ORE User Guide

Quaternion Risk Management

28 May 2019

Document History

Date	Author	Comment
7 October 2016	Quaternion	initial release
28 April 2017	Quaternion	updates for release 2
7 December 2017	Quaternion	updates for release 3
28 May 2019	Quaternion	updates for release 4

Contents

1 Introduction

The *Open Source Risk Project* [1] aims at providing a transparent platform for pricing and risk analysis that serves as

- a benchmarking, validation, training, and teaching reference,
- an extensible foundation for tailored risk solutions.

Its main software project is *Open Source Risk Engine* (ORE), an application that provides

- a Monte Carlo simulation framework for contemporary risk analytics and value adjustments
- simple interfaces for trade data, market data and system configuration
- simple launchers and result visualisation in Jupyter, Excel, LibreOffice
- unit tests and various examples.

ORE is open source software, provided under the Modified BSD License. It is based on QuantLib, the open source library for quantitative finance [2].

Audience

The project aims at reaching quantitative risk management practitioners (be it in financial institutions, audit firms, consulting companies or regulatory bodies) who are looking for accessible software solutions, and quant developers in charge of the implementation of pricing and risk methods similar to those in ORE. Moreover, the project aims at reaching academics and students who would like to teach or learn quantitative risk management using a freely available, contemporary risk application.

Contributions

Quaternion Risk Management [3] is committed to sponsoring the Open Source Risk project through ongoing project administration, through providing an initial release and a series of subsequent releases in order to achieve a wide analytics, product and risk factor class coverage. The community is invited to contribute to ORE, for example through feedback, discussions and suggested enhancement in the forum on the ORE site [1], as well as contributions of ORE enhancements in the form of source code. See the FAQ section on the ORE site [1] on how to get involved.

Scope and Roadmap

ORE currently provides portfolio pricing, cash flow generation, sensitivity analysis, stress testing and a range of contemporary derivative portfolio analytics. The latter are based on a Monte Carlo simulation framework which yields the evolution of various **credit exposure** measures:

- EE aka EPE (Expected Exposure or Expected Positive Exposure)
- ENE (Expected Negative Exposure, i.e. the counterparty's perspective)

- 'Basel' exposure measures relevant for regulatory capital charges under internal model methods
- PFE (Potential Future Exposure at some user defined quantile)

and **derivative value adjustments**

- CVA (Credit Value Adjustment)
- DVA (Debit Value Adjustment)
- FVA (Funding Value Adjustment)
- COLVA (Collateral Value Adjustment)
- MVA (Margin Value Adjustment)

for portfolios with netting, variation and initial margin agreements.

The sensitivity framework yields further **market risk measures** such as ORE's parametric Value at Risk which takes deltas, vegas, gammas and cross gammas into account. This may be used to benchmark initial margin models such ISDA'S Standard Initial Margin Model.

Subsequent ORE releases will also compute **regulatory capital charges** for counterparty credit risk under the new standardised approach (SA-CCR), and the Monte Carlo based market risk measures will be complemented by parametric methods, e.g. for benchmarking various initial margin calculation models applied in cleared and non-cleared derivatives business.

The product coverage of the fourth release of ORE in March 2018 is sketched in the following table.

Product	Pricing and Cashflows	Sensitivity Analysis	Stress Testing	Exposure Simulation & XVA
Fixed and Floating Rate Bonds/Loans	Y	Y	Y	N
Interest Rate Swaps	Y	Y	Y	Y
Caps/Floors	Y	Y	Y	Y
Swaptions	Y	Y	Y	Y
Constant Maturity Swaps, CMS Caps/Floors	Y	Y	Y	Y
FX Forwards	Y	Y	Y	Y
Cross Currency Swaps	Y	Y	Y	Y
FX Options	Y	Y	Y	Y
Equity Forwards	Y	Y	Y	Y
Equity Swaps	Y	Y	Y	N
Equity Options	Y	Y	Y	Y
Commodity Forwards	Y	Y	N	N
Commodity Options	Y	Y	N	N
CPI Swaps	Y	Y	N	Y
Year-on-Year Inflation Swaps	Y	Y	N	Y
Credit Default Swaps	Y	Y	N	N

Table 1: ORE product coverage.

Future releases will further extend the product range and analytics coverage indicated in the table above, expand on the market risk analytics, add integrated credit/market risk analytics.

The simulation models applied in ORE's risk factor evolution implement the models discussed in detail in *Modern Derivatives Pricing and Credit Exposure Analysis* [20]: The IR/FX/INF/EQ risk factor evolution is based on a cross currency model consisting of an arbitrage free combination of Linear Gauss Markov models for all interest rates and lognormal processes for FX rates and EQ prices, Dodgeson-Kainth models for inflation. The model components are calibrated to cross currency discounting and forward curves, Swaptions, FX Options, EQ Options and CPI caps/floors.

Further Resources

- Open Source Risk Project site: `http://www.opensourcerisk.org`
- Frequently Asked Questions: `http://www.opensourcerisk.org/faqs`
- Forum: `http://www.opensourcerisk.org/forum`
- Source code and releases: `https://github.com/opensourcerisk/engine`
- Language bindings: `https://github.com/opensourcerisk/ore-swig`
- Follow ORE on Twitter `@OpenSourceRisk` for updates on releases and events

Organisation of this document

This document focuses on instructions how to use ORE to cover basic workflows from individual deal analysis to portfolio processing. After an overview over the core ORE data flow in section 3 and installation instructions in section 4 we start in section 5 with a series of examples that illustrate how to launch ORE using its command line application, and we discuss typical results and reports. We then illustrate in section 6 interactive analysis of resulting 'NPV cube' data. The final sections of this text document ORE parametrisation and the structure of trade and market data input.

2 Release Notes

This section summarises the high level changes between release 3 (December 2017) and 4 (March 2019).

INSTRMENTS

- Commodity Forward and Option, see example 5.24
- Equity Swap, see extended example 5.16
- CMS Spread Option (Cap/Floor, Digital Cap/Floor), see example 5.25

MARKETS

- New calendars: Chile, Colombia, Malaysia, Peru, Philippines, Thailand
- New IBOR indexes: CHF SARON, CLP CAMARA, COP IBR, DEM LIBOR, DKK OIS, NOWA, PHP PHIREF, RUB MOSPRIME, SEK SIOR, THB BIBOR

- New inflation idexes and regions: DKCPI, SECPI
- Equity index added

TERM STRUCTURES

- Cap/Floor smile volatility surface added
- Cross currency basis swap helper (with MtM Reset) added
- Cross currency fixed vs. float swap helper added, see example 5.29
- Discount ratio curves added, see example 5.28
- Correlation term structure added (to support CMS spread products)

ANALYTICS

- KVA added (thanks to Roland Kapl)

UNIT TESTS

- Unit tests suites extended to 429 cases in total
- Data driven tests added in ORE Data
- Now using boost's automated test suite creation and registration

EXAMPLES

- ORE has 29 examples now vs 23 in the previous release

USER GUIDE

- Extended to 186 pages

BUILDING ORE

- CMake build system added, see end of section 4.2

LANGUAGE BINDINGS

- ORE SWIG projected added, to support ORE in Python, see `https://github.com/OpenSourceRisk/ORE-SWIG`

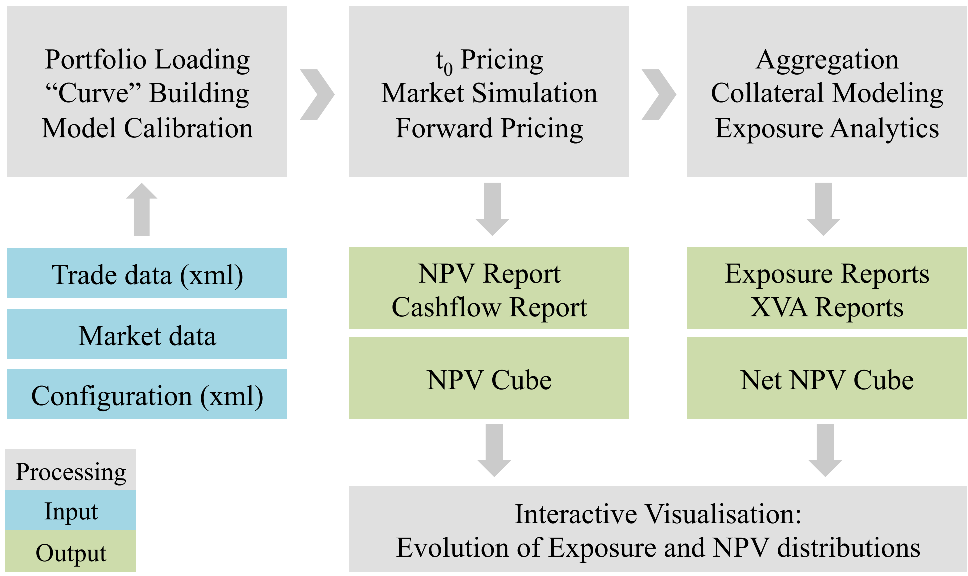

Figure 1: Sketch of the ORE process, inputs and outputs.

3 ORE Data Flow

The core processing steps followed in ORE to produce risk analytics results are sketched in Figure 1. All ORE calculations and output are generated in three fundamental process steps as indicated in the three boxes in the upper part of the figure. In each of these steps appropriate data (described below) is loaded and results are generated, either in form of a human readable report, or in an intermediate step as pure data files (e.g. NPV data, exposure data).
The overall ORE process needs to be parametrised using a set of configuration XML files which is the subject of section 7. The portfolio is provided in XML format which is explained in detail in sections 8 and 9. Note that ORE comes with 'Schema' files for all supported products so that any portfolio xml file can be validated before running through ORE. Market data is provided in a simple three-column text file with unique human-readable labelling of market data points, as explained in section 10.
The first processing step (upper left box) then comprises

- loading the portfolio to be analysed,
- building any yield curves or other 'term structures' needed for pricing,
- calibration of pricing and simulation models.

The second processing step (upper middle box) is then

- portfolio valuation, cash flow generation,
- going forward - conventional risk analysis such as sensitivity analysis and stress testing, standard-rule capital calculations such as SA-CCR, etc,
- and in particular, more time-consuming, the market simulation and portfolio valuation through time under Monte Carlo scenarios.

This process step produces several reports (NPV, cashflows etc) and in particular an **NPV cube**, i.e. NPVs per trade, scenario and future evaluation date. The cube is written to a file in both condensed binary and human-readable text format.
The third processing step (upper right box) performs more 'sophisticated' risk analysis by post-processing the NPV cube data:

- aggregating over trades per netting set,
- applying collateral rules to compute simulated variation margin as well as simulated (dynamic) initial margin posting,
- computing various XVAs including CVA, DVA, FVA, MVA for all netting sets, with and without taking collateral (variation and initial margin) into account, on demand with allocation to the trade level.

The output of this process step are XVA reports and the 'net' NPV cube, i.e. after aggregation, netting and collateral.
The example section 5 demonstrates for representative product types how the described processing steps can be combined in a simple batch process which produces the mentioned reports, output files and exposure evolution graphs in one 'go'.
Moreover, both NPV cubes can be further analysed interactively using a visualisation tool introduced in section 6.1. And finally, sections 6.2 and 6.3 demonstrate how ORE processes can be launched in spreadsheets and key results presented automatically within the same sheet.

4 Getting and Building ORE

You can get ORE in two ways, either by downloading a release bundle as described in section 4.1 (easiest if you just want to use ORE) or by checking out the source code from the github repository as described in section 4.2 (easiest if you want to build and develop ORE).

4.1 ORE Releases

ORE releases are regularly provided in the form of source code archives, Windows executables `ore.exe`, example cases and documentation. Release archives will be provided at `https://github.com/opensourcerisk/engine/releases`.

The release consists of a single archive in zip format

- `ORE-<VERSION>.zip`

When unpacked, it creates a directory `ORE-<VERSION>` with the following files respectively subdirectories

1. `App/`
2. `Docs/`
3. `Examples/`
4. `FrontEnd/`

5. `OREAnalytics/`
6. `OREData/`
7. `QuantExt/`
8. `ThirdPartyLibs/`
9. `tools/`
10. `xsd/`
11. `userguide.pdf`

The first three items and `userguide.pdf` are sufficient to run the compiled ORE application on the list of examples described in the user guide (this works on Windows only). The Windows executables are located in `App/bin/Win32/Release/` respectively `App/bin/x64/Release/`. To continue with the compiled executables:

- Ensure that the scripting language Python is installed on your computer, see also section 4.3 below;
- Move on to the examples in section 5.

The release bundle does contain the ORE source code, which is sufficient to build ORE from sources manually as follows (if you build ORE for development purposes, we recommend using git though, see section 4.2):

- Set up Boost as described in section 4.2.2, unless already installed
- Set up QuantLib 1.11 [2, 4] from its github or sourceforge download page, unless already installed; QuantLib needs to be located in this project directory `ORE-<VERSION>`. Alternatively, you can create a symbolic link named QuantLib here that points to the actual QuantLib directory
- Build QuantExt, OREData, OREAnalytics, App (in this order) as described in section 4.2.3
- Note that ThirdPartyLibs does not need to be built, it contains RapdidXml, header only code for reading and writing XML files
- Move on to section 4.3 and the examples in section 5.

Open `Docs/html/index.html` to see the API documentation for QuantExt, OREData and OREAnalytics, generated by doxygen.

4.2 Building ORE

ORE's source code is hosted on github.com at `https://github.com/opensourcerisk/engine` using `git`, a free and open source distributed version control system.

4.2.1 Git

To access the current code base on GitHub, one needs to get `git` installed first.

1. Install and setup Git on your machine following instructions at [5]

2. Fetch ORE from github by running the following:

   ```
   % git clone https://github.com/opensourcerisk/engine.git ore
   ```

   This will create a folder 'ore' in your current directory that contains the codebase.

3. Initially, the QuantLib subdirectory under `ore` is empty as it is a submodule pointing to the official QuantLib repository. To pull down locally, use the following commands:

   ```
   % cd ore
   % git submodule init
   % git submodule update
   ```

4.2.2 Boost

QuantLib and ORE depend on the boost C++ libraries. Hence these need to be installed before building QuantLib and ORE. On all platforms the minimum required boost version is 1_63 as of ORE release 4.

Windows

1. Download the pre-compiled binaries for MSVC-14 (MSVC2015) from [6]

   - 32-bit: [6]\VERSION\boost_VERSION-msvc-14.0-32.exe\download
   - 64-bit: [6]\VERSION\boost_VERSION-msvc-14.0-64.exe\download

2. Start the installation file and choose an installation folder. Take a note of that folder as it will be needed later on.

3. Finish the installation by clicking Next a couple of times.

Alternatively, compile all Boost libraries directly from the source code:

1. Open a Visual Studio Tools Command Prompt

   - 32-bit: VS2015/VS2013 x86 Native Tools Command Prompt
   - 64-bit: VS2015/VS2013 x64 Native Tools Command Prompt

2. Navigate to the boost root directory

3. Run bootstrap.bat

4. Build the libraries from the source code

   - 32-bit:
     ```
     .\b2 --stagedir=.\lib\Win32\lib --build-type=complete toolset=msvc-14.0 \
     address-model=32 --with-test --with-system --with-filesystem \
     --with-serialization --with-regex --with-date_time stage
     ```

- 64-bit:
```
.\b2 --stagedir=.\lib\x64\lib --build-type=complete toolset=msvc-14.0 \
address-model=64 --with-test --with-system --with-filesystem \
--with-serialization --with-regex --with-date_time stage
```

Unix

1. Download Boost from [7] and build following the instructions on the site
2. Define the environment variable BOOST that points to the boost directory (so includes should be in BOOST and libs should be in BOOST/stage/lib)

4.2.3 ORE Libraries and Application

Windows

1. Download and install Visual Studio Community Edition (Version 2013 or later). During the installation, make sure you install the Visual C++ support under the Programming Languages features (disabled by default).
2. To configure the boost paths in Visual Studio open any of the Visual Studio solution files in item 3 below and select View → Other Windows → Property Manager. It does not matter which solution you open, if it is for example the QuantExt solution you should see two Projects 'QuantExt' and 'quantexttest-suite' in the property manager. Expand any of them (e.g. QuantExt) and then one of the Win32 or x64 configurations. The settings will be specific for the Win32 or x64 configuration but otherwise it does not matter which of the projects or configurations you expand, they all contain the same configuration file. You should now see 'Microsoft.Cpp.Win32.user' respectively 'Microsoft.Cpp.x64.user' depending on whether you chose a Win32 or a x64 configuration. Click on this file to open the property pages. Select VC++ Directories and then add your boost directory to the 'Include Directories' entry. Likewise add your boost library directory to the 'Library Directories' entry. If for example your boost installation is in `C:\boost_1_57_0` and the libraries reside in the `stage\lib` subfolder, add `C:\boost_1_57_0` to the 'Include Directories' entry and `C:\boost_1_57_0\stage\lib` to the 'Library Directories' entry. Press OK. (Alternatively, create and use an environment variable `%BOOST%` pointing to your directory `C:\boost_1_57_0` instead of the directory itself.) If you want to configure the boost paths for Win32 resp. x64 as well, repeat the previous step for 'Microsoft.Cpp. Win32.user' respectively 'Microsoft.Cpp.x64.user'. To complete the configuration just close the property manager window.
3. Open each of the sub-projects and compile them in the following order: QuantLib, QuantExt, OREData, OREAnalytics and App. For each project, do the following:
   - Switch to the correct platform (i.e. Win32 or x64) from the Configuration Manager. The selection should match the pre-compiled version of Boost. Trying to compile using a mixed configuration (e.g. Boost 64-bit and 32-bit QuantLib) will fail.
   - Compile the project: Build → Build Solution
   - Once the compilation is complete, run the test suite.

