dxFeed Market Data: Instrument Profile Format

August 2019

Information in this document is subject to change without notice and does not represent a commitment on the part of dxFeed.

Please visit www.dxfeed.com for latest information updates.

Contents

1. Instrument Profile Format

1.1. Introduction

This document describes an open format to represent and exchange basic profile information about market instruments. It is aimed to cover all types of tradable and indicative instruments, including stocks, funds, bonds, indexes, futures, options, and other. The format provides common framework to represent necessary information, defines data model, and describes file format for data distribution and exchange.

1.2. Data Model

Every market instrument is represented by a single profile record as a set of field values which define corresponding attributes of the instrument. The set of defined fields and interpretation of their values form the abstraction layer of the format. Each field is characterized by its name, meaning, applicability to specific instrument types, interpretation rules and data format. For the purpose of readability and interoperability all values are represented in textual form even if their primary use is non-textual (e.g. numbers and dates). All values must use Unicode standard for representation of their textual form.

1.3. Data Availability

Except for a few identification fields many other fields are either optional or simply not applicable for a given instrument type. Inapplicable fields shall be ignored if present and they shall be left empty when exported. Also in many cases value of certain fields is not known by the data source. Such fields are considered to have an empty value and they are usually represented by a text of length 0. Empty or missing fields shall be interpreted as undefined or unknown. Import utility is allowed to process empty fields by keeping previous values or by using some default values. In such cases value “-” (or number “-1”) is used as instruction to erase previous field value.

1.4. Data Formats

- Text – value is a textual information, such as company name or exchange code
- Formatted text – complex fields use proprietary formats, such as list of exchanges
- Number – value is a number, such as contract size or strike price; value can be either integer or floating; floating values must use dot character as a decimal point; numeric value 0 is often considered an empty value
- Date – value is a date, such as last trading day or expiration date; value must be formatted using YYYY-MM-DD pattern; date 1970-01-01 is often considered an empty value

1.5. Format Extensibility

The format can be extended by addition of new fields and new literals (enumerated values) for existing fields. All parties working with format shall be prepared to deal with such extensions. Standard method is to ignore unknown fields as if they were not there. If some known and important field uses unknown literal then application can either ignore profile altogether, replace unknown literal with some ‘default’ one or show error to an operator and ask him to resolve situation.

1.6. Symbology

This format assumes that instrument profiles use symbology as defined on corresponding exchanges and national markets. Extensions and augmentations of "native" symbology are formally beyond the scope of this format. However, there are several common notations that are currently in use (see Section 1):

- Currency symbols replace 3rd letter with dollar sign in order to avoid name conflicts with stocks and indexes. This rule applies to symbol and underlying fields only, it does not apply to currency fields. Example: "USD", "EUR", "JPY" become "US$", "EU$", "JP$".
- Future symbols use prefix "/" for convenience. Example: "/YGM9".
- Option symbols use prefix "." for convenience. Example: ".ZYEAD".

1.7. List of Fields

In order to simplify interoperability the syntax of field names is restricted: names may use only capital Latin letters, decimal digits and underscore character, and they shall start with Latin letter.

- TYPE, text – type of instrument; takes precedence in conflict cases with other fields; mandatory field; may not be empty; example: "STOCK", "FUTURE", "OPTION"
- SYMBOL, text – identifier of instrument; preferable an *international* one in Latin alphabet; mandatory field; may not be empty; example: "GOOG", "/YGM9", ".ZYEAD"
- DESCRIPTION, text – description of instrument; preferable an *international* one in Latin alphabet; example: "Google Inc.", "Mini Gold Futures,Jun-2009,ETH"
- LOCAL_SYMBOL, text – identifier of instrument in *national* language; shall be empty if same as SYMBOL field
- LOCAL_DESCRIPTION, text – description of instrument in *national* language; shall be empty if same as DESCRIPTION field
- COUNTRY, text – country of origin (incorporation) of corresponding company or parent entity; shall use two-letter country code from ISO 3166-1 standard; see ISO 3166-1 on Wikipedia; example: "US", "RU"
- OPOL, text – Official Place Of Listing, the organization that have listed this instrument; instruments with multiple listings shall use separate profiles for each listing; shall use Market Identifier Code (MIC) from ISO 10383 standard (see ISO 10383 on Wikipedia or MIC homepage) or custom dxFeed values (see Custom OPOL values chapter); example: "XNAS", "RTSX"
- EXCHANGE_DATA, text – exchange-specific data required to properly identify instrument when communicating with exchange; uses exchange-specific format
- EXCHANGES, formatted text – list of exchanges where instrument is quoted or traded; shall use following format:
  <VALUE> ::= <empty> | <LIST>
  <LIST> ::= <MIC> | <MIC> <semicolon> <LIST>
  the list shall be sorted by MIC; example: "ARCX;CBSX ;XNAS;XNYS"
- CURRENCY, text – currency of quotation, pricing and trading; shall use three-letter currency code from ISO 4217 standard; see ISO 4217 on Wikipedia; example: "USD", "RUB"
- BASE_CURRENCY, text – base currency of currency pair (FOREX instruments); shall use three-letter currency code similarly to CURRENCY field
- CFI, text – Classification of Financial Instruments code; mandatory field for OPTION instruments as it is the only way to distinguish Call/Put type, American/European exercise, Cash/Physical delivery; shall use six-letter CFI code from ISO 10962 standard; allowed to use 'X' extensively and to omit trailing letters (assumed to be 'X'); see ISO 10962 on Wikipedia; example: "ESNTPB", "ESXXXX", "ES" , "OPASPS"

