# FiMaLib – Financial Mathematics Library

An open source financial mathematics library and market data database

# 0) Versioning

This document is intended as a working guide for the FiMaLib project. It is a high level description of the various parts of the project. Detailed development notes may be supplied in separate documents

## 0.1) Version history

| Date | Version | Author(s) | Description |
|---|---|---|---|
| 03-Mar-2017 | 0.1 | Peter Werno | Initial version with rudimentary layout and description of the various parts |

## 0.2) Table of contents

## Inhaltsverzeichnis


**0) Versioning** **2**
*0.1) Version history* *2*
*0.2) Table of contents* *2*

**1) General layout** **2**
*1.1) Structural diagram of the FiMaLib system* *3*
*1.2) Programming principles* *3*
1.2.1) Java coding conventions 3
1.2.2) SQL naming conventions 3

**2) Instrument Statics** **5**
*2.1) The instrument statics database* *5*
2.1.1) Keeping track of changes in the static data 5
2.1.2) Major tables in the instrument statics database 5
2.1.3) Daycount convention tables in the FiMaLib DB 8
2.1.4) User management tables in the FiMaLib DB 9
*2.2) The instrument statics webservice* *9*

**3) Instrument market data** **9**
*3.1) The instrument market data database* *9*
*3.2) The instrument market data webservice* *9*


# 1) General layout

The FiMaLib system consists of a number of parts, most of which can be used independently to solve special problems.