Alternatively, open the `oreEverything_*.sln` and build the entire solution (again, make sure to select the correct platform in the configuration manager first).

Unix

1. Build QuantLib as usual.

```
% cd QuantLib
% ./autogen.sh
% ./configure --with-boost-include=$BOOST --with-boost-lib=$BOOST/stage/lib
% make -j4
```

2. Build QuantExt

```
% cd QuantExt
% ./autogen.sh
% ./configure
% make -j4
```

   This will build both the QuantExt library and test suite.

3. Run the test suite

```
% ./test/quantext-test-suite
```

4. Build OREData, OREAnalytics and their test suites.

   Follow the same steps as for QuantExt. To run the unit test suites, do

```
% ./test/ored-test-suite
```

   and

```
% ./test/orea-test-suite
```

   in the respective library directories.

5. Build App/ore

```
% cd App
% ./autogen.sh
% ./configure
% make -j4
```

   Note: On Linux systems, the 'locale' settings can negatively affect the ORE process and output. To avoid this, we recommend setting the environment variable `LC_NUMERIC` to `C`, e.g. in a bash shell, do

```
% export LC_NUMERIC=C
```

   before running ORE or any of the examples below. This will suppress thousand separators in numbers when converted to strings.

6. Run Examples (see section 5)

```
% cd Examples/Example_1
% python run.py
```

Building on Unix with CMake

Recent releases of ORE (since end of 2018) can be built with CMake on Unix systems, as follows.

1. Change to the ORE project directory that contains the `QuantLib`, `QuantExt`, etc, folders; create subdirectory `build` and change to subdirectory `build`

2. Configure CMake by invoking

   `cmake -DBOOST_ROOT=$BOOST -DBOOST_LIBRARYDIR=$BOOST/stage/lib ..`

   Alternatively, set environment variables `BOOST_ROOT` and `BOOST_LIBRARYDIR` and run

   `cmake ..`

3. Build all ORE libraries including QuantLib by invoking

   `make -j4`

4. Build all ORE libraries including QuantLib plus the API documentation for QuantExt, OREData and OREAnalytics, generated by doxygen, by invoking

   `make all -j4`

5. Alternatively the individual API documentation for each library can be built by invoking

   `make doc_quantext`

   `make doc_ored`

   `make doc_orea`

6. Unset `LD_LIBRARY_PATH` respectively `DYLD_LIBRARY_PATH` before running the ORE executable or the test suites below, in order not to override the rpath information embedded into the libaries built with CMake

7. Run all test suites by invoking

   `ctest -j4`

Notice that if the boost libraries are not installed they might not be found during runtime because of a missing rpath tag in their path. Run the script `rename_libs.sh` to set the rpath tag in all libraries located in `$BOOST/stage/lib`.
The .cpp and .hpp files included in the build process need to be explicitly specified in the various `CMakeLists.txt` files in the project directory, as in the `Makefile.am` files of the GNU build system. However for the CMake build system we have provided a python script (in `Tools/update_cmake_files.py`) that automates this process.
The advantage of the CMake build system is that it builds fewer intermediate libraries and is hence significantly faster than the GNU automake based build. Moreover it can be extended to cover both Unix and Windows builds.

Building on Windows with CMake

The same set of CMakeLists.txt files allows building ORE on Windows. The following instructions use the Ninja build system (ninja-build.org) that covers the role of make on Unix systems and calls the Visual Studio C++ compiler and linker.

1. Open a command prompt, change to directory C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC and run

   ```
   vcvarsall.bat x64
   ```

   to initialize the compiler-related environment.

2. Change to the ORE project directory that contains the QuantLib, QuantExt, etc, folders; create subdirectory build and change to subdirectory build

3. Configure CMake e.g. by invoking

   ```
   cmake -D BOOST_LIBRARYDIR=/path/to/boost/root/lib64-msvc-14.0  \
             -D CMKAE_BUILD_TYPE=Release \
             -D MSVC_RUNTIME=static \
             -G Ninja ..
   ```

   where the -G option allows choosing the desired build system generator.

4. Build all ORE libraries including QuantLib by invoking

   ```
   ninja
   ```

   Ninja automatically utilizes all available threads, unless specified with the -j option.

5. Run all test suites by invoking

   ```
   ctest -j4
   ```

4.3 Python and Jupyter

Python (version 3.5 or higher) is required to run the examples in section 5 and plot exposure evolutions. Moreover, we use Jupyter [8] in section 6 to visualise simulation results. Both are part of the 'Anaconda Open Data Science Analytics Platform' [9]. Anaconda installation instructions for Windows, OS X and Linux are available on the Anaconda site, with graphical installers for Windows[1], Linux and OS X.
With Linux and OS X, the following environment variable settings are required

- set LANG and LC_ALL to en_US.UTF-8 or en_GB.UTF-8
- set LC_NUMERIC to C.

The former is required for both running the Python scripts in the examples section, as well as successful installation of the following packages.
The full functionality of the Jupyter notebook introduced in section 6.1 requires furthermore installing

[1]With Windows, after a fresh installation of Python the user may have to run the python command once in a command shell so that the Python executable will be found subsequently when running the example scripts in section 5.

- jupyter_dashboards: `https://github.com/jupyter-incubator/dashboards`
- ipywidgets: `https://github.com/ipython/ipywidgets`
- pythreejs: `https://github.com/jovyan/pythreejs`
- bqplot: `https://github.com/bloomberg/bqplot`

With Python and Anaconda already installed, this can be done by running these commands

- `conda install -c conda-forge ipywidgets`
- `pip install jupyter_dashboards`
- `jupyter dashboards quick-setup --sys-prefix`
- `conda install -c conda-forge bqplot`
- `conda install -c conda-forge pythreejs`

Note that the bqplot installation requires the environment settings mentioned above.

5 Examples

The examples shown in table 2 are intended to help with getting started with ORE, and to serve as plausibility checks for the simulation results generated with ORE.
All example results can be produced with the Python scripts `run.py` in the ORE release's `Examples/Example_#` folders which work on both Windows and Unix platforms. In a nutshell, all scripts call ORE's command line application with a single input XML file

```
ore[.exe] ore.xml
```

They produce a number of standard reports and exposure graphs in PDF format. The structure of the input file and of the portfolio, market and other configuration files referred to therein will be explained in section 7.

ORE is driven by a number of input files, listed in table 3 and explained in detail in sections 7 to 11. In all examples, these input files are either located in the example's sub directory `Examples/Example_#/Input` or the main input directory `Examples/Input` if used across several examples. The particular selection of input files is determined by the 'master' input file `ore.xml`.
The typical list of output files and reports is shown in table 4. The names of output files can be configured through the master input file `ore.xml`. Whether these reports are generated also depends on the setting in `ore.xml`. For the examples, all output will be written to the directory `Examples/Example_#/Output`.
Note: When building ORE from sources on Windows platforms, make sure that you copy your `ore.exe` to the binary directory `bin/win32/` respectively `bin/x64/`. Otherwise the examples may be run using the pre-compiled executables which come with the ORE release.

Example	Description
1	Vanilla at-the-money Swap with flat yield curve
2	Vanilla Swap with normal yield curve
3	European Swaption
4	Bermudan Swaption
5	Callable Swap
6	Cap/Floor
7	FX Forward European FX Option
8	Cross Currency Swap without notional reset
9	Cross Currency Swap with notional reset
10	Three-Swap portfolio with netting and collateral XVAs - CVA, DVA, FVA, MVA, COLVA Exposure and XVA Allocation to trade level
11	Basel exposure measures - EE, EPE, EEPE
12	Long term simulation with horizon shift
13	Dynamic Initial Margin and MVA
14	Minimal Market Data Setup
15	Sensitivity Analysis and Stress Testing
16	Equity Derivatives Exposure
17	Inflation Swap Exposure
18	Bonds and Amortisation Structures
19	Swaption Pricing with Smile
20	Credit Default Swap Pricing
21	Constant Maturity Swap Pricing
22	Option Sensitivity Analysis with Smile
23	Forward Rate Agreement and Averaging OIS Exposure
24	Commodity Forward and Option Pricing and Sensitivity
25	CMS Spread with (Digital) Cap/Floor Pricing, Sensitivity and Exposures
26	Bootstrap Consistency
27	BMA Basis Swap Pricing and Sensitivity
28	Discount Ratio Curves
29	Curve Building using Fixed vs. Float Cross Currency Helpers

Table 2: ORE examples.

5.1 Interest Rate Swap Exposure, Flat Market

We start with a vanilla single currency Swap (currency EUR, maturity 20y, notional 10m, receive fixed 2% annual, pay 6M-Euribor flat). The market yield curves (for both discounting and forward projection) are set to be flat at 2% for all maturities, i.e. the Swap is at the money initially and remains at the money on average throughout its life. Running ORE in directory `Examples/Example_1` with

```
python run.py
```

yields the exposure evolution in

```
Examples/Example_1/Output/*.pdf
```

and shown in figure 2. Both Swap simulation and Swaption pricing are run with calls to the ORE executable, essentially

```
ore[.exe] ore.xml
ore[.exe] ore_swaption.xml
```

which are wrapped into the script `Examples/Example_1/run.py` provided with the ORE release. It is instructive to look into the input folder in Examples/Example_1, the content of the main input file `ore.xml`, together with the explanations in section 7.
This simple example is an important test case which is also run similarly in one of the unit test suites of ORE. The expected exposure can be seen as a European option on the underlying netting set, see also appendix A.3. In this example, the expected exposure at

File Name	Description
`ore.xml`	Master input file, selection of further inputs below and selection of analytics
`portfolio.xml`	Trade data
`netting.xml`	Collateral (CSA) data
`simulation.xml`	Configuration of simulation model and market
`market.txt`	Market data snapshot
`fixings.txt`	Index fixing history
`curveconfig.xml`	Curve and term structure composition from individual market instruments
`conventions.xml`	Market conventions for all market data points
`todaysmarket.xml`	Configuration of the market composition, relevant for the pricing of the given portfolio as of today (yield curves, FX rates, volatility surfaces etc)
`pricingengines.xml`	Configuration of pricing methods by product

Table 3: ORE input files

File Name	Description
`npv.csv`	NPV report
`flows.csv`	Cashflow report
`curves.csv`	Generated yield (discount) curves report
`xva.csv`	XVA report, value adjustments at netting set and trade level
`exposure_trade_*.csv`	Trade exposure evolution reports
`exposure_nettingset_*.csv`	Netting set exposure evolution reports
`rawcube.csv`	NPV cube in readable text format
`netcube.csv`	NPV cube after netting and colateral, in readable text format
`*.dat`	Intermediate storage of NPV cube and scenario data in binary format
`*.pdf`	Exposure graphics produced by the python script `run.py` after ORE completed

Table 4: ORE output files

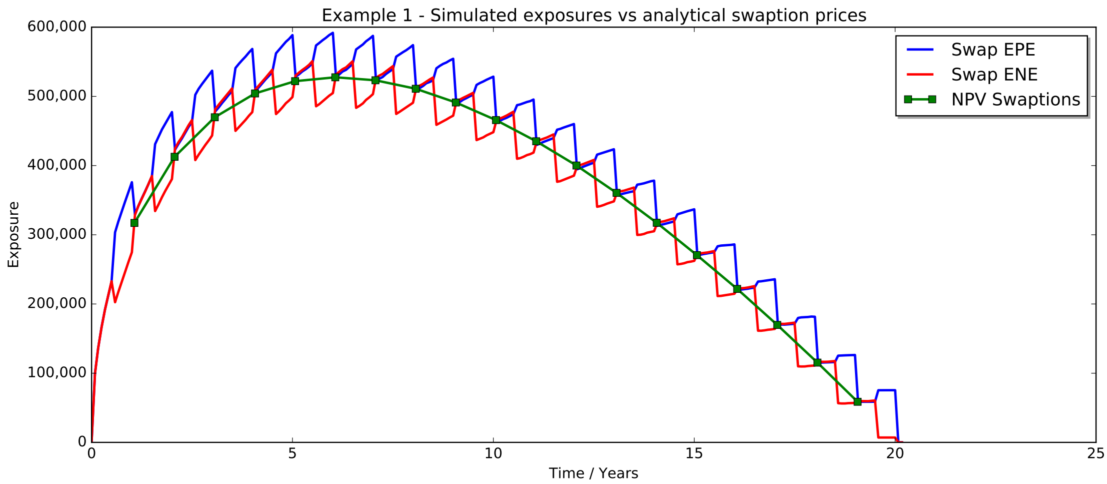

Figure 2: Vanilla ATM Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are European Swaption prices. The simulation was run with monthly time steps and 10,000 Monte Carlo samples to demonstrate the convergence of EPE and ENE profiles. A similar outcome can be obtained more quickly with 5,000 samples on a quarterly time grid which is the default setting of Example_1.

some future point in time, say 10 years, is equal to the European Swaption price for an option with expiry in 10 years, underlying Swap start in 10 years and underlying Swap maturity in 20 years. We can easily compute such standard European Swaption prices for all future points in time where both Swap legs reset, i.e. annually in this case[2]. And

[2] Using closed form expressions for standard European Swaption prices.

if the simulation model has been calibrated to the points on the Swaption surface which are used for European Swaption pricing, then we can expect to see that the simulated exposure matches Swaption prices at these annual points, as in figure 2. In Example_1 we used co-terminal ATM Swaptions for both model calibration and Swaption pricing. Moreover, as the the yield curve is flat in this example, the exposures from both parties' perspectives (EPE and ENE) match not only at the annual resets, but also for the period between annual reset of both legs to the point in time when the floating leg resets. Thereafter, between floating leg (only) reset and next joint fixed/floating leg reset, we see and expect a deviation of the two exposure profiles.

5.2 Interest Rate Swap Exposure, Realistic Market

Moving to `Examples/Example_2`, we see what changes when using a realistic (non-flat) market environment. Running the example with

```
python run.py
```

yields the exposure evolution in

```
Examples/Example_2/Output/*.pdf
```

shown in figure 3. In this case, where the curves (discount and forward) are upward

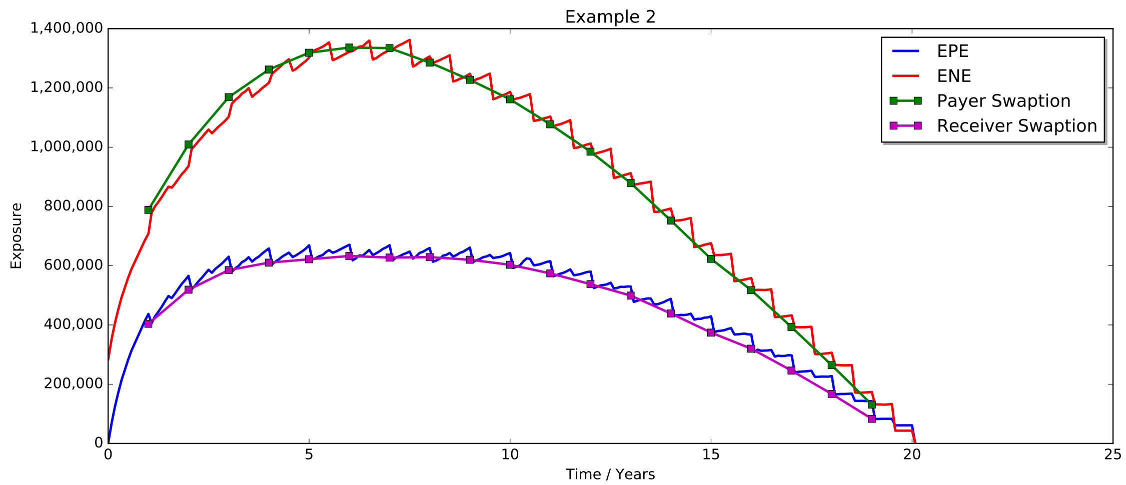

Figure 3: Vanilla ATM Swap expected exposure in a realistic market environment as of 05/02/2016 from both parties' perspectives. The Swap is the same as in figure 2 but receiving fixed 1%, roughly at the money. The symbols are the prices of European payer and receiver Swaptions. Simulation with 5000 paths and monthly time steps.

sloping, the receiver Swap is at the money at inception only and moves (on average) out of the money during its life. Similarly, the Swap moves into the money from the counterparty's perspective. Hence the expected exposure evolutions from our perspective (EPE) and the counterparty's perspective (ENE) 'detach' here, while both can still be be reconciled with payer or respectively receiver Swaption prices.