- **ISIN**, text – International Securities Identifying Number; shall use twelve-letter code from ISO 6166 standard; see ISO 6166 on Wikipedia or ISIN on Wikipedia; example: “DE0007100000”, “US38259P5089”
- **SEDOL**, text – Stock Exchange Daily Official List; shall use seven-letter code assigned by London Stock Exchange; see SEDOL on Wikipedia or SEDOL on LSE; example: “2310967”, “5766857”
- **CUSIP**, text – Committee on Uniform Security Identification Procedures code; shall use nine-letter code assigned by CUSIP Services Bureau; see CUSIP on Wikipedia; example: “38259P508”
- **ICB**, number – Industry Classification Benchmark; shall use four-digit number from ICB catalog; see ICB on Wikipedia or ICB homepage; example: “9535”
- **SIC**, number – Standard Industrial Classification; shall use four-digit number from SIC catalog; see SIC on Wikipedia or SIC structure; example: “7371”
- **MULTIPLIER**, number – market value multiplier; example: “100”, “33.2”
- **PRODUCT**, text – product for futures and options on futures (underlying asset name); example: “/YG”
- **UNDERLYING**, text – primary underlying symbol for options; example: “C”, “/YGM9”
- **SPC**, number – shares per contract for options; example: “1”, “100”
- **ADDITIONAL_UNDERLYINGS**, formatted text – additional underlyings for options, including additional cash; shall use following format:
  <VALUE> ::= <empty> | <LIST>
  <LIST> ::= <AU> | <AU> <semicolon> <space> <LIST>
  <AU> ::= <UNDERLYING> <space> <SPC>
  the list shall be sorted by UNDERLYING; example: “SE 50”, “FIS 53; US$ 45.46”
- **MMY**, text – maturity month-year as provided for corresponding FIX tag (200); can use several different formats depending on data source:
  YYYYMM – if only year and month are specified
  YYYYMMDD – if full date is specified
  YYYYMMwN – if week number (within a month) is specified
- **EXPIRATION**, date – date of expiration; example: “2009-01-17”
- **LAST_TRADE**, date – date of last trading day; example: “2009-01-16”
- **STRIKE**, number – strike price for options; example: “80”, “22.5”**OPTION_TYPE**, text – type of option; shall use one of following values:
  STAN = Standard Options
  LEAP = Long-term Equity AnticiPation Securities
  SDO = Special Dated Options
  BINY = Binary Options
  FLEX = FLexible EXchange Options
  VSO = Variable Start Options
  RNGE = Range
- **EXPIRATION_STYLE**, text – expiration cycle style such as “Weeklys”, “Quarterlys”
- **SETTLEMENT_STYLE**, text – settlement price determination style such as “Open”, “Close”
- **PRICE_INCREMENTS**, formatted text – minimum allowed price increments with corresponding price ranges; shall use following format:
  <VALUE> ::= <empty> | <LIST>
  <LIST> ::= <INCREMENT> | <RANGE> <semicolon> <space> <LIST>
  <RANGE> ::= <INCREMENT> <space> <UPPER_LIMIT>
  the list shall be sorted by UPPER_LIMIT; example: “0.25”, “0.01 3; 0.05”
- **TRADING_HOURS**, formatted text – trading schedule; example: “NewYorkUS(rt=0300;0=p04000930r09301600a16002000)”
- **FIRST_INTEREST_DATE**, date - for fixed coupon rate bonds: the periodic interest payment that the issuer makes during the life of the bond; example: “20090117”.
- **INTEREST_RATE,** number - the periodic interest payment that the issuer makes during the life of the bond, denominated in %; example: “2.68”

- ISSUE_DATE, date - the date on which an instrument is issued and begins to accrue interest; example: “20090127”.