## 1.1) Structural diagram of the FiMaLib system

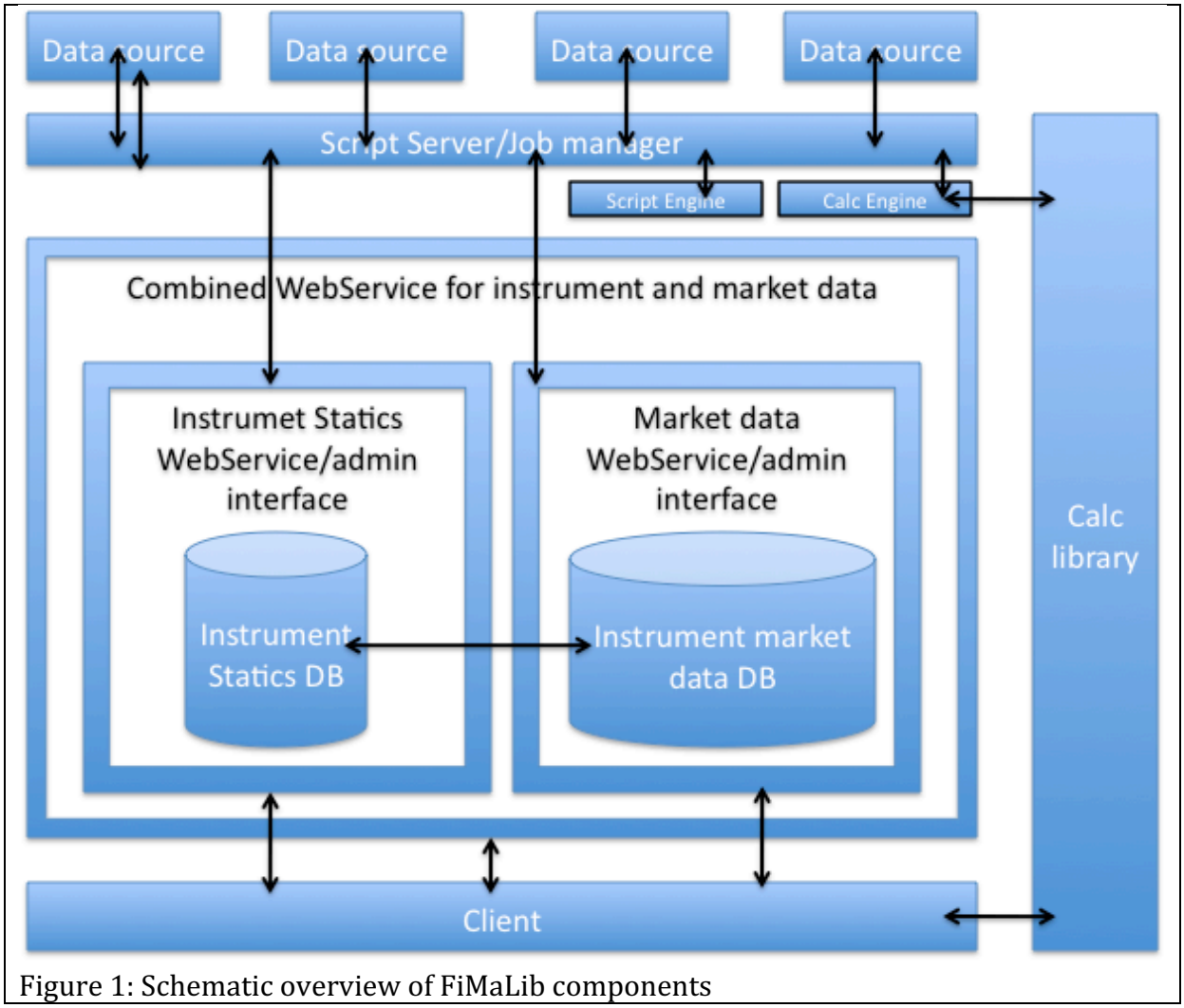

Figure 1: Schematic overview of FiMaLib components

## 1.2) Programming principles

The majority of the FiMaLib system will be coded in Java, using SQL (and potentially other) database systems for the data storage.

### 1.2.1) Java coding conventions

tbd

### 1.2.2) SQL naming conventions

Tables in the sql databases of FiMaLib will be prepended with the code “tbl” in lower case, followed by a descriptive information of what data is stored in this table using the same naming convention used for Java classes and using the plural form (e.g. “tblInstruments” or “tblInstrumentTypes”).

Fields in FiMaLib SQL tables are preceded with the code “fld” in lower case, followed by a descriptive information of what data is stored in this field using the same naming convention used for Java classes and using the singular form (e.g. “fldID”, “fldName”, “fldInstrumentType”).

Exception: If the field contains foreign keys (i.e. IDs in a different table), then the field name is preceded by "fk" instead of "fld". If a field is (or is part of) a primary key, the name is preceded by "pk" instead of "fld".

## 1.3 Access rights

While the FiMaLib project intends to provide an open source database where all data should be public data, the project may be used by institutions where selective access to data is necessary. Therefore, FiMaLib includes a user account management as well as an access right strategy.

Within the open source database, user accounts are required for **manipulating** data, while all data stored in the public database is readable by any user, including a "guest" user.

### 1.3.1 User accounts

A very simplistic user table is defined in the statics database which takes on basic information on the user. A user in this table may be linked to a separate access rights system such as a NIS, NetInfo, LDAP or other system. In this case, only the FiMaLib-internal user ID and the external ID reference will be used. Otherwise, FiMaLib also comes with its own user access system. In that case, the whole tblUsers table will be used.

#### *1.3.1.1 The user table tblUsers*

The user table "tblUsers" comes with the following fields, allowing for a fully-fledged user management for the FiMaLib database system.

| Field name | Field type | Description |
|---|---|---|
| fldUserID | Integer/numeric | An auto-number integer which is used as a foreign key in all other tables that store a user info (e.g. the historization tables in chapter 2.1.1) |
| fldExternalID | Variant/Text | A reference to an external user account management system (depending on the configuration) |
| fldUserName | Varchar(1024) | The user/login name |
| fldPassword | Varchar(1024) | The user's password encrypted |
| fldFirstName | Varchar(128) | The user's first name(s) |
| fldLastName | Varchar(128) | The user's last name |
| fldAddress | Varchar(1024) | The user's address (street, zip, city, Country, etc.) |
| fldEMail | Varchar(1024) | The user's email address |
| fldCreationDate | Datetime | When the user entry was created |
| fldConfirmed | Boolean | Whether the user email was confirmed |
| fldDeleted | Boolean | Logically deletion flag |
| fldDeletionDate | Datetime | When the user was deleted |

### 1.3.2 Access rights

# 2) Instrument Statics

The instrument statics part contains information about financial instruments that have a descriptive character – i.e. they describe the financial instrument. This data is usually considered "static", which means this data usually does not change during the lifetime of the instrument (in contrast to dynamic or market data which is dependent on the point in time that they are observed). Unfortunately, static data occasionally changes (e.g. a bond coupon may change, a currency may see a devaluation and subsequently a split, a company may get purchased, renamed, closed, etc.). Therefore, also static data needs to be historised properly to reflect these changes.

## 2.1) The instrument statics database

The instrument statics database is the database containing the data describing the details of a financial instrument. As instrument data is often inter-linked, a relational database appears the best fit. However, instrument static data also brings certain features usually implemented in the form of objects (e.g. with inheritance). To cater for this feature, the "initial" tables are highly normalized.