5.3 European Swaption Exposure

This demo case in folder `Examples/Example_3` shows the exposure evolution of European Swaptions with cash and physical delivery, respectively, see figure 4. The delivery type

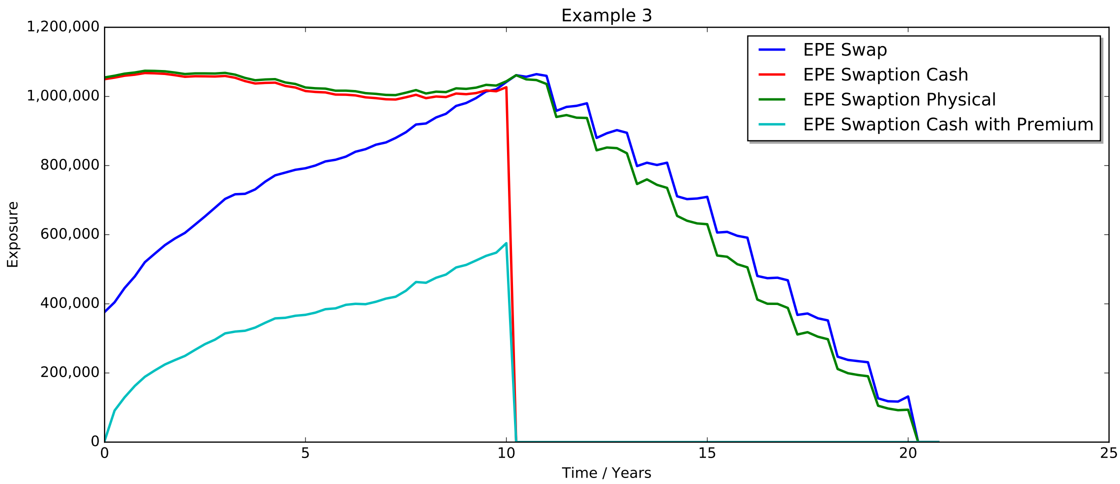

Figure 4: European Swaption exposure evolution, expiry in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

(cash vs physical) yields significantly different valuations as of today due to the steepness of the relevant yield curves (EUR). The cash settled Swaption's exposure graph is truncated at the exercise date, whereas the physically settled Swaption exposure turns into a Swap-like exposure after expiry. For comparison, the example also provides the exposure evolution of the underlying forward starting Swap which yields a somewhat higher exposure after the forward start date than the physically settled Swaption. This is due to scenarios with negative Swap NPV at expiry (hence not exercised) and positive NPVs thereafter. Note the reduced EPE in case of a Swaption with settlement of the option premium on exercise date.

5.4 Bermudan Swaption Exposure

This demo case in folder `Examples/Example_4` shows the exposure evolution of Bermudan rather than European Swaptions with cash and physical delivery, respectively, see figure 5. The underlying Swap is the same as in the European Swaption example in section 5.3. Note in particular the difference between the Bermudan and European Swaption exposures with cash settlement: The Bermudan shows the typical step-wise decrease due to the series of exercise dates. Also note that we are using the same Bermudan option pricing engines for both settlement types, in contrast to the European case, so that the Bermudan option cash and physical exposures are identical up to the first exercise date. When running this example, you will notice the significant difference in computation time compared to the European case (ballpark 30 minutes here for 2 Swaptions, 1000 samples, 90 time steps). The Bermudan example takes significantly more computation time because we use an LGM grid engine for pricing under scenarios in this case. In a realistic context one would more likely resort to American Monte Carlo simulation, feasible in ORE, but not provided in the current release. However, this

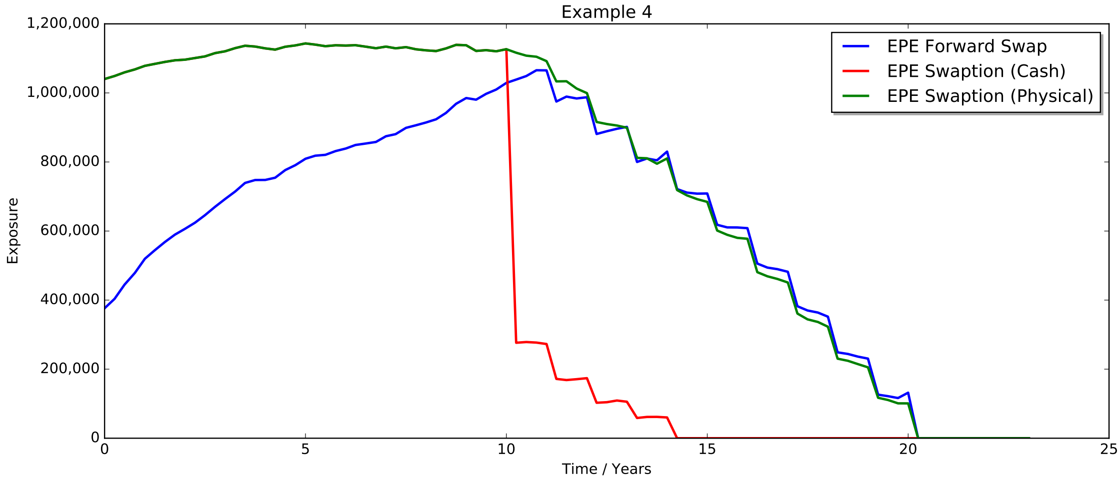

Figure 5: Bermudan Swaption exposure evolution, 5 annual exercise dates starting in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

implementation can be used to benchmark any faster / more sophisticated approach to Bermudan Swaption exposure simulation.

5.5 Callable Swap Exposure

This demo case in folder `Examples/Example_5` shows the exposure evolution of a European callable Swap, represented as two trades - the non-callable Swap and a Swaption with physical delivery. We have sold the call option, i.e. the Swaption is a right for the counterparty to enter into an offsetting Swap which economically terminates all future flows if exercised. The resulting exposure evolutions for the individual components (Swap, Swaption), as well as the callable Swap are shown in figure 6. The example is an extreme case where the underlying Swap is deeply in the money (receiving fixed 5%), and hence the call exercise probability is close to one. Modify the Swap and Swaption fixed rates closer to the money ($\approx$ 1%) to see the deviation between net exposure of the callable Swap and the exposure of a 'short' Swap with maturity on exercise.

5.6 Cap/Floor Exposure

The example in folder `Examples/Example_6` generates exposure evolutions of several Swaps, caps and floors. The example shown in figure 7 ('portfolio 1') consists of a 20y Swap receiving 3% fixed and paying Euribor 6M plus a long 20y Collar with both cap and floor at 4% so that the net exposure corresponds to a Swap paying 1% fixed.
The second example in this folder shown in figure 8 ('portfolio 2') consists of a short Cap, long Floor and a long Collar that exactly offsets the netted Cap and Floor.
Further three test portfolios are provided as part of this example. Run the example and inspect the respective output directories `Examples/Example_7/Output/portfolio_#`. Note that these directories have to be present/created before running the batch with `python run.py`.

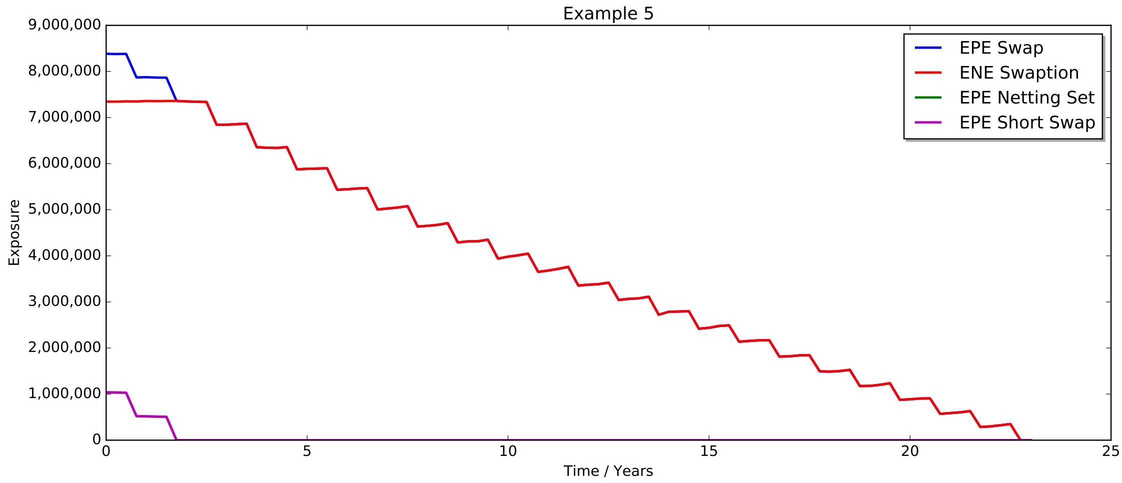

Figure 6: European callable Swap represented as a package consisiting of non-callable Swap and Swaption. The Swaption has physical delivery and offsets all future Swap cash flows if exercised. The exposure evolution of the package is shown here as 'EPE NettingSet' (green line). This is covered by the pink line, the exposure evolution of the same Swap but with maturity on the exercise date. The graphs match perfectly here, because the example Swap is deep in the money and exercise probability is close to one. Simulation with 5000 paths and quarterly time steps.

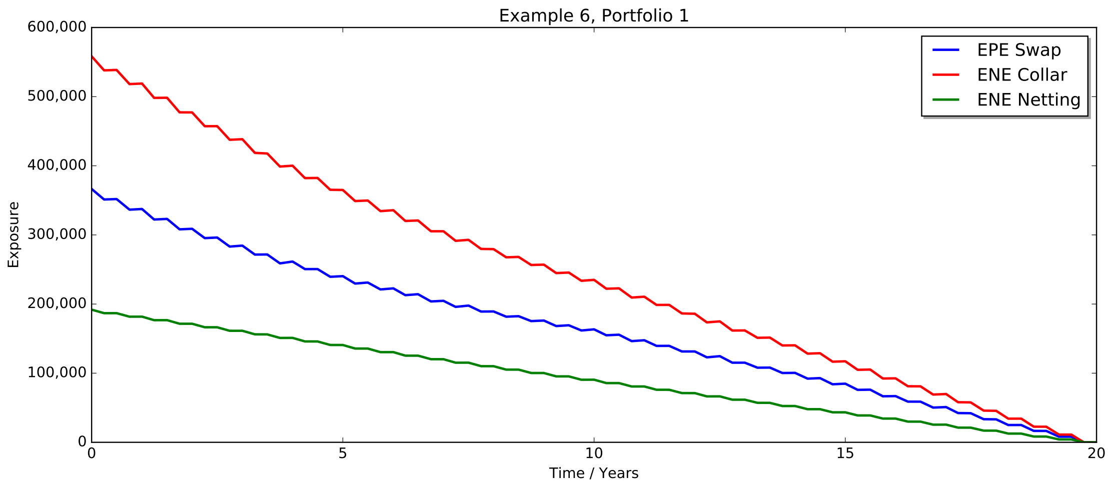

Figure 7: Swap+Collar, portfolio 1. The Collar has identical cap and floor rates at 4% so that it corresponds to a fixed leg which reduces the exposure of the Swap, which receives 3% fixed. Simulation with 1000 paths and quarterly time steps.

5.7 FX Forward and FX Option Exposure

The example in folder `Examples/Example_7` generates the exposure evolution for a EUR / USD FX Forward transaction with value date in 10Y. This is a particularly simple show case because of the single cash flow in 10Y. On the other hand it checks the cross currency model implementation by means of comparison to analytic limits - EPE and ENE at the trade's value date must match corresponding Vanilla FX Option prices, as

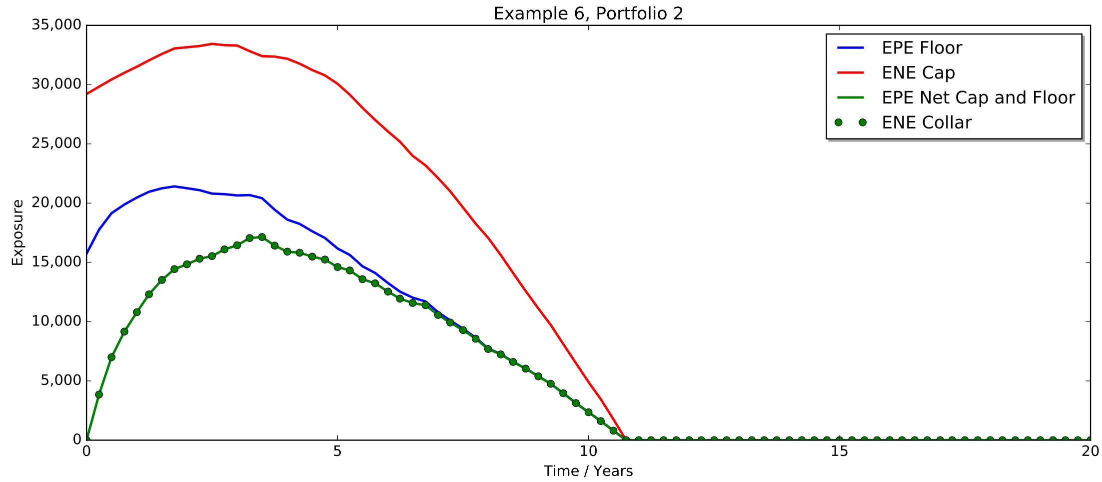

Figure 8: Short Cap and long Floor vs long Collar, portfolio 2. Simulation with 1000 paths and quarterly time steps.

shown in figure 9.

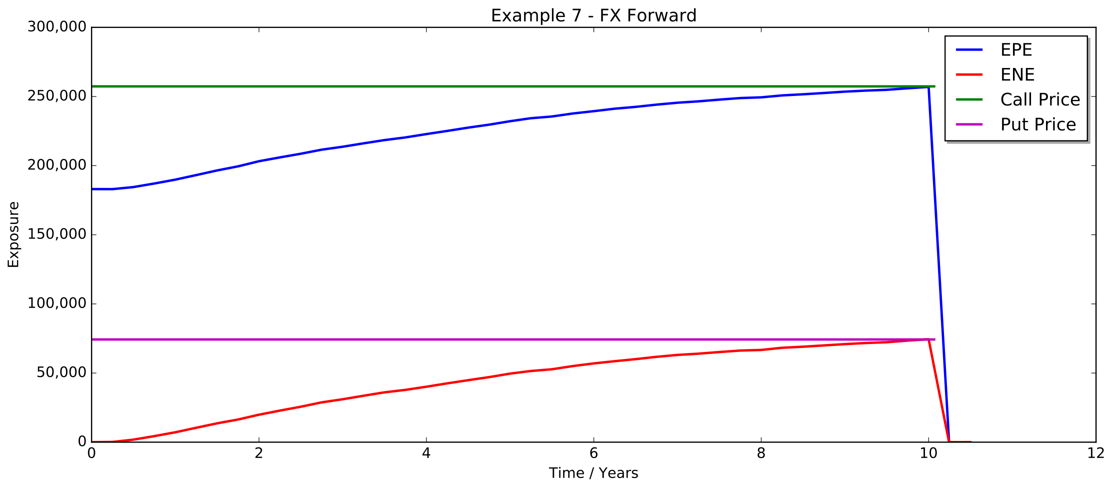

Figure 9: EUR/USD FX Forward expected exposure in a realistic market environment as of 26/02/2016 from both parties' perspectives. Value date is obviously in 10Y. The flat lines are FX Option prices which coincide with EPE and ENE, respectively, on the value date. Simulation with 5000 paths and quarterly time steps.