1.8. List of Types

- BOND – debt instruments, excluding money market funds; see Bond on Wikipedia
- CURRENCY – national currency from ISO 4217 standard; see currency symbology
- ETF – exchange-traded fund; see ETF on Wikipedia
- FOREX – foreign exchange market; see Forex on Wikipedia
- FUTURE – futures contract, derivative instrument; see Futures on Wikipedia
- INDEX – non-tradable market performance indicators
- MONEY_MARKET_FUND – funds that invest in short-term debt instruments; see Money market fund on Wikipedia
- MUTUAL_FUND – investment funds, excluding ETFs and money market funds; see Mutual fund on Wikipedia
- OPTION – option contract, derivative instrument; see Option on Wikipedia
- OTHER – non-tradable miscellaneous instruments
- PRODUCT – grouping instrument for futures, aka futures product
- STOCK – tradable equities, excluding ETFs and mutual funds; see Stock on Wikipedia
- SPREAD – composite virtual instrument consisting of two or several individual instruments that represents multileg order

1.9. Field Applicability

	CURRENCY	FOREX	BOND	INDEX	STOCK	ETF	MUTUAL_FUND	MONEY_MARKET_FUND	PRODUCT	FUTURE	OPTION	SPREAD	OTHER
TYPE	M	M	M	M	M	M	M	M	M	M	M	M	M
SYMBOL	M	M	M	M	M	M	M	M	M	M	M	M	M
DESCRIPTION	+	+	+	+	+	+	+	+	+	+	+	+	+
LOCAL_SYMBOL	+	+	+	+	+	+	+	+	+	+	+	+	+
LOCAL_DESCRIPTION	+	+	+	+	+	+	+	+	+	+	+	+	+
COUNTRY	M	-	+	+	+	+	+	+	+	+	+	+	+
OPOL	+	+	+	+	+	+	+	+	+	+	+	+	+
EXCHANGE_DATA	+	+	+	+	+	+	+	+	+	+	+	+	+
EXCHANGES	+	+	+	+	+	+	+	+	+	+	+	+	+
CURRENCY	M	M	M	M	M	M	M	M	M	M	M	M	M
BASE_CURRENCY	-	M	-	-	-	-	-	-	-	-	-	-	-
CFI	+	+	+	+	+	+	+	+	+	+	M	+	+
ISIN*		-	+	-	+	+	+	+	-	-	-	-	+
SEDOL*	-	-	+	-	+	+	+	+	-	-	-	-	+
CUSIP*	-	-	+	-	+	+	+	+	-	-	-	-	+
ICB	-	-	+	+	+	+	+	+	+	+	+	-	+
SIC	-	-	+	+	+	+	+	+	+	+	+	-	+
MULTIPLIER	-	-	-	-	-	-	-	-	-	M	M	M	+
PRODUCT	-	-	-	-	-	-	-	-	-	M	+	+	-
UNDERLYING	-	-	-	-	-	-	-	-	-	-	M	-	-
SPC	-	-	-	-	-	-	-	-	-	-	M	-	-
ADDITIONAL_UNDERLYINGS	-	-	-	-	-	-	-	-	-	-	+	-	-
MMY	-	-	+	-	-	-	-	-	-	+	+	+	+
EXPIRATION	-	-	+	-	-	-	-	-	-	M	M	+	+
LAST_TRADE	-	-	+	-	-	-	-	-	-	M	M	+	+
STRIKE	-	-	-	-	-	-	-	-	-	-	M	-	-
OPTION_TYPE	-	-	-	-	-	-	-	-	-	-	+	-	-
EXPIRATION_STYLE	-	-	-	-	-	-	-	-	-	-	+	-	-
SETTLEMENT_STYLE	-	-	-	-	-	-	-	-	-	-	+	-	-
PRICE_INCREMENTS	+	+	+	+	+	+	+	+	+	+	+	+	+
TRADING_HOURS	M	M	M	M	M	M	M	M	M	M	M	M	M
FIRST_INTEREST_DATE	-	-	+	-	-	-	-	-	-	-	-	-	-
INTEREST_RATE	-	-	+	-	-	-	-	-	-	-	-	-	-
ISSUE_DATE	-	-	M	-	-	-	-	-	-	-	+	-	-

"M" – Mandatory field; shall always be specified; records can be rejected if value is not specified.
"+" – Optional field; shall be specified if value is known.
"-" – Inapplicable field; shall never be specified; value shall be ignored if specified.

