?

This happens because of the **UTF-8 encoding**, which is used for some tag values.

6.9. Is it normal to have a large Bid-Ask Spread on the CTA and NASDAQ UTP feeds?

Yes, **it is normal**, as the buyer or seller uses the Bid-Ask Spread to avoid the National Best Bid and Offer (NBBO).

6.10. Why are some fundamental characteristics of an instrument missing on the CTA and NASDAQ UTP feeds?

Some fundamental characteristics, such as `SEDOL` and `ISIN` codes, are not available by default on these feeds. To obtain these, please contact the S&P Capital IQ Real-Time Solutions Project Manager in charge of your market data subscription.

6.11. Why are small trades (odd-lots) missing from the consolidated US markets, such as CTA and NASDAQ UTP?

Small trades (odd-lots) are available only on primary markets, such as NASDAQ TotalView, NYSE Trades and NYSE Arca Trades.

7. FeedOS API and FeedOS Toolbox – Features and Options

This section details the main characteristics of FeedOS API and FeedOS Toolbox.

7.1. In which programming languages is the API available?

The FeedOS API is currently available in **C++**, **C#/.NET** and **Java**. The main features of the API (such as Subscriptions) are also available in **FIX Protocol**.

7.2. What are the differences between the API versions?

There are slight differences between the three versions of the FeedOS API:

- **C++ API** is the most robust and fastest of the three versions.
- **C#/.NET API** comes as a wrapper around the C++ version.
- **Java API** is the lightest version.

7.3. Which API and environment do you recommend?

S&P Capital IQ Real-Time Solutions recommends you use the **C++ FeedOS API** on **Linux 64-bit**.

See also:

- *FeedOS API Guide*

7.4. Which development environments do you support?

You can install the FeedOS API on several environments, including **different versions of Microsoft® Windows®, Unix and Unix-like** operating systems.

See also:

- *FeedOS API Guide*

7.5. How often do you upgrade the API? How do I get the latest version?

S&P Capital IQ Real-Time Solutions upgrades the FeedOS API **quite often**. As a client, you are **automatically notified** when a new version of the API is released.

To obtain the latest version of the FeedOS API, connect to the **S&P Capital IQ Real-Time Solutions Resource Repository**, available at https://download.quanthouse.com/

7.6. Do you support previous API versions?

Although S&P Capital IQ Real-Time Solutions still supports previous versions of the FeedOS API, we strongly suggest you **use the latest available version**.

7.7. What is the Toolbox package? Is it part of the API?

The FeedOS Toolbox package is a **stand-alone set of command-line tools**. Although it is **not part of the FeedOS API**, the FeedOS Toolbox offers similar functionality and helps you discover and understand the underlying principles of the FeedOS API.

Thus, with the FeedOS Toolbox you can query and download the referential and quotation data available on a Subscription Server, check the feed status, subscribe to one or more instruments and perform other operations, similar to the functions in the applications you develop by integrating the FeedOS API.

See also:

- *FeedOS Toolbox User's Guide*.

7.8. Where can I get the Toolbox package?

The FeedOS Toolbox and the associated documentation are available at https://download.quanthouse.com/

7.9. Is the Toolbox compatible with all the environments?

The FeedOS Toolbox is available for the following platforms:

- Microsoft® Windows® 32-bit
- Linux 32-bits and 64-bit
- Apple® Mac OS® 64-bit.

7.10. What is FEEDOS_CLI?

FEEDOS_CLI is a command-line client available as a tool in the FeedOS Toolbox. FEEDOS_CLI helps you send the same type of requests (with the same features, names and parameters) you can send in the C++ FeedOS API.

See also:

- *FeedOS Toolbox User's Guide*

7.11. What else is in the Toolbox?

The FeedOS Toolbox also includes other tools, such as `referential_export`, `quotation_export`, `get_histo`, `ref_varticksize_tables_export`, `tradecond_dictionary_export`, `feed_replay` etc.