FX Option Exposure

This example (in folder `Examples/Example_7`, as the FX Forward example) illustrates the exposure evolution for an FX Option, see figure 10. Recall that the FX Option value

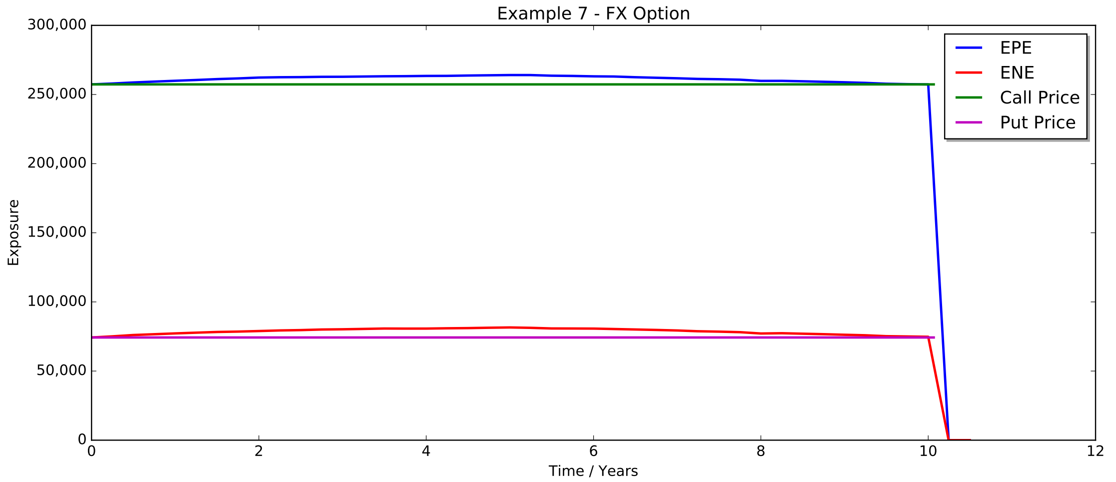

Figure 10: EUR/USD FX Call and Put Option exposure evolution, same underlying and market data as in section 5.7, compared to the call and put option price as of today (flat line). Simulation with 5000 paths and quarterly time steps.

$NPV(t)$ as of time $0 \leq t \leq T$ satisfies

$$\frac{NPV(t)}{N(t)} = \text{Nominal} \times \mathbb{E}_t \left[\frac{(X(T) - K)^+}{N(T)} \right]$$

$$NPV(0) = \mathbb{E} \left[\frac{NPV(t)}{N(t)} \right] = \mathbb{E} \left[\frac{NPV^+(t)}{N(t)} \right] = EPE(t)$$

One would therefore expect a flat exposure evolution up to option expiry. The deviation from this in ORE's simulation is due to the pricing approach chosen here under scenarios. A Black FX option pricer is used with deterministic Black volatility derived from today's volatility structure (pushed or rolled forward, see section 7.4.3). The deviation can be removed by extending the volatility modelling, e.g. implying model consistent Black volatilities in each simulation step on each path.

5.8 Cross Currency Swap Exposure, without FX Reset

The case in `Examples/Example_8` is a vanilla cross currency Swap. It shows the typical blend of an Interest Rate Swap's saw tooth exposure evolution with an FX Forward's exposure which increases monotonically to final maturity, see figure 11.

5.9 Cross Currency Swap Exposure, with FX Reset

The effect of the FX resetting feature, common in Cross Currency Swaps nowadays, is shown in `Examples/Example_9`. The example shows the exposure evolution of a EUR/USD cross currency basis Swap with FX reset at each interest period start, see figure 12. As expected, the notional reset causes an exposure collapse at each period start when the EUR leg's notional is reset to match the USD notional.

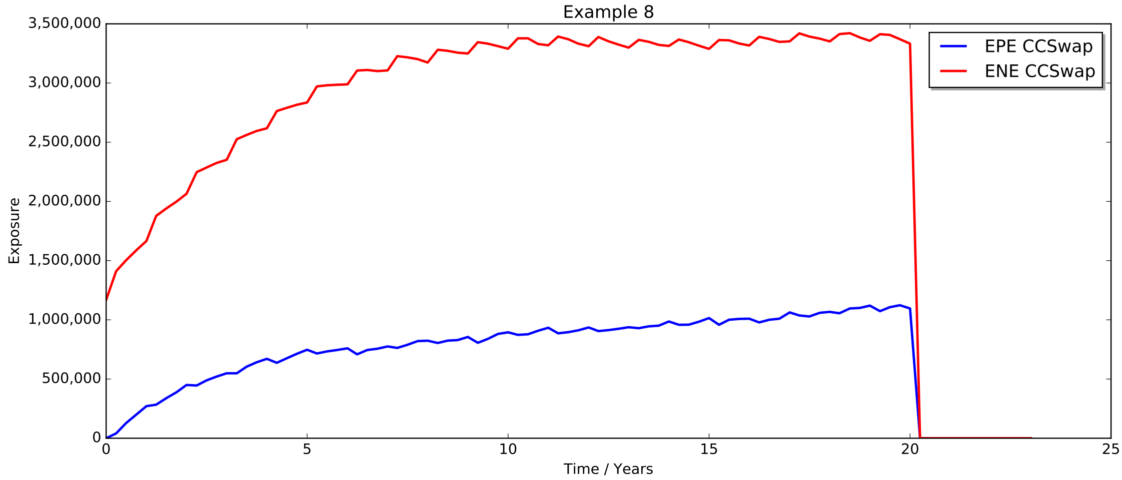

Figure 11: Cross Currency Swap exposure evolution without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

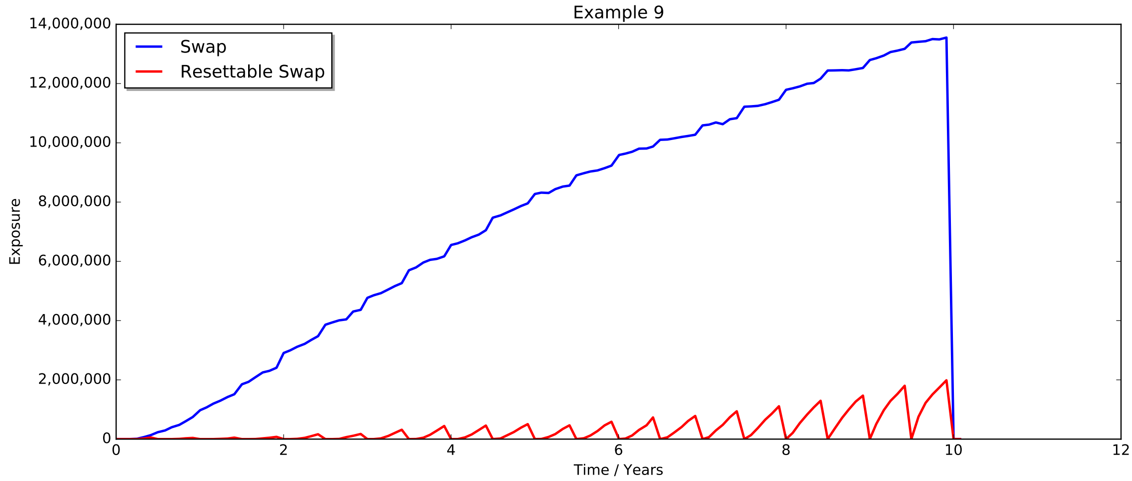

Figure 12: Cross Currency Basis Swap exposure evolution with and without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

5.10 Netting Set, Collateral, XVAs, XVA Allocation

In this example (see folder `Examples/Example_10`) we showcase a small netting set consisting of three Swaps in different currencies, with different collateral choices

- no collateral - figure 13,
- collateral with threshold (THR) 1m EUR, minimum transfer amount (MTA) 100k EUR, margin period of risk (MPOR) 2 weeks - figure 14
- collateral with zero THR and MTA, and MPOR 2w - figure 15

The exposure graphs with collateral and positive margin period of risk show typical spikes. What is causing these? As sketched in appendix A.11, ORE uses a *classical* *collateral model* that applies collateral amounts to offset exposure with a time delay that corresponds to the margin period of risk. The spikes are then caused by instrument cash flows falling between exposure measurement dates d_1 and d_2 (an MPOR apart), so that a collateral delivery amount determined at d_1 but settled at d_2 differs significantly from the closeout amount at d_2 causing a significant residual exposure for a short period of time. See for example [22] for a recent detailed discussion of collateral modelling. The approach currently implemented in ORE corresponds to *Classical+* in [22], the more conservative approach of the classical methods. The less conservative alternative, *Classical-*, would assume that both parties stop paying trade flows at the beginning of the MPOR, so that the P&L over the MPOR does not contain the cash flow effect, and exposure spikes are avoided. Note that the size and position of the largest spike in figure 14 is consistent with a cash flow of the 40 million GBP Swap in the example's portfolio that rolls over the 3rd of March and has a cash flow on 3 March 2020, a bit more than four years from the evaluation date.

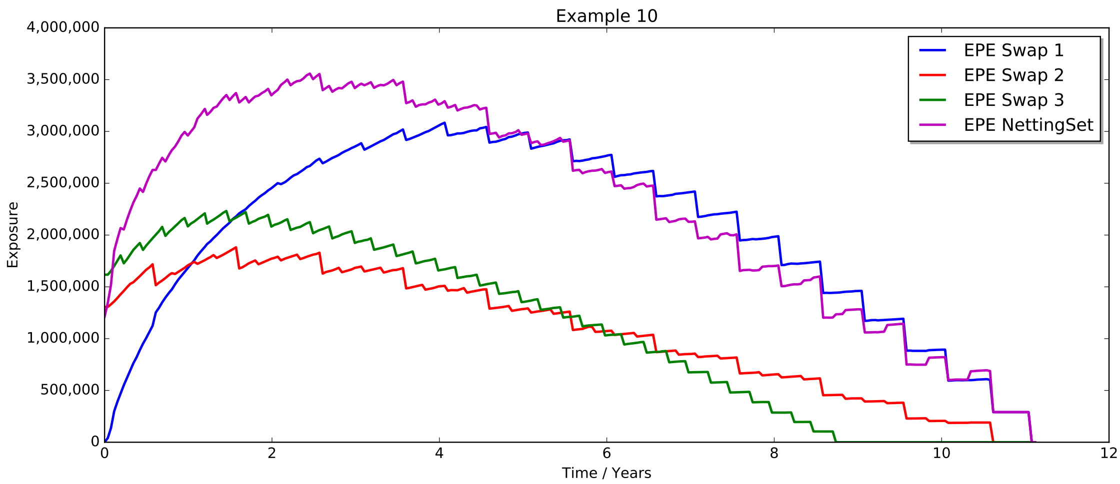

Figure 13: Three Swaps netting set, no collateral. Simulation with 5000 paths and bi-weekly time steps.

CVA, DVA, FVA, COLVA, MVA, Collateral Floor

We use one of the cases in `Examples/Example_10` to demonstrate the XVA outputs, see folder `Examples/Example_10/Output/collateral_threshold_dim`.

The summary of all value adjustments (CVA, DVA, FVA, COLVA, MVA, as well as the Collateral Floor) is provided in file `xva.csv`. The file includes the allocated CVA and DVA numbers to individual trades as introduced in the next section. The following table illustrates the file's layout, omitting the three columns containing allocated data.

TradeId	NettingSetId	CVA	DVA	FBA	FCA	COLVA	MVA	CollateralFloor	BaselEPE	BaselEEPE
	CPTY_A	6,521	151,193	-946	72,103	2,769	-14,203	189,936	113,260	1,211,770
Swap_1	CPTY_A	127,688	211,936	-19,624	100,584	n/a	n/a	n/a	2,022,590	2,727,010
Swap_3	CPTY_A	71,315	91,222	-11,270	43,370	n/a	n/a	n/a	1,403,320	2,183,860
Swap_2	CPTY_A	68,763	100,347	-10,755	47,311	n/a	n/a	n/a	1,126,520	1,839,590

The line(s) with empty TradeId column contain values at netting set level, the others contain uncollateralised single-trade VAs. Note that COLVA, MVA and Collateral Floor are only available at netting set level at which collateral is posted.

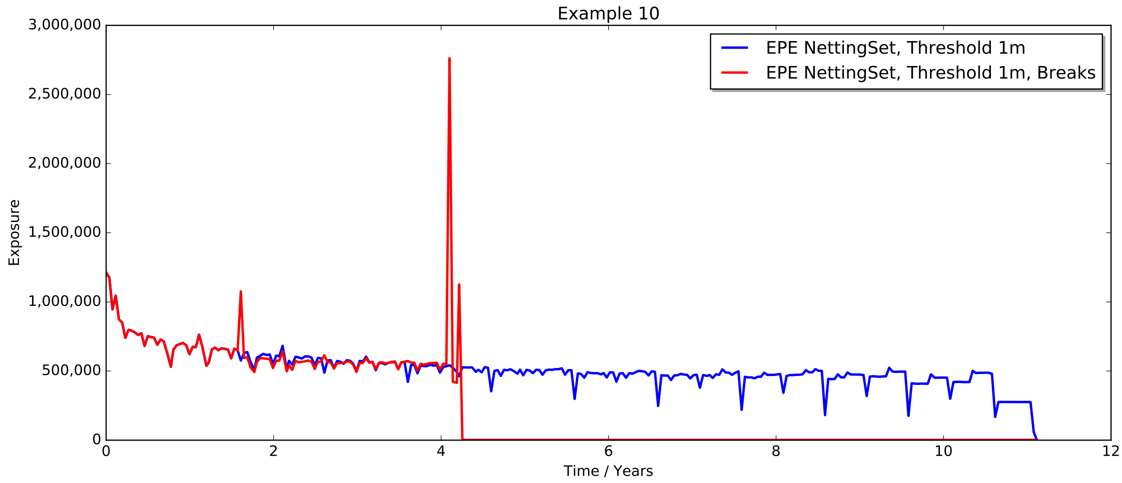

Figure 14: Three Swaps netting set, THR=1m EUR, MTA=100k EUR, MPOR=2w. The red evolution assumes that the each trade is terminated at the next break date. The blue evolution ignores break dates. Simulation with 5000 paths and bi-weekly time steps.

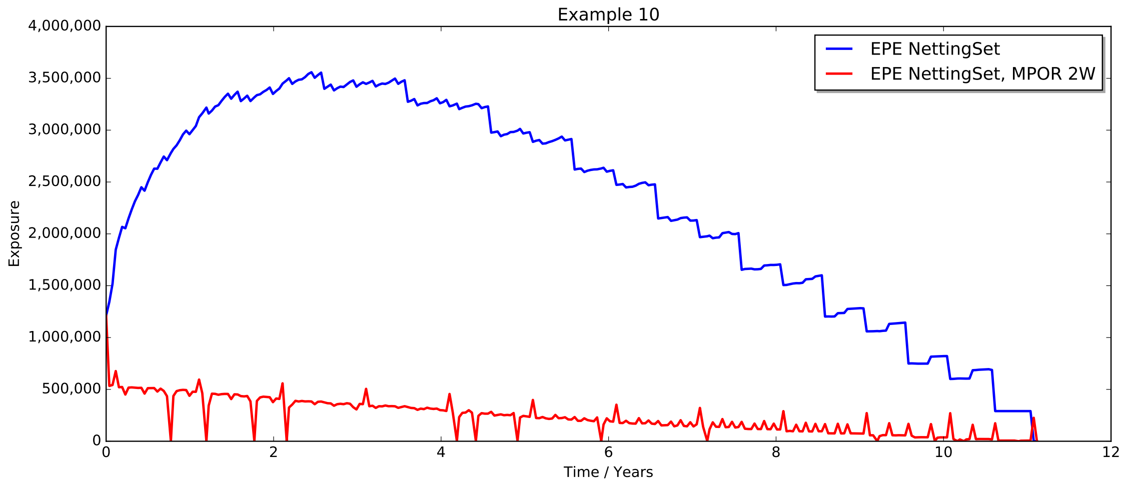

Figure 15: Three Swaps, THR=MTA=0, MPOR=2w. Simulation with 5000 paths and bi-weekly time steps.

Detailed output is written for COLVA and Collateral Floor to file `colva_nettingset_*.csv` which shows the incremental contributions to these two VAs through time.

Exposure Reports & XVA Allocation to Trades

Using the example in folder `Examples/Example_10` we illustrate here the layout of an exposure report produced by ORE. The report shows the exposure evolution of Swap_1 without collateral which - after running Example_10 - is found in folder `Examples/Example_10/Output/collateral_none/exposure_trade_Swap_1.csv`:

TradeId	Date	Time	EPE	ENE	AllocEPE	AllocENE	PFE	BaselEE	BaselEEE
Swap_1	05/02/16	0.0000	0	1,711,748	0	0	0	0	0
Swap_1	19/02/16	0.0383	38,203	1,749,913	-1,200,677	511,033	239,504	38,202	38,202
Swap_1	04/03/16	0.0765	132,862	1,843,837	-927,499	783,476	1,021,715	132,845	132,845
Swap_1	...	...	...	...	...	...	...	...	...