### 2.1.1) Keeping track of changes in the static data

Most tables in the statics database have a corresponding shadow-table (preceded with an additional "hist" preceding the table name, e.g. "histtblInstruments"). These tables have three additional fields, which show what kind of change was last performed. These three fields will be part of the primary key of the "hist" table – on top of the fields making up the primary key of the underlying table.

| Field name | Field type | Description |
|---|---|---|
| fkHistChangedBy | Integer/numeric | A link to the table tblUsers, indicating which person changed this data |
| pkHistChangeDate | Datetime | The date/time when the change was done |
| pkHistChangeType | Integer/numeric | The type of change:<br>1 = insertion<br>2 = update<br>3 = deletion |

### 2.1.2) Major tables in the instrument statics database

This part describes the tables used for storing instrument static data in the FiMaLib DB:

#### *2.1.2.1) Tables related to the instrument type (tblInstrumentTypes, tblInstrumentTypeFields, tblVersions)*

These tables contain information about instrument types. This information must be synchronized with the instrument types in the public FiMaLib database and therefore is

provided as defaults with the FiMaLib code (i.e. it must be ensured that a Java object that stores a "CDS rate" instrument has the same instrument type id and fields for "CDS rate" in the public FiMaLib DB as well as in any user-operated version of the FiMaLib DB). Furthermore, the information in this table should not be changed except for when a new version of FiMaLib is installed. In this case, these tables will be (re)created and filled with all necessary information. Obviously, these tables do NOT have shadow-tables as described in 2.1.1)

Attention: FiMaLib (Java) code should be downward-compatible! As such, the Java code for a higher version of FiMaLib should be able to cope with a user-installed FiMaLib database with a lower version!

#### 2.1.2.1.1) tblVersions

This table contains only one entry – the version of the FiMaLib static DB installed. It must not be changed except when installing a different FiMaLib static DB version! The version number uses a triplet of major and minor version number and a release number, such as "1.2.7", split into three fields as follows:

| Field Name | Field type | Description |
|---|---|---|
| fldMajorVersion | Integer/numeric | The major version ID |
| fldMinorVersion | Integer/numeric | The minor version ID |
| fldRelease | Integer/numeric | The release |

#### 2.1.2.1.2) tblInstrumentTypes

Primary key: fldInstrumentTypeID

| Field name | Field type | Description |
|---|---|---|
| fldInstrumentTypeID | Integer/numeric | The FiMaLib ID of the instrument type |
| fkParentInstrumentTypeID | Integer/numeric | The parent type (e.g. a floating rate note is a child instrument type of "Bond") |
| fldInstrumentTypeName | Varchar (256) | A short description of the instrument type. Attention: This may be prone to localization problems |
| fldFiMaLibClass | Varchar(1024) | The class name of the FiMaLib statics library that implements the class. |

This table is prefilled with instrument types based on the version of the FiMaLib library (here, the last field also relates to the implementation of the FiMaLib class, which must be an implementation of the interface defined in org.fimalib.instruments). A definition of all implemented instrument types can be found in the definitions of the interface org.fimalib.instruments.Instrument. As per Version 1.0.1, the content of this table is:

| ID | Parent | Name | Class* | Since Vers. |
|---|---|---|---|---|
| 1 | Null | Generic Instrument | InstrumentImpl | 1.0.1 |
| 2 | Null | Legal Entity | LegalEntityImpl | 1.0.1 |
| 3 | Null | Stock | StockImpl | 1.0.1 |