*For some markets ISIN, CUSIP and SEDOL fields are available only with the corresponding license

1.10. Custom OPOL values

Code	Name
AFX	Alternative FX
DBI	Deutsche Boerse indices
DJI	Dow Jones indices
dxFeed	dxFeed-generated symbols or indicators
FTSE	FTSE GDS Indices
FRI	FTSE (Financial Times Stock Exchange Russell indices)
INAV	Intraday Net Asset Value & Indicative Optimized Portfolio Value (INAV)
MSCII	MSCI indices
MSFX	Morningstar FX
RTI	Russell Tick indices
SPI	S&P Indices
TFX	TrueFX
TEI	Trading Economics indices
BTRX	Bittrex
CCI	Cryptocurrency
CXBINA	Binance
CXBITF	Bitfinex
CXBITS	Bitstamp
CXBMEX	Bitmex
CXCCAP	Coincap
CXCEX	CEX.IO
CXDXF	dxFeed-generated crypto-indicators
CXETHF	Ethfinex
CXEXMO	Exmo
CXFLYR	Bitflyer
CXGMNI	Gemini
CXHITB	HitBTC
CXHUO	Huobi Global
CXOKFT	OkCoin
CXOKEX	OkEx
CXMATE	Coinmate
CXPLNX	Poloniex
GDAX	Coinbase PRO
KBE	Kraken

1.11. Simple File Format

Instrument profiles can be written to a file in a simple format, which is based on a CSV (Comma-Separated Values) format as defined in RFC 4180 – see CSV on Wikipedia or RFC 4180 homepage. The basic CSV format is extended to support several alternating record formats which formally violates CSV restrictions. To avoid possible confusion files in this format shall use extension IPF. The similarities and differences are:

- Each record is located on a separate line. Each line ends with a line break (CRLF). Last line with a record also ends with line break. Empty lines (parsing artifacts) shall be ignored.
- Records contain fields, separated by commas. Last field must not be followed by a comma.
- Header line is not used. Instead records can be of 2 variants: metadata and profile.
- Metadata records define list of fields used by profile records for given instrument type.
- Fields use standard CSV quotation rules by enclosing in double-quotes.
- UTF-8 encoding is used throughout entire file.

Metadata record can be distinguished by special format of its first field: it starts with hash character and ends with "::=TYPE" text, and the text in the middle specifies instrument type. The rest of metadata record defines list of fields (in addition to TYPE) that are used by subsequent profile records for specified instrument type. Thus metadata record looks like BNF definition of profile record. Note that field "TYPE" must be the first field for each profile record and it determines what format is used by this record.
If some line starts with hash character but does not form correct metadata record (i.e. first field does not end with "::=TYPE" text), then this line is considered a comment line and shall be ignored. Note that comment line must follow CSV format in regard to correct field separation and quotation. In this regard commenting valid but unused CSV lines works perfectly, but plain comments shall avoid double quote characters or they can easily violate CSV specification.
These extensions allow single IPF file to contain profiles for different instrument types with different record formats. They also allow concatenation of several files with different record formats for same instrument types into single one because each new metadata record redefines record format.

1.12. Sample File

```
#STOCK::=TYPE,SYMBOL,DESCRIPTION,CURRENCY
STOCK,GOOG,Google Inc.,USD
#FUTURE::=TYPE,SYMBOL,DESCRIPTION,CURRENCY,MULTIPLIER,PRODUCT,LAST_TRADE
FUTURE,/YGM9,"Mini Gold Futures,Jun-2009,ETH",USD,33.2,/YG,2009-06-26
```

2. Transform Language

The source of instrument profiles often lacks certain important data or interprets some data differently. As a result instrument profiles shall be modified to make data complete and consistent. This operation is called data transformation and is performed using instructions written in a special “transform” programming language. The complete set of instructions is executed for each instrument profile independently and modified profiles are then distributed to recipients.
Syntax of transform language is based on Java programming language, although set of supported features is much smaller. The main syntax difference is “switch” statement – there is no such thing as a “break” statements and no “fall through” from top to bottom – instead each label must have exactly one execution statement (although it can be a block statement, etc.).