The exposure measures EPE, ENE and PFE, and the Basel exposure measures EE_B and EEE_B, are defined in appendix A.3. Allocated exposures are defined in appendix A.12. The PFE quantile and allocation method are chosen as described in section 7.1.3. In addition to single trade exposure files, ORE produces an exposure file per netting set. The example from the same folder as above is:

NettingSet	Date	Time	EPE	ENE	PFE	ExpectedCollateral	BaselEE	BaselEEE
CPTY_A	05/02/16	0.0000	1,203,836	0	1,203,836	0	1,203,836	1,203,836
CPTY_A	19/02/16	0.0383	1,337,713	137,326	3,403,460	0	1,337,651	1,337,651
CPTY_A	...	...	...	...	...	...	...	...

Allocated exposures are missing here, as they make sense at the trade level only, and the expected collateral balance is added for information (in this case zero as collateralisation is deactivated in this example).

The allocation of netting set exposure and XVA to the trade level is frequently required by finance departments. This allocation is also featured in `Examples/Example_10`. We start again with the uncollateralised case in figure 16, followed by the case with threshold 1m EUR in figure 17. In both cases we apply the *marginal* (Euler) allocation method as

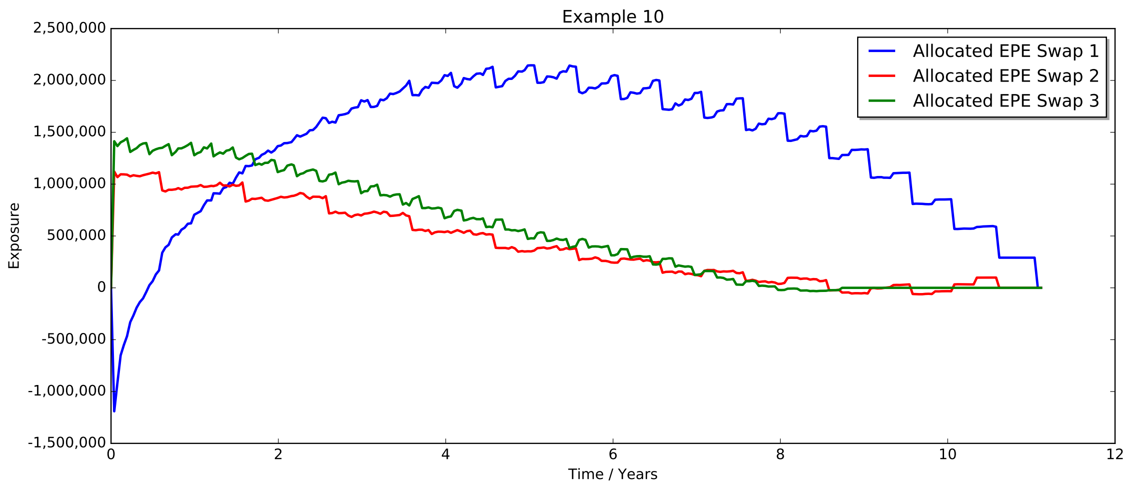

Figure 16: Exposure allocation without collateral. Simulation with 5000 paths and bi-weekly time steps.

published by Pykhtin and Rosen in 2010, hence we see the typical negative EPE for one of the trades at times when it reduces the netting set exposure. The case with collateral moreover shows the typical spikes in the allocated exposures. The analytics results also feature allocated XVAs in file `xva.csv` which are derived from the allocated exposure profiles. Note that ORE also offers alternative allocation methods to the marginal method by Pykhtin/Rosen, which can be explored with `Examples/Example_10`.

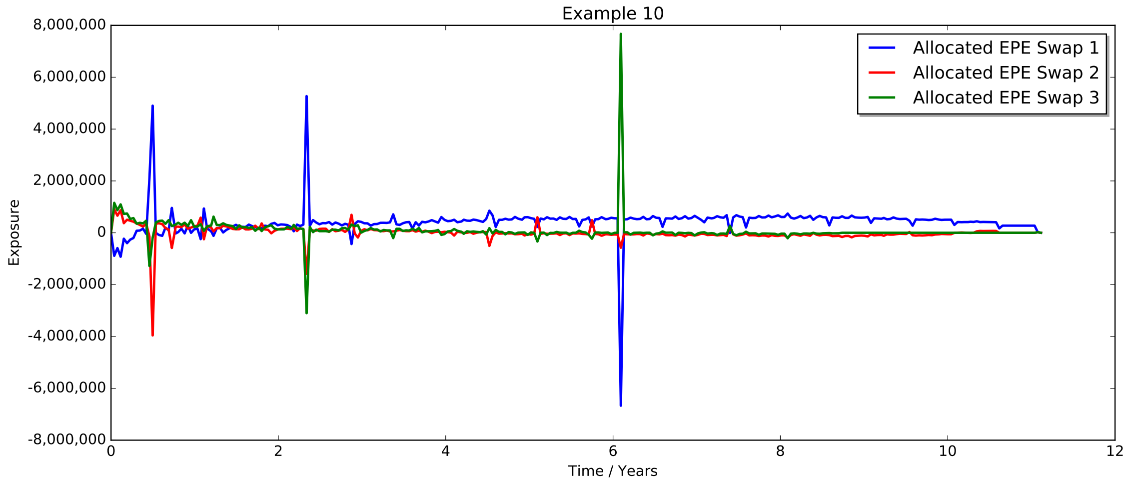

Figure 17: Exposure allocation with collateral and threshold 1m EUR. Simulation with 5000 paths and bi-weekly time steps.

5.11 Basel Exposure Measures

Example `Example_11` demonstrates the relation between the evolution of the expected exposure (EPE in our notation) to the 'Basel' exposure measures EE_B, EEE_B, EPE_B and EEPE_B as defined in appendix A.3. In particular the latter is used in internal model methods for counterparty credit risk as a measure for the exposure at default. It is a 'derivative' of the expected exposure evolution and defined as a time average over the running maximum of EE_B up to the horizon of one year.

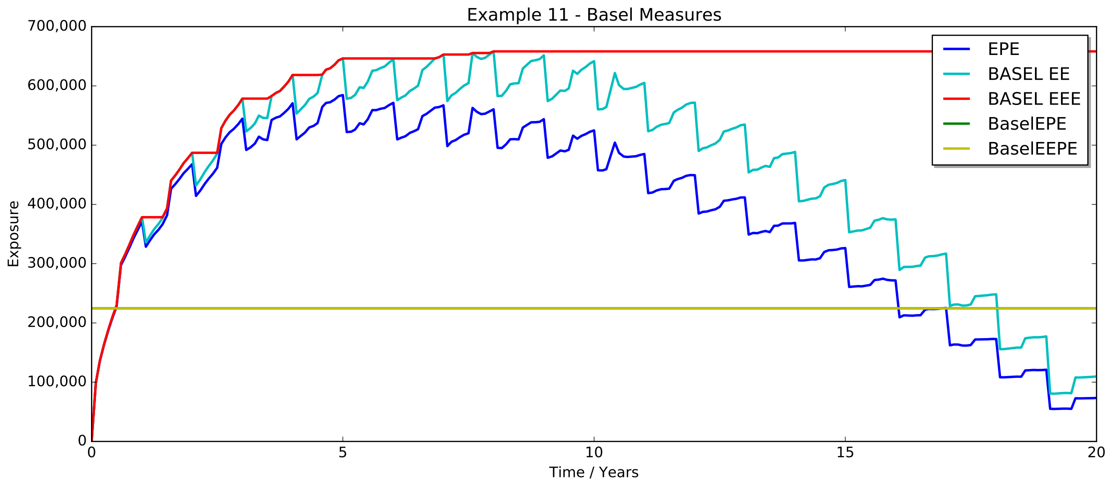

Figure 18: Evolution of the expected exposure of Vanilla Swap, comparison to the 'Basel' exposure measures EEE_B, EPE_B and EEPE_B. Note that the latter two are indistinguishable in this case, because the expected exposure is increasing for the first year.

5.12 Long Term Simulation with Horizon Shift

The example in folder Example_12 finally demonstrates an effect that, at first glance, seems to cause a serious issue with long term simulations. Fortunately this can be avoided quite easily in the Linear Gauss Markov model setting that is used here.
In the example we consider a Swap with maturity in 50 years in a flat yield curve environment. If we simulate this naively as in all previous cases, we obtain a particularly noisy EPE profile that does not nearly reconcile with the known exposure (analytical Swaption prices). This is shown in figure 19 ('no horizon shift'). The origin of this issue is the width of the risk-neutral NPV distribution at long time horizons which can turn out to be quite small so that the Monte Carlo simulation with finite number of samples does not reach far enough into the positive or negative NPV range to adequately sample the distribution, and estimate both EPE and ENE in a single run. Increasing the number of samples may not solve the problem, and may not even be feasible in a realistic setting. The way out is applying a 'shift transformation' to the Linear Gauss Markov model, see Example_12/Input/simulation2.xml in lines 92-95:

```
<ParameterTransformation>
  <ShiftHorizon>30.0</ShiftHorizon>
  <Scaling>1.0</Scaling>
</ParameterTransformation>
```

The effect of the 'ShiftHorizon' parameter T is to apply a shift to the Linear Gauss Markov model's $H(t)$ parameter (see appendix A.1) *after* the model has been calibrated, i.e. to replace:

$$H(t) \rightarrow H(t) - H(T)$$

It can be shown that this leaves all expectations computed in the model (such as EPE and ENE) invariant. As explained in [20], subtracting an H shift effectively means performing a change of measure from the 'native' LGM measure to a T-Forward measure with horizon T, here 30 years. Both negative and positive shifts are permissible, but only negative shifts are connected with a T-Forward measure and improve numerical stability.
In our experience it is helpful to place the horizon in the middle of the portfolio duration to significantly improve the quality of long term expectations. The effect of this change (only) is shown in the same figure 19 ('shifted horizon'). Figure 20 further illustrates the origin of the problem and its resolution: The rate distribution's mean (without horizon shift or change of measure) drifts upwards due to convexity effects (note that the yield curve is flat in this example), and the distribution's width is then too narrow at long horizons to yield a sufficient number of low rate scenarios with contributions to the Swap's *EPE* (it is a floating rate payer). With the horizon shift (change of measure), the distribution's mean is pulled 'back' at long horizons, because the convexity effect is effectively wiped out at the chosen horizon, and the expected rate matches the forward rate.

5.13 Dynamic Initial Margin and MVA

This example in folder Examples/Example_13 demonstrates Dynamic Initial Margin calculations (see also appendix A.8) for a number of elementary products:

- A single currency Swap in EUR (case A),

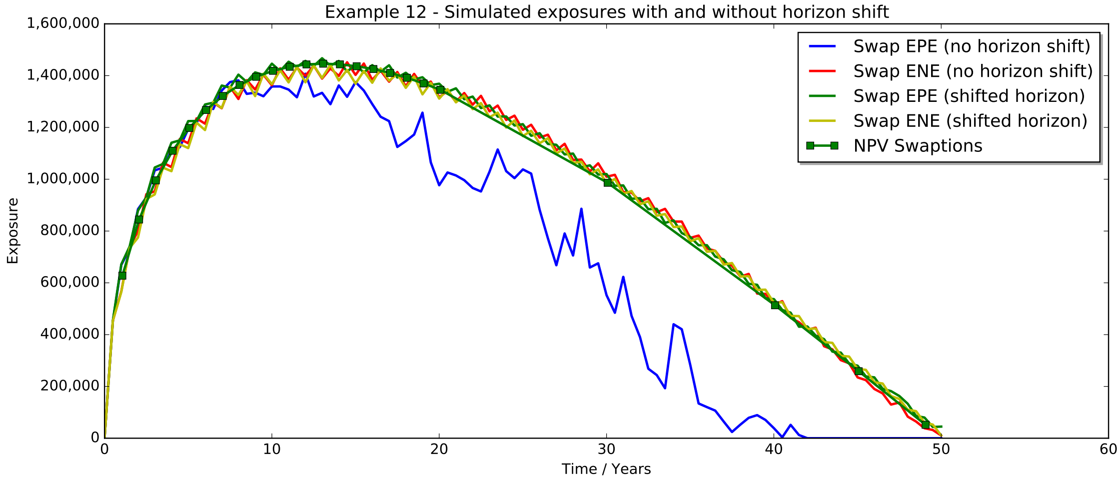

Figure 19: Long term Swap exposure simulation with and without horizon shift.

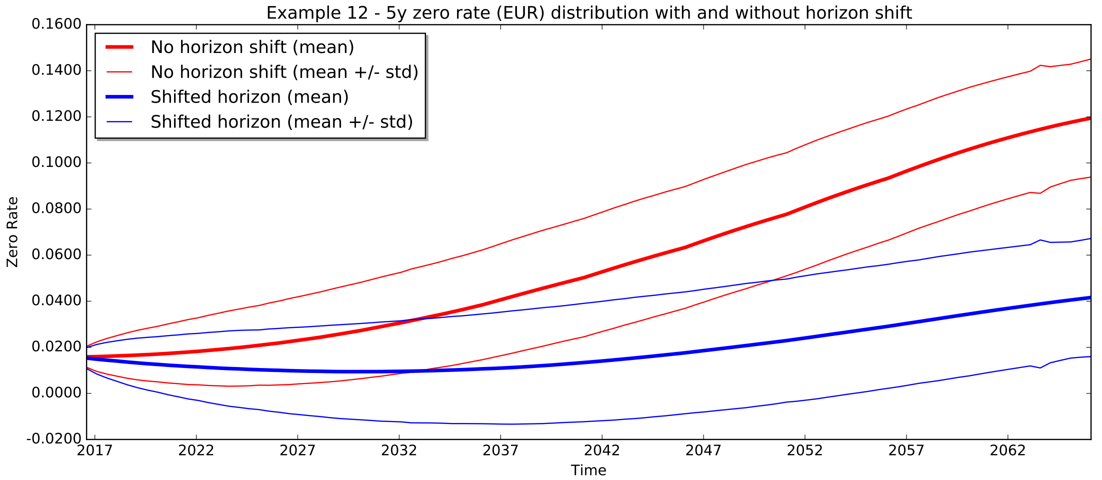

Figure 20: Evolution of rate distributions with and without horizon shift (change of measure). Thick lines indicate mean values, thin lines are contours of the rate distribution at ± one standard devation.

- a European Swaption in EUR with physical delivery (case B),
- a single currency Swap in USD (case C), and
- a EUR/USD cross currency Swap (case D).

The examples can be run as before with

```
python run_A.py
```

and likewise for cases B, C and D. The essential results of each run are are visualised in the form of

- evolution of expected DIM
- regression plots at selected future times

illustrated for cases A and B in figures 21 - 24. In the three swap cases, the regression orders do make a noticable difference in the respective expected DIM evolution. In the Swaption case B, first and second order polynomial choice makes a difference before option expiry. More details on this DIM model and its performance can be found in [21, 23].

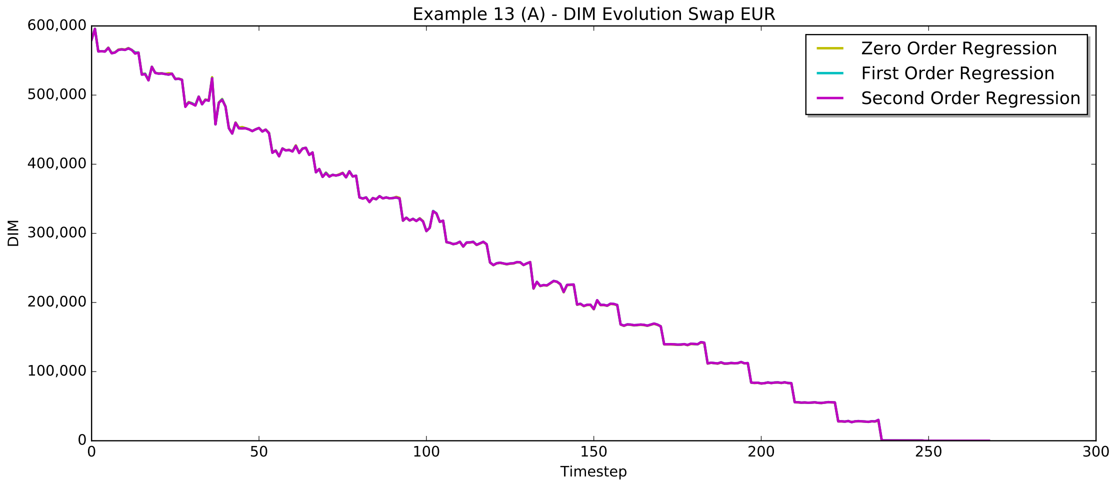

Figure 21: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swap of Example 13 A. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order (first and second order curves are not distinguishable here). The simulation uses 1000 samples and a time grid with bi-weekly steps in line with the Margin Period of Risk.