See also:

- section 8. Handling Market Data
- *FeedOS Toolbox User's Guide*

8. Handling Market Data

This section provides information on handling the market data available on a Subscription Server.

8.1. What is the best way to handle market and instrument data available on a Subscription Server?

The best way is to export/download the data you need by using the functions and features available in the FeedOS API and then process it as you see fit.

See also:

- the following questions in this section
- *FeedOS API Guide.*

8.2. How do I find out what markets (and branches) are available on a Subscription Server?

The best way to get details on the markets and branches available on the data feed is to **use the** `REFERENTIAL_DumpStructure` **function of the FeedOS API**.

To retrieve the attributes of a specific market, **use the** `REFERENTIAL_GetMarkets` **function of the FeedOS API**.

You can also **use the** `REFERENTIAL.DumpStructure` and `REFERENTIAL.GetMarkets` **requests of the FEEDOS_CLI tool, available in the FeedOS Toolbox**, as shown in the example below:

```
FEEDOS_CLI (username@serverIP:port) > dumpstructure
```

See also:

- *FeedOS API Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

8.3. How do I manually search for instruments?

The best way to get details on a specific instrument available on the data feed is to **use the FeedOS API function** `REFERENTIAL_Lookup`. The service returns multiple results depending on the input criteria. This function is more suitable for interactive use.

You can also **use the** `REFERENTIAL.Lookup` **request of the FEEDOS_CLI tool, available in the FeedOS Toolbox**, as shown in the example below:

```
### search for two instruments, whose Symbol is VOD and StdMaturity is september 2012
FEEDOS_CLI (username@serverIP:port) > lookup VOD 2 narrow ({StdMaturity 201209})
```

See also:

- *FeedOS API Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

8.4. How do I get details about an instrument?

The dedicated service to access instruments by their identifier in the data feed is **the** `REFERENTIAL_GetInstruments` **function of the FeedOS API**.

You can also **use the** `geti` **tool available in the FeedOS Toolbox**, as shown in the example below:

```
### display the referential details of the specified instrument
FEEDOS_CLI (username@serverIP:port) > geti XNAS@AAPL
or
FEEDOS_CLI (username@serverIP:port) > geti 12345678
or
FEEDOS_CLI (username@serverIP:port) > geti 12/12345
```

See also:

- *FeedOS API Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

8.5. How do I export the referential data?

The best way to export the referential data available on a Subscription Server is to **use the** `REFERENTIAL_Download` **function of the FeedOS API.**

You can also **use the** `referential_export` **tool available in the FeedOS Toolbox**, as shown in the example below:

```
### download all Futures from CME and Eurex
referential_export -server IP PORT LOGIN PASSWORD XCME,XEUR FUT all

### download all Equities and Indices from all exchanges
referential_export -server IP PORT LOGIN PASSWORD all all E,MRI
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.6. Can I perform incremental downloads of the referential data?

Yes, **by setting the** `LastUpdateTimestamp` **parameter** of the `REFERENTIAL_Download` **request of the FeedOS API** to the value of the last update timestamp the Subscription Server returned when performing the previous download.

See also:

- *FeedOS API Guide*

8.7. How do I detect the creation and/or the modification of a financial instrument?

The best way to detect changes in the final instrument inventory is to **specify the** `LastUpdateTimestamp` **parameter of the** `REFERENTIAL_Download` **request of the FeedOS API**.

You can also **use the** `-since` ***DATE*** **option with the** `referential_export` **tool available in the FeedOS Toolbox**, which sends the instruments created or modified after the specified date, as shown in the example below:

```
referential_export -since 2012-05-01 -server IP PORT LOGIN PASSWORD
```

See also:

- questions 7.7. What is the Toolbox package? Is it part of the API? and 8.5. How do I export the referential data?
- *FeedOS API Guide*
- *FeedOS Toolbox User's Guide*

8.8. How do I export the quotation data?

The best way to export the quotation data available on a Subscription Server is to **use the** `QUOTATION_Download` **function of the FeedOS API**.

You can also **use the** `quotation_export` **tool available in the FeedOS Toolbox**, as shown in the example below:

```
### download all Futures and Options from EUREX and LIFFE
quotation_export -server IP PORT LOGIN PASSWORD XEUR,XLIF FUT,OPT

### download all Futures from XCME market
quotation_export -server IP PORT LOGIN PASSWORD XCME FUT FF
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.9. What is the difference between QUOTATION_DownloadInstruments, QUOTATION_SnapshotInstrumentsL1 and QUOTATION_SubscribeInstrumentsL1?