Here is a list of features and constructs supported by transform language:

```
// Basic expression syntax from lowest precedence to highest precedence:
A ? B : C                          // Conditional expressions
(A || !B) && C || D                // Logical expressions
A == B || C != D                   // Equality expressions
A < B || C >= D                    // Relational expressions
A like "pattern"                   // "like" expressions (relational)
A in (B, C, D)                     // "in" expressions (relational)
A + (B - C)                        // Additive expressions
A * (B / D) % E                    // Multiplicative expressions
!A, +B, -C                         // Unary expressions
replaceAll(A, "pattern", B)        // Replacement expressions (functions)
replaceFirst(A, "pattern", B)

// Terminal literals:
SYMBOL, CUSIP, SPC, STRIKE, …      // Field references
"It's a \"text\"\n", 'hi'          // Textual constants
0, 123, 3.45, .5, 2.0              // Numeric constants
true, false                        // Boolean constants

// Statement constructs:
A; B; C;                           // Simple statements
{A; B; C;}                         // Block statements
if (A) B; if (A) B; else C;        // "if-then-else" statements
switch (A) {                       // "switch" statements
case B:  C;
case D:  {E; F;}
default: if (G) H; else I;
}
A = B; A += B; A -= B;             // Assignment statements
A *= B; A /= B; A %= B;

// Procedures (terminal statements):
delete();                          // Deletes current instrument profile.
primary("IBM");                    // Selects new primary UNDERLYING "IBM"
                                   // from additional underlyings.
rename("MSFT", "GOOG");            // Renames UNDERLYING (primary and additional)
                                   // from "MSFT" to "GOOG".
cmeproduct("Wheat Futures,,ETH", 50, 1, 1, 100);
                                   // Adjusts CME products for missing data.

Here is a sample transform program:
// Delete FLEX options - they are not currently supported.
if (OPTION_TYPE == "FLEX") delete();

// Rename underlyings with subclasses to proper symbology.
rename("BBIB", "BBI/B");
rename("CBSA", "CBS/A");
rename("TRXB", "TRX/B");
```

```
rename("VIAB", "VIA/B");

// Move European OEX options to XEO underlying.
if (UNDERLYING == "OEX" && CFI like "O.E...") UNDERLYING = "XEO";

// Move FRO options to proper underlyings (AMEX FRO Settlement Indexes).
if (OPTION_TYPE == "BINY")
switch (UNDERLYING) {
case "AAPL": UNDERLYING = "AXO";
case "C":    UNDERLYING = "EFH";
case "IBM":  UNDERLYING = "TSB";
case "SPY":  UNDERLYING = "SQY";
}

// Mark options with "Weeklys"/"Quarterlys" flag.
if (OPTION_TYPE == "SDO" && UNDERLYING == "SPX" && SYMBOL like "\.JX[ABDE]..")
    EXPIRATION_STYLE = "Weeklys";
if (OPTION_TYPE == "SDO" && UNDERLYING == "XSP" && SYMBOL like "\.JG[ABDE]..")
    EXPIRATION_STYLE = "Weeklys";
if (OPTION_TYPE == "SDO" && UNDERLYING == "OEX" && SYMBOL like "\.RZ[ABDE]..")
    EXPIRATION_STYLE = "Weeklys";
if (OPTION_TYPE == "SDO" && UNDERLYING == "XEO" && SYMBOL like "\.KF[ABDE]..")
    EXPIRATION_STYLE = "Weeklys";
if (OPTION_TYPE == "SDO" && EXPIRATION_STYLE == "")
    EXPIRATION_STYLE = "Quarterlys";

// Adjust last trading day to reflect reality.
if (TYPE == "OPTION" && OPTION_TYPE != "SDO" && LAST_TRADE != 0)
    LAST_TRADE -= 1;
if (TYPE == "OPTION" && EXPIRATION_STYLE != "Quarterlys" && LAST_TRADE != 0 &&
  UNDERLYING in ("DJX", "DXL", "GOX", "HGX", "MID", "MNX", "MSH", "NDX",
  "OIX", "OSX", "RMN", "RUI", "RUT", "SML", "SOX", "SPX", "XSP"))
    LAST_TRADE -= 1;
```