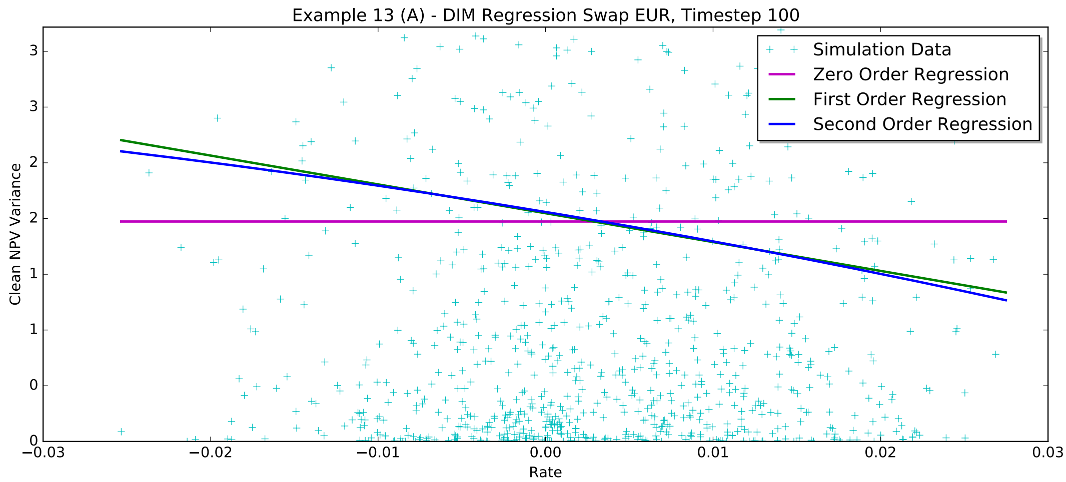

Figure 22: Regression snapshot at time step 100 for the EUR Swap of Example 13 A.

5.14 Minimal Market Data Setup

The example in folder `Examples/Example_14` demonstrates using a minimal market data setup in order to rerun the vanilla Swap exposure simulation shown in `Examples/Example_1`.

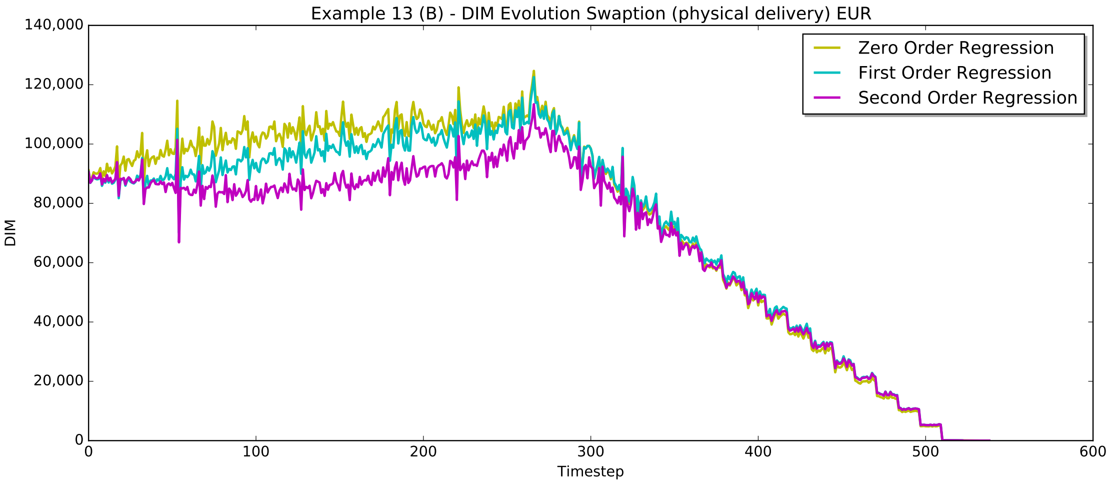

Figure 23: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swaption of Example 13 B with expiry in 10Y around time step 100. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order. The simulation uses 1000 samples and a time grid with bi-weekly steps in line with the Margin Period of Risk.

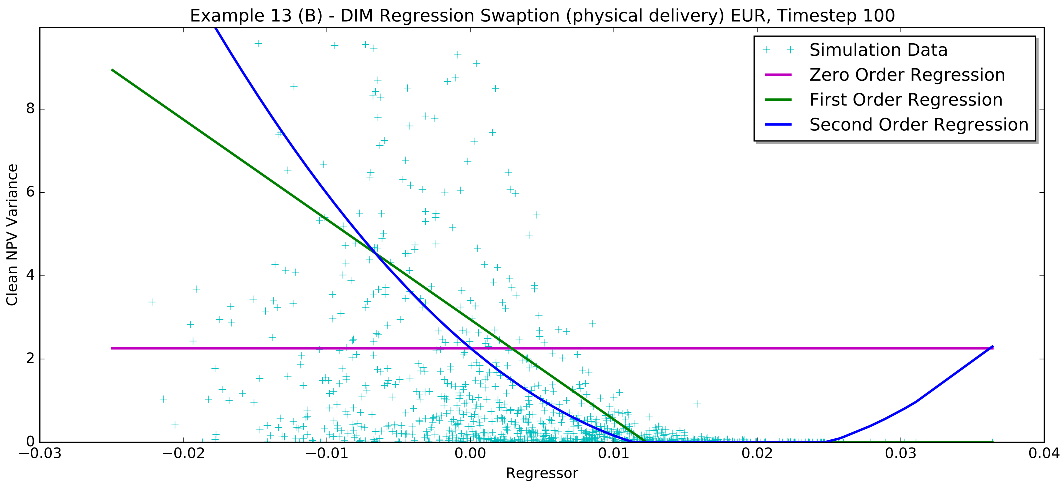

Figure 24: Regression snapshot at time step 100 (before expiry) for the EUR Swaption of Example 13 B.

The minimal market data uses single points per curve where possible.

5.15 Sensitivity Analysis, Stress Testing and Parametric Value-at-Risk

The example in folder `Examples/Example_15` demonstrates the calculation of sensitivities and stress scenarios. The portfolio used in this example consists of

- a vanilla swap in EUR

- a cross currency swap EUR-USD
- a resettable cross currency swap EUR-USD
- a FX forward EUR-USD
- a FX call option on USD/GBP
- a FX put option on USD/EUR
- an European swaption
- a Bermudan swaption
- a cap and a floor in USD
- a cap and a floor in EUR
- a fixed rate bond
- a floating rate bond with floor
- an Equity call option, put option and forward on S&P500
- an Equity call option, put option and forward on Lufthansa
- a CPI Swap referencing UKRPI
- a Year-on-Year inflation swap referencing EUHICPXT
- a USD CDS.

The sensitivity configuration in `sensitivity.xml` aims at computing the following sensitivities

- discount curve sensitivities in EUR, USD; GBP, CHF, JPY, on pillars 6M, 1Y, 2Y, 3Y, 5Y, 7Y, 10Y, 15Y, 20Y (absolute shift of 0.0001)
- forward curve sensitivities for EUR-EURIBOR 6M and 3M indices, EUR-EONIA, USD-LIBOR 3M and 6M, GBP-LIBOR 3M and 6M, CHF-LIBOR-6M and JPY-LIBOR-6M indices (absolute shift of 0.0001)
- yield curve shifts for a bond benchmark curve in EUR (absolute shift of 0.0001)
- FX spot sensitivities for USD, GBP, CHF, JPY against EUR as the base currency (relative shift of 0.01)
- FX vegas for USDEUR, GBPEUR, JPYEUR volatility surfaces (relative shift of 0.01)
- swaption vegas for the EUR surface on expiries 1Y, 5Y, 7Y, 10Y and underlying terms 1Y, 5Y, 10Y (relative shift of 0.01)
- caplet vegas for EUR and USD on an expiry grid 1Y, 2Y, 3Y, 5Y, 7Y, 10Y and strikes 0.01, 0.02, 0.03, 0.04, 0.05. (absolute shift of 0.0001)

- credit curve sensitivities on tenors 6M, 1Y, 2Y, 5Y, 10Y (absolute shift of 0.0001).
- Equity spots for S&P500 and Lufthansa
- Equity vegas for S&P500 and Lufthansa at expiries 6M, 1Y, 2Y, 3Y, 5Y
- Zero inflation curve deltas for UKRPI and EUHICPXT at tenors 6M, 1Y, 2Y, 3Y, 5Y, 7Y, 10Y, 15Y, 20Y
- Year on year inflation curve deltas for EUHICPXT at tenors 6M, 1Y, 2Y, 3Y, 5Y, 7Y, 10Y, 15Y, 20Y

Furthermore, mixed second order derivatives ("cross gammas") are computed for discount-discount, discount-forward and forward-forward curves in EUR.
By definition the sensitivities are zero rate sensitivities and optionlet sensitivities, no par sensitivities are provided. The sensitivity analysis produces three output files.
The first, `scenario.csv`, contains the shift direction (`UP`, `DOWN`, `CROSS`), the base NPV, the scenario NPV and the difference of these two for each trade and sensitivity key. For an overview over the possible scenario keys see 7.5.
The second file, `sensitivity.csv`, contains the shift size (in absolute terms always) and first ("Delta") and second ("Gamma") order finite differences computed from the scenario results. Note that the Delta and Gamma results pure differences, i.e. they are not divided by the shift size.
The third file, `crossgamma.csv` contains second order mixed differences according to the specified cross gamma filter, along with the shift sizes for the two factors involved. Again the reported result is not divided by the shift sizes.
The stress scenario definition in `stresstest.xml` defines two stress tests:

- `parallel_rates`: Rates are shifted in parallel by 0.01 (absolute). The EUR bond benchmark curve is shifted by increasing amounts 0.001, ..., 0.009 on the pillars 6M, ..., 20Y. FX Spots are shifted by 0.01 (relative), FX vols by 0.1 (relative), swaption and cap floor vols by 0.0010 (absolute). Credit curves are not yet shifted.
- `twist`: The EUR bond benchmark curve is shifted by amounts -0.0050, -0.0040, -0.0030, -0.0020, 0.0020, 0.0040, 0.0060, 0.0080, 0.0100 on pillars 6M, 1Y, 2Y, 3Y, 5Y, 7Y, 10Y, 15Y, 20Y.

The corresponding output file `stresstest.csv` contains the base NPV, the NPV under the scenario shifts and the difference of the two for each trade and scenario label.

Finally, this example demonstrates a parametric VaR calculation based on the sensitivity and cross gamma output from the sensitivity analysis (deltas, vegas, gammas, cross gammas) and an external covariance matrix input. The result in `var.csv` shows a breakdown by portfolio, risk class (All, Interest Rate, FX, Inflation, Equity, Credit) and risk type (All, Delta & Gamma, Vega). The results shown are Delta Gamma Normal VaRs for the 95% and 99% quantile, the holding period is incorporated into the input covariances. Alternatively, one can choose a Monte Carlo VaR which means that the sensitivity based P&L distribution is evaluated with MC simulation assuming normal respectively log-normal risk factor distribution.

5.16 Equity Derivatives Exposure

The example in folder `Examples/Example_16` demonstrates the computation of NPV, sensitivities, exposures and XVA for a portfolio of OTC equity derivatives. The portfolio used in this example consists of:

- an equity call option denominated in EUR ("Luft")
- an equity put option denominated in EUR ("Luft")
- an equity forward denominated in EUR ("Luft")
- an equity call option denominated in USD ("SP5")
- an equity put option denominated in USD ("SP5")
- an equity forward denominated in USD ("SP5")
- an equity Swap in USD with return type "price" ("SP5")
- an equity Swap in USD with return type "total" ("SP5")

The step-by-step procedure for running ORE is identical for equities as for other asset classes; the same market and portfolio data files are used to store the equity market data and trade details, respectively. For the exposure simulation, the calibration parameters for the equity risk factors can be set in the usual `simulation.xml` file.
Looking at the MtM results in the output file `npv.csv` we observe that put-call parity ($V_{Fwd} = V_{Call} - V_{Put}$) is observed as expected. Looking at Figure 25 we observe that the Expected Exposure profile of the equity call option trade is relatively smooth over time, while for the equity forward trade the Expected Exposure tends to increase as we approach maturity. This behaviour is similar to what we observe in sections 5.7 and 5.7.

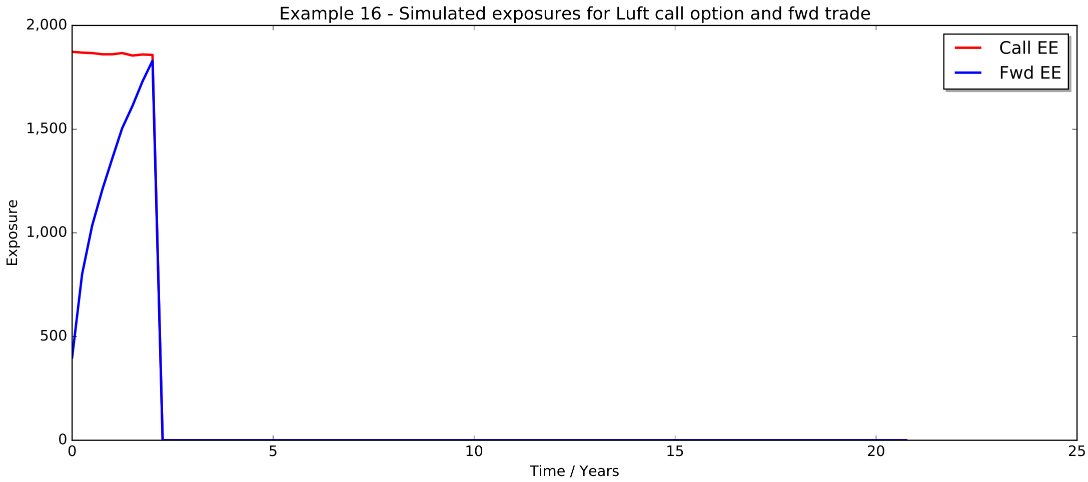

Figure 25: Equity ("Luft") call option and OTC forward exposure evolution, maturity in approximately 2.5 years. Simulation with 10000 paths and quarterly time steps.

5.17 Inflation Swap Exposures

The example portfolio in folder `Examples/Example_17` contains two CPI Swaps and one Year-on-Year Inflation Swap. The terms of the three trades are as follows:

- CPI Swap 1: Exchanges on 2036-02-05 a fixed amount of 20m GBP for a 10m GBP notional inflated with UKRPI with base CPI 210
- CPI Swap 2: Notional 10m GBP, maturity 2021-07-18, exchanging GBP Libor for GBP Libor 6M vs. 2% x CPI-Factor (Act/Act), inflated with index UKRPI with base CPI 210
- YOY Swap: Notional 10m EUR, maturity 2021-02-05, exchanging fixed coupons for EUHICPXT year-on-year inflation coupons
- YOY Swap with capped/floored YOY leg: Notional 10m EUR, maturity 2021-02-05, exchanging fixed coupons for EUHICPXT year-on-year inflation coupons, YOY leg capped with 0.03 and floored with 0.005
- YOY Swap with scheduled capped/floored YOY leg: Notional 10m EUR, maturity 2021-02-05, exchanging fixed coupons for EUHICPXT year-on-year inflation coupons, YOY leg capped with cap schedule and floored with floor schedule

The example generates cash flows, NPVs, exposure evolutions, XVAs, as well as two exposure graphs for CPI Swap 1 respectively the YOY Swap. For the YOY Swap and the both YOY Swaps with capped/floored YOY leg, the example generates their cash flows, NPVs, exposure evolutions, XVAs and sensitivities. Figure 26 shows the CPI Swap exposure evolution.