| ID | Parent | Name | Class* | Since Vers. |
|---|---|---|---|---|
| 4 | Null | Commodity | CommodityImpl | 1.0.1 |
| 5 | Null | Loan | LoanImpl | 1.0.1 |
| 6 | Null | Bond | BondImpl | 1.0.1 |
| 7 | Null | Rate | RateImpl | 1.0.1 |
| 8 | Null | Curve | CurveImpl | 1.0.1 |
| 9 | Null | FX | CurrencyImpl | 1.0.1 |
| 10 | Null | Derivative | DerivativeImpl | 1.0.1 |
| 11 | Null | Portfolio | PortfolioImpl | 1.0.1 |
| 12 | Null | Country | CountryImpl | 1.0.1 |
| 3001 | 3 | Common Stock | CommonStockImpl | 1.0.1 |
| 3002 | 3 | Preferred Stock | PreferredStockImpl | 1.0.1 |
| 4001 | 4 | Energy Commodity | EnergyCommodityImpl | 1.0.1 |
| 4002 | 4 | Metal Commodity | MetalCommodityImpl | 1.0.1 |
| 4003 | 4 | Agricultural Commodity | AgriculturalCommodityImpl | 1.0.1 |
| 4004 | 4 | Lifestock Commodity | LifestockCommodityImpl | 1.0.1 |
| 5001 | 5 | Final Payment Loan | FinalPaymentLoanImpl | 1.0.1 |
| 5002 | 5 | Defined Payments Loan | DefinedPaymentsLoanImpl | 1.0.1 |
| 5003 | 5 | Amortizing Loan | AmortizingLoanImpl | 1.0.1 |
| 6001 | 6 | Bullet Bond | BulletBondImpl | 1.0.1 |
| 6002 | 6 | Floating Rate Note | FRNImpl | 1.0.1 |
| 6003 | 6 | Convertible Bond | ConvertibleBondImpl | 1.0.1 |
| 7001 | 7 | Par Rate | ParRateImpl | 1.0.1 |
| 7002 | 7 | Zero Rate | ZeroRateImpl | 1.0.1 |
| 7003 | 7 | Swap Rate | SwapRateImpl | 1.0.1 |
| 7004 | 7 | Forward Rate | ForwardRateImpl | 1.0.1 |
| 7005 | 7 | Exchange Rate | ExchangeRateImpl | 1.0.1 |
| 8001 | 8 | Yield Curve | YieldCurveImpl | 1.0.1 |
| 8002 | 8 | Par Curve | ParCurveImpl | 1.0.1 |
| 8003 | 8 | Zero Curve | ZeroCurveImpl | 1.0.1 |
| 8004 | 8 | Swap Curve | SwapCurveImpl | 1.0.1 |
| 10001 | 10 | Forward | ForwardImpl | 1.0.1 |
| 10002 | 10 | Future | FutureImpl | 1.0.1 |
| 10003 | 10 | Option | OptionImpl | 1.0.1 |
| 10004 | 10 | Swaption | SwaptionImpl | 1.0.1 |

* Class name in the package org.fimalib.statics.instruments

#### 2.1.2.1.3) tblFields

primary key: fldFieldID

| Field name | Field type | Description |
|---|---|---|
| fldFieldID | Integer/numeric | |
| fldFieldName | Varchar (256) | Description of what data this field contains (e.g. “Coupon”) |
| fldFieldType | Enumeration | 1 = Integer |

| | | |
|---|---|---|
| | | 2 = Floating point number<br>3 = String (size?)<br>4 = Datetime<br>5 = Boolean |

#### 2.1.2.1.4) tblInstrumentTypeFields

primary key: fkInstrumentTypeID, fkFieldID

| Field name | Field type | Description |
|---|---|---|
| fkInstrumentTypeID | Integer/numeric | A link to the instrument type table (tblInstrumentTypes – see 2.1.2.1.2) |
| fkFieldID | Integer/numeric | A link to the fields table (tblFields – see 2.1.2.1.3) |

### *2.1.2.2) The instruments table (tblInstruments)*

The most important part of tblInstruments is the field fldID, containing the FiMaLib ID. This ID is used to identify the instrument in any other table containing instrument data. The ID uses auto-increment to ensure a unique key for this table. This can however cause problems if a user wants to run his/her own database (e.g. a bank that wants to store its own in-house prices in the FiMaLib DB without showing them to the rest of the world as would be the case when using the public FiMaLib DB) as internal IDs may (most certainly "will") be different from the FiMaLib IDs used in the public database. To solve this problem, a matching can be done using an external ID.

| Field name | Field type | Description |
|---|---|---|
| fldID | Long Integer/numeric | The FiMaLib ID of the instrument |
| fkInstrumentTypeID | Integer/numeric | A link to the table tblInstrumentType |
| fldDescription | Varchar | A description of the instrument |
| fkCreatedBy | Integer/numeric | A link to the table tblUsers, indicating the person that initiated the creation of the instrument |
| fldCreationDate | Datetime | The date/time when the instrument was created in the FiMaLib DB |
| fkLastChangedBy | Integer/numeric | A link the the table tblUsers, indication the person that last changed a static detail of the instrument |
| fldLastChangedDate | Datetime | The date/time when the instrument static data was last changed. |

## 2.1.3) Daycount convention tables in the FiMaLib DB

#### 2.1.4) User management tables in the FiMaLib DB

FiMaLib should be able to use different types of user access/rights management. In some cases, the actual management will happen in external systems (e.g. using personal digital certificates, using a Windows domain account, etc.). Nonetheless, many tables require a user ID (see 2.1.1 for example). The system must therefore ensure that the user tables are filled with data from external systems when a uses accesses the service.

2.1.4.1) the user table (tblUsers)
This table stores the users' data

### 2.2) The instrument statics webservice

## 3) Instrument market data

### 3.1) The instrument market data database

### 3.2) The instrument market data webservice