The FeedOS API function `QUOTATION_DownloadInstruments` retrieves a static snapshot of the values for all (or some groups of) instruments in general (not necessarily known) at the download time.

`QUOTATION_SnapshotInstrumentsL1` gets a static snapshot of the values for one (or several) known instrument(s) in particular has (have) at the snapshot time.

`QUOTATION_SubscribeInstrumentsL1` retrieves in real-time all the values that one (or several) known instrument(s) in particular has (have) at the subscription time plus any other subsequent update of the values during the trading day, until you terminate the subscription.

See also:

- *FeedOS API Guide*

8.10. To how many instruments can I subscribe?

You can subscribe to as many instruments as you want, but make sure you do not exceed **10,000 instruments per batch**.

Still, you can **bypass this limit** and subscribe to even more instruments, **by using the** `QUOTATION_ChgSubscribeInstrumentsL1Add` **function of the FeedOS API**.

However, please keep in mind that this function has the same limit. Hence, you should not add more than 10,000 instruments to an existing subscription batch.

See also:

- *FeedOS API Guide*

8.11. How do I export the dynamic tick size tables?

The best way to export the dynamic tick size tables available on a Subscription Server is to **use the** `REFERENTIAL_GetVariablePriceIncrementTables` **function of the FeedOS API**.

You can also **use the** `ref_varticksize_tables_export` **tool available in the FeedOS Toolbox**, as shown below:

```
ref_varticksize_tables_export -server IP PORT LOGIN PASSWORD
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.12. How do I export the trade conditions dictionary?

The best way to export the trade conditions dictionary available on a Subscription Server is to **use the** `REFERENTIAL_GetTradeConditionsDictionary` **function of the FeedOS API**.

You can also **use the** `tradecond_dictionary_export` **tool available in FeedOS Toolbox**, as shown below:

```
tradecond_dictionary_export -server IP PORT LOGIN PASSWORD
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.13. How do I retrieve an instrument's historical values?

The best way to get an instrument's historical values available on a Subscription Server is to **use the** `QUOTATION_GetHistoryDaily` **function of the FeedOS API**.

You can also **use the** `get_histo` **tool available in the FeedOS Toolbox**, as shown in the example below:

```
### get Open Close High Low daily values for DAX Future, starting 2012-01-01
get_histo -daily -begin 2012-01-01 -server IP PORT LOGIN PASSWORD XEUR@FDAX1206
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.14. How do I retrieve the daily volume traded quantity?

The best way to retrieve the daily volume traded quantity and the price is to subscribe to Level1 and retrieve the values provided by the `LastTradeQty` and `LastPrice` **quotation tags of the FeedOS API**.

You can also use:

- the `QUOTATION.SnapshotInstrumentsL1` **request** – for the current volume and price of the instruments
- the `QUOTATION.SubscribeInstrumentsL1` **subscription** – for the evolution of volumes and prices

  of the tool FEEDOS_CLI, available in the FeedOS Toolbox, as shown in the example below:

```
FEEDOS_CLI (username@serverIP:port) > snapshotinstrumentsL1 XCME@USDU0

OR

FEEDOS_CLI (username@serverIP:port) > subscribeinstrumentsL1 XCME@USDZ9
```

See also:

- *FeedOS API Guide*
- *FeedOS Quotation Tags Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

8.15. How do I retrieve the trading status?

The best way to handle the trading status is to subscribe to L1 and retrieve the values provided by the `TradingStatus` **quotation tag of the FeedOS API**.

You can also use:

- the `QUOTATION.SnapshotInstrumentsL1` **request** – for the current trading status
- the `QUOTATION.SubscribeInstrumentsL1` **subscription** – for the evolution of the trading status

  of the tool FEEDOS_CLI, available in the FeedOS Toolbox, as shown in the example below:

```
FEEDOS_CLI (username@serverIP:port) > snapshotinstrumentsL1 XCME@USDU0

OR

FEEDOS_CLI (username@serverIP:port) > subscribeinstrumentsL1 XCME@USDZ9
```

See also:

- *FeedOS API Guide*
- *FeedOS Quotation Tags Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

8.16. How do I replay data?

The best way to replay the data available on a Subscription Server is to **use the** `QUOTATION_MulticastData` **function of the FeedOS API**.

You can also **use the** `feed_replay` **tool available in the FeedOS Toolbox**, as shown in the example below:

```
### get Open Close High Low daily values for DAX Future, starting 2012-01-01
get_histo -daily -begin 2012-01-01 -server IP PORT LOGIN PASSWORD XEUR@FDAX1206
```

See also:

- *FeedOS API Guide*
- question 7.7. What is the Toolbox package? Is it part of the API?
- *FeedOS Toolbox User's Guide*

8.17. What are the Order Book Magic Numbers?

FeedOS handles prices as numeric values. To specify particular cases for limit prices, such as Open, Close etc., you should use **Magic Numbers**, which behave as special numeric values directly linked to a specific quotation situation. Currently, the magic numbers include:

- **Unquoted** – sent instead of prices that are zero
- **AT_BEST** – for At-Best prices
- **AT_OPEN** – for Limit-On-Open-Order prices
- **PEG** – for PEG prices.

See also:

- *FeedOS API Guide*

8.18. Are intra-day changes (corporate actions) available?

Currently, this feature is **not implemented**.

8.19. Are trade cancellations/corrections available?

Currently, this feature is **not implemented**.

9. Troubleshooting Connection Problems

This section gives you details about troubleshooting the API connection problems.

9.1. Do I need to make any changes to my current system to be able to connect to a Subscription Server?

Before installing the FeedOS API, **make sure your system meets the recommended requirements** – hardware, network, operating system, and additional third-party products and libraries, as detailed in *FeedOS API Guide*.

9.2. How can I avoid latency issues?

To avoid any latency issues, **make sure your system meets the optimum hardware requirements**, as detailed in *FeedOS API Guide*.

9.3. Why does my API disconnect?

Usually, the FeedOS Client API disconnects because **the application is not processing the data it receives fast enough** (to put it differently, there is too much data queued on the server side). You should review your code to identify any reasons for slow performance, particularly in the *user threading scheme*.

See also:

- *FeedOS API Guide*

9.4. How can I avoid message enqueuing?

To avoid message enqueuing on the server side, **make sure you have enough bandwidth**.

Also, in the C++ version of the FeedOS API, **check the user threading scheme for concurrent threads** running on the client side.

See also:

- *FeedOS API Guide*

9.5. How do I subscribe to the Feed Status?

The best way to subscribe to the Feed Status and display the feed name, the list of internal source IDs and its status is to **use the** `CONNECTION.SubscribeToFeedStatus` **command of the FeedOS API**.

You can also **use the** `CONNECTION.SubscribeToFeedStatus` **command of the tool FEEDOS_CLI, available in the FeedOS Toolbox**, as shown in the example below:

```
FEEDOS_CLI (username@serverIP:port) > SubscribeToFeedStatus
```

See also:

- *FeedOS API Guide*
- questions 7.7. What is the Toolbox package? Is it part of the API? and 7.10. What is FEEDOS_CLI?
- *FeedOS Toolbox User's Guide*

9.6. What happens after I subscribe to the Feed Status?

After you subscribe to the Feed Status, you **receive FeedStatusUpdate notifications** via the `notif_FeedStatusUpdate` callback. If a disruption in the service occurs, all the instruments related to the feed IDs are impacted.

INDEX

G

H