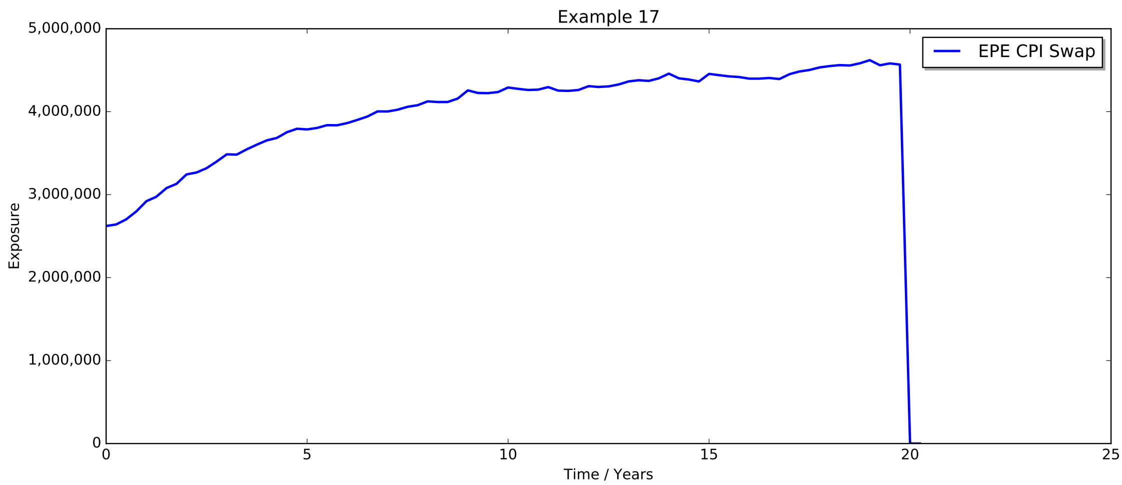

Figure 26: CPI Swap 1 exposure evolution. Simulation with 1000 paths and quarterly time steps.

Figure 27 shows the evolution of the 5Y maturity Year-on-Year inflation swap for comparison. Note that the inflation simulation model (Dodgson-Kainth, see appendix A.1) yields the evolution of inflation indices and inflation zero bonds which allows spanning future inflation zero curves and the pricing of CPI swaps. To price Year-on-Year inflation Swaps under future scenarios, we imply Year-on-Year inflation curves from zero inflation curves[3]. Note that for pricing Year-on-Year Swaps as of today we use a separate inflation curve bootstrapped from quoted Year-on-Year inflation Swaps.

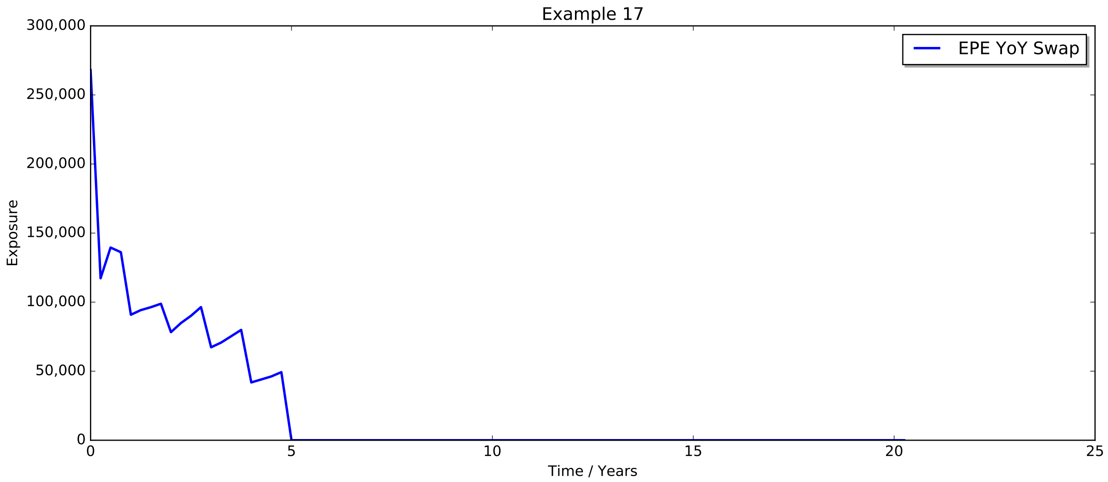

Figure 27: Year-on-Year Inflation Swap exposure evolution. Simulation with 1000 paths and quarterly time steps.

5.18 Bonds and Amortisation Structures

The example in folder `Examples/Example_18` computes NPVs and cash flow projections for a vanilla bond portfolio consisting of a range of bond products, in particular demonstrating amortisation features:

- fixed rate bond
- floating rate bond linked to Euribor 6M
- bond switching from fixed to floating
- bond with 'fixed amount' amortisation
- bond with percentage amortisation relative to the initial notional
- bond with percentage amortisation relative to the previous notional
- bond with fixed annuity amortisation
- bond with floating annuity amortisation (this example needs QuantLib 1.10 or higher to work, in particular the amount() method in the Coupon class needs to be virtual)
- bond with fixed amount amortisation followed by percentage amortisation relative to previous notional

[3]Currently we discard the required (small) convexity adjustment. This will be supplemented in a subsequent release.

After running the example, the results of the computation can be found in the output files `npv.csv` and `flows.csv`, respectively.

Note that the amortisation features used here are linked to the LegData structure, hence not limited to the Bond instrument, see section 8.3.6.

5.19 Swaption Pricing with Smile

This example in folder `Examples/Example_19` demonstrates European Swaption pricing with and without smile. Calling

```
python run.py
```

will launch two ORE runs using config files `ore_flat.xml` and `ore_smile.xml`, respectively. The only difference in these is referencing alternative market configurations `todaymarket_flat.xml` and `todaysmarket_smile.xml` using an ATM Swaption volatility matrix and a Swaption cube, respectively. NPV results are written to `npv_flat.cvs` and `npv_smile.csv`.

5.20 Credit Default Swap Pricing

This example in folder `Examples/Example_20` demonstrates Credit Default Swap pricing via ORE. Calling

```
python run.py
```

will launch a single ORE run to process a single name CDS example and to generate NPV and cash flows in the usual result files.

CDS can be included in sensitivity analysis and stress testing. Exposure simulation for credit derivatives will follow in the next ORE release.

5.21 CMS and CMS Cap/Floor Pricing

This example in folder `Examples/Example_21` demonstrates the pricing of CMS and CMS Cap/Floor using a portfolio consisting of a CMS Swap (CMS leg vs. fixed leg) and a CMS Cap. Calling

```
python run.py
```

will launch a single ORE run to process the portfolio and generate NPV and cash flows in the usual result files.

CMS structures can be included in sensitivity analysis, stress testing and exposure simulation.

5.22 Option Sensitivity Analysis with Smile

The example in folder `Examples/Example_22` demonstrates the current state of sensitivity calculation for European options where the volatility surface has a smile.

The portfolio used in this example consists of

- an equity call option denominated in USD ("SP5")
- an equity put option denominated in USD ("SP5")

- a receiver swaption in EUR
- an FX call option on EUR/USD

Refer to appendix A.13 for the current status of sensitivity implementation with smile. In this example the setup is as follows

- today's market is configured with volatility smile for all three products above
- simulation market has two configurations, to simulate "ATM only" or the "full surface"; "ATM only" means that only ATM volatilities are to be simulated and shifts to ATM vols are propagated to the respective smile section (see appendix A.13);
- the sensitivity analysis has two corresponding configurations as well, "ATM only" and "full surface"; note that the "full surface" configuration leads to explicit sensitivities by strike only in the case of Swaption volatilities, for FX and Equity volatilities only ATM sensitivity can be specified at the moment and sensitivity output is currently aggregated to the ATM bucket (to be extended in subsequent releases).

The respective output files end with "`_fullSurface.csv`" respectively "`_atmOnly.csv`".

5.23 FRA and Average OIS Exposure

This example in folder `Examples/Example_23` demonstrates pricing, cash flow projection and exposure simulation for two additional products

- Forward Rate Agreements
- Averaging Overnight Index Swaps

using a minimal portfolio of four trades, one FRA and three OIS. The essential results are in `npv.csv`, `flows.csv` and four `exposure_trade_*.csv` files.

5.24 Commodity Forward and Option

The example in folder `Examples/Example_24` demonstrates pricing and sensitivity analysis for

- Commodity Forwards
- European Commodity Options

using a minimal portfolio of four forwards and two options referencing WTI and Gold. The essential results are in `npv.csv` and `sensitivity.csv`.

5.25 CMS Spread with (Digital) Cap/Floor

The example in folder `Examples/Example_25` demonstrates pricing, sensitivity analysis and exposure simulation for

- Capped/Floored CMS Spreads
- CMS Spreads with Digital Caps/Floors

The example can be run with

```
python run.py
```

and results are in `npv.csv`, `sensitivity.csv`, `exposure_*.csv` and the exposure graphs in `mpl_cmsspread.pdf`.

5.26 Bootstrap Consistency

The example in folder `Examples/Example_26` confirms that bootstrapped curves correctly reprice the bootstrap instruments (FRAs, Interest Rate Swaps, FX Forwards, Cross Currency Basis Swaps) using three pricing setups with

- EUR collateral discounting (configuration xois_eur)
- USD collateral discounting (configuration xois_usd)
- in-currency OIS discounting (configuration collateral_inccy)

all defined in `Examples/Input/todaysmarket.xml`.

The required portfolio files need to be generated from market data and conventions in `Examples/Input` and trade templates in `Examples/Example_26/Helpers`, calling

```
python TradeGenerator.py
```

This will place three portfolio files `*_portfolio.xml` in the input folder. Thereafter, the three consistency checks can be run calling

```
python run.py
```

Results are in three files `*_npv.csv` and should show zero NPVs for all benchmark instruments.

5.27 BMA Basis Swap

The example in folder `Examples/Example_27` demonstrates pricing and sensitivity analysis for a series of USD Libor 3M vs. Averaged BMA (SIFMA) Swaps that correspond to the instruments used to bootstrap the BMA curve.
The example can be run with

```
python run.py
```

and results are in `npv.csv` and `sensitivity.csv`.

5.28 Discount Ratio Curves

The example in folder `Examples/Example_28` shows how to use a yield curve built from a DiscountRatio segment. In particular, it builds a GBP collateralized in EUR discount curve by referencing three other discount curves:

- a GBP collateralised in USD curve
- a EUR collateralised in USD curve
- a EUR OIS curve i.e. a EUR collateralised in EUR curve

The implicit assumption in building the curve this way is that EUR/GBP FX forwards collateralised in EUR have the same fair market rate as EUR/GBP FX forwards collateralised in USD. This assumption is illustrated in the example by the NPV of the two forward instruments in the portfolio returning exactly 0 under both discounting regimes i.e. under USD collateralization with direct curve building and under EUR collateralization with the discount ratio modified "GBP-IN-EUR" curve.
Also, in this example, an assumption is made that there are no direct GBP/EUR FX forward or cross currency quotes available which in general is false. The example s merely for illustration.
Both collateralizaton scenarios can be run calling `python run.py`.

5.29 Curve Building using Fixed vs. Float Cross Currency Helpers

The example in folder `Examples/Example_29` demonstrates using fixed vs. float cross currency swap helpers. In particular, it builds a TRY collateralised in USD discount curve using TRY annual fixed vs USD 3M Libor swap quotes.
The portfolio contains an at-market fixed vs. float cross currency swap that is included in the curve building. The NPV of this swap should be zero when the example is run, using `python run.py` or "directly" calling `ore[.exe] ore.xml`.

6 Launchers and Visualisation

6.1 Jupyter

ORE comes with an experimental Jupyter notebook for launching ORE batches and in particular for drilling into NPV cube data. The notebook is located in directory `FrontEnd/Python/Visualization/npvcube`. To launch the notebook, change to this directory and follow instructions in the `Readme.txt`. In a nutshell, type[4]

```
jupyter notebook
```

to start the ipyton console and open a browser window. From the list of files displayed in the browser then click

```
ore_jupyter_dashboard.ipynb
```

to open the ORE notebook. The notebook offers

- launching an ORE job
- selecting an NPV cube file and netting sets or trades therein
- plotting a 3d exposure probability density surface
- viewing exposure probability density function at a selected future time
- viewing expected exposure evolution through time

The cube file loaded here by default when processing all cells of the notebook (without changing it or launching a ORE batch) is taken from `Example_7` (FX Forwards and FX Options).

6.2 Calc

ORE comes with a simple LibreOffice Calc [10] sheet as an ORE launcher and basic result viewer. This is demonstrated on the example in section 5.1. It is currently based on the stable LibreOffice version 5.0.6 and tested on OS X.
To launch Calc, open a terminal, change to directory `Examples/Example_1`, and run

```
./launchCalc.sh
```

The user can choose a configuration (one of the `ore*.xml` files in Example_1's subfolder Input) by hitting the 'Select' button. Initially Input/ore.xml is pre-selected. The ORE process is then kicked off by hitting 'Run'. Once completed, standard ORE reports (NPV, Cashflow, XVA) are loaded into several sheets. Moreover, exposure evolutions can then be viewed by hitting 'View' which shows the result in figure 28.
This demo uses simple Libre Office Basic macros which call Python scripts to execute ORE. The Libre Office Python uno module (which comes with Libre Office) is used to communicate between Python and Calc to upload results into the sheets.

[4]With Mac OS X, you may need to set the environment variable `LANG` to `en_US.UTF-8` before running jupyter, as mentioned in the installation section 4.3.

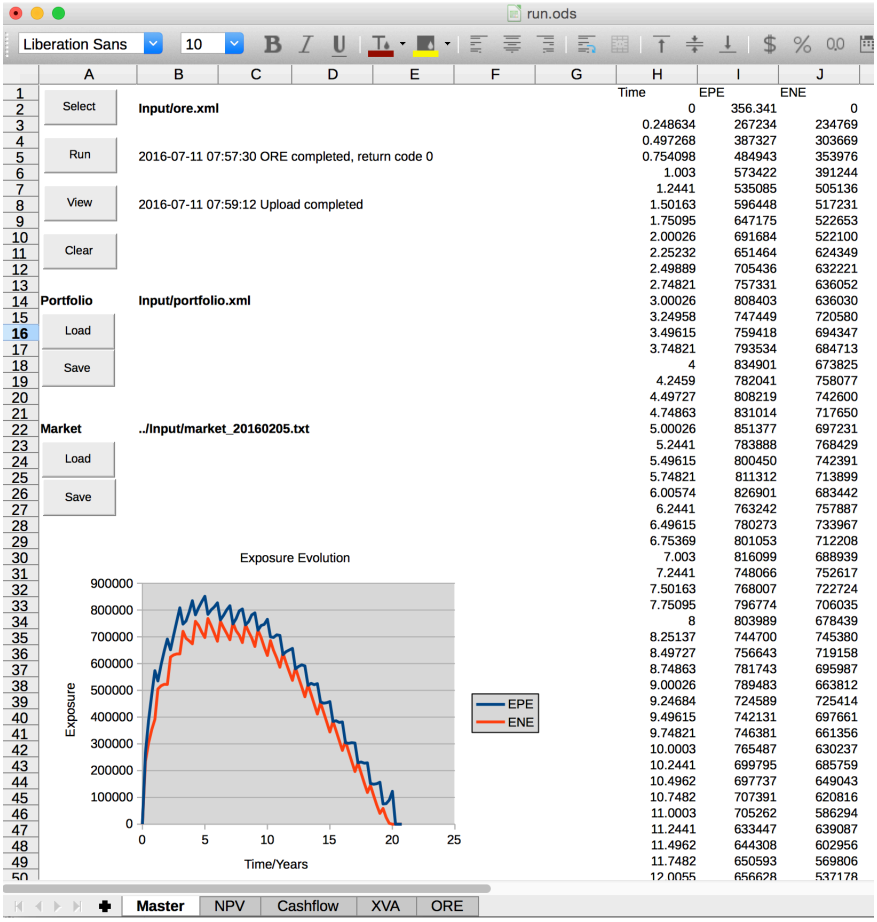

Figure 28: Calc sheet after hitting 'Run'.

6.3 Excel

ORE also comes with a basic Excel sheet to demonstrate launching ORE and presenting results in Excel. This demo is more self-contained than the Calc demo in the previous section, as it uses VBA only rather than calls to external Python scripts. The Excel demo is available in Example_1. Launch `Example_1.xlsm`. Then modify the paths on the first sheet, and kick off the ORE process.

7 Parametrisation

A run of ORE is kicked off with a single command line parameter

```
ore[.exe] ore.xml
```

which points to the 'master input file' referred to as `ore.xml` subsequently. This file is the starting point of the engine's configuration explained in the following sub section. An overview of all input configuration files respectively all output files is shown in Table 3 respectively Table 4. To set up your own ORE configuration, it might be not be necessary to start from scratch, but instead use any of the examples discussed in section 5 as a boilerplate and just change the folders, see section 7.1, and the trade data, see section 8, together with the netting definitions, see section 9.

7.1 Master Input File: `ore.xml`

The master input file contains general setup information (paths to configuration, trade data and market data), as well as the selection and configuration of analytics. The file has an opening and closing root element `<ORE>`, `</ORE>` with three sections

- Setup
- Markets
- Analytics

which we will explain in the following.

7.1.1 Setup

This subset of data is easiest explained using an example, see listing 1.

Listing 1: ORE setup example

```xml
<Setup>
  <Parameter name="asofDate">2016-02-05</Parameter>
  <Parameter name="inputPath">Input</Parameter>
  <Parameter name="outputPath">Output</Parameter>
  <Parameter name="logFile">log.txt</Parameter>
  <Parameter name="logMask">255</Parameter>
  <Parameter name="marketDataFile">../../Input/market_20160205.txt</Parameter>
  <Parameter name="fixingDataFile">../../Input/fixings_20160205.txt</Parameter>
  <Parameter name="implyTodaysFixings">Y</Parameter>
  <Parameter name="curveConfigFile">../../Input/curveconfig.xml</Parameter>
  <Parameter name="conventionsFile">../../Input/conventions.xml</Parameter>
  <Parameter name="marketConfigFile">../../Input/todaysmarket.xml</Parameter>
  <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
  <Parameter name="portfolioFile">portfolio.xml</Parameter>
  <!-- None, Unregister, Defer or Disable -->
  <Parameter name="observationModel">Disable</Parameter>
</Setup>
```

Parameter names are self explanatory: Input and output path are interpreted relative from the directory where the ORE executable is executed, but can also be specified using absolute paths. All file names are then interpreted relative to the 'inputPath' and 'outputPath', respectively. The files starting with `../../Input/` then point to files in the global Example input directory `Example/Input/*`, whereas files such as `portfolio.xml` are local inputs in `Example/Example_#/Input/`.
Parameter `logMask` determines the verbosity of log file output. Log messages are internally labelled as Alert, Critical, Error, Warning, Notice, Debug, associated with logMask values 1, 2, 4, 8, ..., 64. The logMask allows filtering subsets of these categories and controlling the verbosity of log file output[5]. LogMask 255 ensures maximum verbosity.
When ORE starts, it will initialise today's market, i.e. load market data and fixings, and build all term structures as specified in `todaysmarket.xml`. Moreover, ORE will load the trades in `portfolio.xml` and link them with pricing engines as specified in `pricingengine.xml`. When parameter `implyTodaysFixings` is set to Y, today's fixings

[5]by bitwise comparison of the the external logMask value with each message's log level

would not be loaded but implied, relevant when pricing/bootstrapping off hypothetical market data as e.g. in scenario analysis and stress testing.

The last parameter `observationModel` can be used to control ORE performance during simulation. The choices *Disable* and *Unregister* yield similarly improved performance relative to choice *None*. For users familiar with the QuantLib design - the parameter controls to which extent *QuantLib observer notifications* are used when market and fixing data is updated and the evaluation date is shifted:

- The 'Unregister' option limits the amount of notifications by unregistering floating rate coupons from indices;
- Option 'Defer' disables all notifications during market data and fixing updates with `ObservableSettings::instance().disableUpdates(true)` and kicks off updates afterwards when enabled again
- The 'Disable' option goes one step further and disables all notifications during market data and fixing updates, and in particular when the evaluation date is changed along a path, with
`ObservableSettings::instance().disableUpdates(false)`
Updates are not deferred here. Required term structure and instrument recalculations are triggered explicitly.

7.1.2 Markets

The `Markets` section (see listing 2) is used to choose market configurations for calibrating the IR, FX and EQ simulation model components, pricing and simulation, respectively. These configurations have to be defined in `todaysmarket.xml` (see section 7.2).

Listing 2: ORE markets

```
<Markets>
  <Parameter name="lgmcalibration">collateral_inccy</Parameter>
  <Parameter name="fxcalibration">collateral_eur</Parameter>
  <Parameter name="eqcalibration">collateral_inccy</Parameter>
  <Parameter name="pricing">collateral_eur</Parameter>
  <Parameter name="simulation">collateral_eur</Parameter>
</Markets>
```

For example, the calibration of the simulation model's interest rate components requires local OIS discounting whereas the simulation phase requires cross currency adjusted discount curves to get FX product pricing right. So far, the market configurations are used only to distinguish discount curve sets, but the market configuration concept in ORE applies to all term structure types.

7.1.3 Analytics

The `Analytics` section lists all permissible analytics using tags `<Analytic type="...">` `... </Analytic>` where type can be (so far) in

- npv
- cashflow

- curves
- simulation
- xva
- sensitivity
- stress

Each `Analytic` section contains a list of key/value pairs to parameterise the analysis of the form `<Parameter name="key">value</Parameter>`. Each analysis must have one key `active` set to Y or N to activate/deactivate this analysis. The following listing 3 shows the parametrisation of the first three basic analytics in the list above.

Listing 3: ORE analytics: npv, cashflow and curves

```
<Analytics>
  <Analytic type="npv">
    <Parameter name="active">Y</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="outputFileName">npv.csv</Parameter>
  </Analytic>
  <Analytic type="cashflow">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">flows.csv</Parameter>
  </Analytic>
  <Analytic type="curves">
    <Parameter name="active">Y</Parameter>
    <Parameter name="configuration">default</Parameter>
    <Parameter name="grid">240,1M</Parameter>
    <Parameter name="outputFileName">curves.csv</Parameter>
  </Analytic>
  <Analytic type="...">
    <!-- ... -->
  </Analytic>
</Analytics>
```

The cashflow analytic writes a report containing all future cashflows of the portfolio. Table 5 shows a typical output for a vanilla swap.

#ID	Type	LegNo	PayDate	Amount	Currency	Coupon	Accrual	fixingDate	fixingValue	Notional
tr123	Swap	0	13/03/17	-111273.76	EUR	-0.00201	0.50556	08/09/16	-0.00201	100000000.00
tr123	Swap	0	12/09/17	-120931.71	EUR	-0.002379	0.50833	09/03/17	-0.002381	100000000.00
...										

Table 5: Cashflow Report

The amount column contains the projected amount including embedded caps and floors and convexity (if applicable), the coupon column displays the corresponding rate estimation. The fixing value on the other hand is the plain fixing projection as given by the forward value, i.e. without embedded caps and floors or convexity.
Note that the fixing value might deviate from the coupon value even for a vanilla coupon, if the QuantLib library was compiled *without* the flag `QL_USE_INDEXED_COUPON` (which

is the default case). In this case the coupon value uses a par approximation for the forward rate assuming the index estimation period to be identical to the accrual period, while the fixing value is the actual forward rate for the index estimation period, i.e. whenever the index estimation period differs from the accrual period the values will be slightly different.
The Notional column contains the underlying notional used to compute the amount of each coupon. It contains `#NA` if a payment is not a coupon payment.
The curves analytic exports all yield curves that have been built according to the specification in `todaysmarket.xml`. Key `configuration` selects the curve set to be used (see explanation in the previous Markets section). Key `grid` defines the time grid on which the yield curves are evaluated, in the example above a grid of 240 monthly time steps from today. The discount factors for all curves with configuration default will be exported on this monthly grid into the csv file specified by key `outputFileName`. The grid can also be specified explicitly by a comma separated list of tenor points such as `1W, 1M, 2M, 3M, ...`.

The purpose of the `simulation` 'analytics' is to run a Monte Carlo simulation which evolves the market as specified in the simulation config file. The primary result is an NPV cube file, i.e. valuations of all trades in the portfolio file (see section Setup), for all future points in time on the simulation grid and for all paths. Apart from the NPV cube, additional scenario data (such as simulated overnight rates etc) are stored in this process which are needed for subsequent XVA analytics.

Listing 4: ORE analytic: simulation

```
<Analytics>
  <Analytic type="simulation">
    <Parameter name="active">Y</Parameter>
    <Parameter name="simulationConfigFile">simulation.xml</Parameter>
    <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="storeFlows">Y</Parameter>
    <Parameter name="cubeFile">cube_A.dat</Parameter>
    <Parameter name="aggregationScenarioDataFileName">scenariodata.dat</Parameter>
    <Parameter name="aggregationScenarioDump">scenariodump.csv</Parameter>
  </Analytic>
</Analytics>
```

The pricing engines file specifies how trades are priced under future scenarios which can differ from pricing as of today (specified in section Setup). Key base currency determines into which currency all NPVs will be converted. Key store scenarios (Y or N) determines whether the market scenarios are written to a file for later reuse. And finally, the key 'store flows' (Y or N) controls whether cumulative cash flows between simulation dates are stored in the (hyper-) cube for post processing in the context of Dynamic Initial Margin and Variation Margin calculations. The additional scenario data (written to the specified file here) is likewise required in the post processor step. These data comprise simulated index fixing e.g. for collateral compounding and simulated FX rates for cash collateral conversion into base currency. The scenario dump file, if specified here, causes ORE to write simulated market data to a human-readable csv file. Only those currencies or indices are written here that are stated in the AggregationScenarioDataCurrencies and AggregationScenarioDataIndices subsections of the simulation files market section, see also section 7.4.3.

The XVA analytic section offers CVA, DVA, FVA and COLVA calculations which can be selected/deselected here individually. All XVA calculations depend on a previously generated NPV cube (see above) which is referenced here via the `cubeFile` parameter. This means one can re-run the XVA analytics without regenerating the cube each time. The XVA reports depend in particular on the settings in the `csaFile` which determines CSA details such as margining frequency, collateral thresholds, minimum transfer amounts, margin period of risk. By splitting the processing into pre-processing (cube generation) and post-processing (aggregation and XVA analysis) it is possible to vary these CSA details and analyse their impact on XVAs quickly without re-generating the NPV cube.

Listing 5: ORE analytic: xva

```
<Analytics>
  <Analytic type="xva">
    <Parameter name="active">Y</Parameter>
    <Parameter name="csaFile">netting.xml</Parameter>
    <Parameter name="cubeFile">cube.dat</Parameter>
    <Parameter name="hyperCube">Y</Parameter>
    <Parameter name="scenarioFile">scenariodata.dat</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="exposureProfiles">Y</Parameter>
    <Parameter name="exposureProfilesByTrade">Y</Parameter>
    <Parameter name="quantile">0.95</Parameter>
    <Parameter name="calculationType">Symmetric</Parameter>
    <Parameter name="allocationMethod">None</Parameter>
    <Parameter name="marginalAllocationLimit">1.0</Parameter>
    <Parameter name="exerciseNextBreak">N</Parameter>
    <Parameter name="cva">Y</Parameter>
    <Parameter name="dva">N</Parameter>
    <Parameter name="dvaName">BANK</Parameter>
    <Parameter name="fva">N</Parameter>
    <Parameter name="fvaBorrowingCurve">BANK_EUR_BORROW</Parameter>
    <Parameter name="fvaLendingCurve">BANK_EUR_LEND</Parameter>
    <Parameter name="colva">Y</Parameter>
    <Parameter name="collateralFloor">Y</Parameter>
    <Parameter name="kva">Y</Parameter>
    <Parameter name="kvaCapitalDiscountRate">0.10</Parameter>
    <Parameter name="kvaAlpha">1.4</Parameter>
    <Parameter name="kvaRegAdjustment">12.5</Parameter>
    <Parameter name="kvaCapitalHurdle">0.012</Parameter>
    <Parameter name="kvaOurPdFloor">0.03</Parameter>
    <Parameter name="kvaTheirPdFloor">0.03</Parameter>
    <Parameter name="kvaOurCvaRiskWeight">0.005</Parameter>
    <Parameter name="kvaTheirCvaRiskWeight">0.05</Parameter>
    <Parameter name="dim">Y</Parameter>
    <Parameter name="mva">Y</Parameter>
    <Parameter name="dimQuantile">0.99</Parameter>
    <Parameter name="dimHorizonCalendarDays">14</Parameter>
    <Parameter name="dimRegressionOrder">1</Parameter>
    <Parameter name="dimRegressors">EUR-EURIBOR-3M,USD-LIBOR-3M,USD</Parameter>
    <Parameter name="dimLocalRegressionEvaluations">100</Parameter>
    <Parameter name="dimLocalRegressionBandwidth">0.25</Parameter>
    <Parameter name="dimScaling">1.0</Parameter>
    <Parameter name="dimEvolutionFile">dim_evolution.txt</Parameter>
    <Parameter name="dimRegressionFiles">dim_regression.txt</Parameter>
    <Parameter name="dimOutputNettingSet">CPTY_A</Parameter>
    <Parameter name="dimOutputGridPoints">0</Parameter>
    <Parameter name="rawCubeOutputFile">rawcube.csv</Parameter>
    <Parameter name="netCubeOutputFile">netcube.csv</Parameter>
    <Parameter name="fullInitialCollateralisation">true</Parameter>
  </Analytic>
</Analytics>
```

Parameters:

- `csaFile:` Netting set definitions file covering CSA details such as margining frequency, thresholds, minimum transfer amounts, margin period of risk

- `cubeFile:` NPV cube file previously generated and to be post-processed here
- `hyperCube:` If set to N, the cube file is expected to have depth 1 (storing NPV data only), if set to Y it is expected to have depth > 1 (e.g. storing NPVs and cumulative flows)
- `scenarioFile:` Scenario data previously generated and used in the post-processor (simulated index fixings and FX rates)
- `baseCurrency:` Expression currency for all NPVs, value adjustments, exposures
- `exposureProfiles:` Flag to enable/disable exposure output for each netting set
- `exposureProfilesByTrade:` Flag to enable/disable stand-alone exposure output for each trade
- `quantile` Confidence level for Potential Future Exposure (PFE) reporting
- `calculationType` Determines the settlement of margin calls: Symmetric - margin for both counterparties settled after the margin period of risk; AsymmetricCVA - margin requested from the counterparty settles with delay, margin requested from us settles immediately; AsymmetricDVA - vice versa).
- `allocationMethod:` XVA allocation method, choices are *None, Marginal, RelativeXVA*
- `marginalAllocationLimit:` The marginal allocation method a la Pykhtin/Rosen breaks down when the netting set value vanishes while the exposure does not. This parameter acts as a cutoff for the marginal allocation when the absolute netting set value falls below this limit and switches to equal distribution of the exposure in this case.
- `exerciseNextBreak:` Flag to terminate all trades at their next break date before aggregation and the subsequent analytics
- `cva, dva, fva, colva, collateralFloor, dim, mva:` Flags to enable/disable these analytics.
- `dvaName:` Credit name to look up the own default probability curve and recovery rate for DVA calculation
- `fvaBorrowingCurve:` Identifier of the borrowing yield curve
- `fvaLendingCurve:` Identifier of the lending yield curve
- `kva:` Flag to enable setting the kva ccr parameters.
- `kvaCapitalDiscountRate, kvaAlpha, kvaRegAdjustment, kvaCapitalHurdle, kvaOurPdFloor, kvaTheirPdFloor kvaOurCvaRiskWeight, kvaTheirCvaRiskWeight:` the kva CCR parameters (see A.9 and A.10.
- `dimQuantile:` Quantile for Dynamic Initial Margin (DIM) calculation
- `dimHorizonCalendarDays:` Horizon for DIM calculation, 14 calendar days for 2 weeks, etc.

- `dimRegressionOrder`: Order of the regression polynomial (netting set clean NPV move over the simulation period versus netting set NPV at period start)
- `dimRegressors`: Variables used as regressors in a single- or multi-dimensional regression; these variable names need to match entries in the `simulation.xml`'s AggregationScenarioDataCurrencies and AggregationScenarioDataIndices sections (only these scenario data are passed on to the post processor); if the list is empty, the NPV will be used as a single regressor
- `dimLocalRegressionEvaluations`: Nadaraya-Watson local regression evaluated at the given number of points to validate polynomial regression. Note that Nadaraya-Watson needs a large number of samples for meaningful results. Evaluating the local regression at many points (samples) has a significant performance impact, hence the option here to limit the number of evaluations.
- `dimLocalRegressionBandwidth`: Nadaraya-Watson local regression bandwidth in standard deviations of the independent variable (NPV)
- `dimScaling`: Scaling factor applied to all DIM values used, e.g. to reconcile simulated DIM with actual IM at t_0
- `dimEvolutionFile`: Output file name to store the evolution of zero order DIM and average of nth order DIM through time
- `dimRegressionFiles`: Output file name(s) for a DIM regression snapshot, comma separated list
- `dimOutputNettingSet`: Netting set for the DIM regression snapshot
- `dimOutputGridPoints`: Grid point(s) (in time) for the DIM regression snapshot, comma separated list
- `rawCubeOutputFile`: File name for the trade NPV cube in human readable csv file format (per trade, date, sample)
- `netCubeOutputFile`: File name for the aggregated NPV cube in human readable csv file format (per netting set, date, sample) *after* taking collateral into account
- `fullInitialCollateralisation`: If set to `true`, then for every netting set, the collateral balance at $t = 0$ will be set to the NPV of the setting set. The resulting effect is that EPE, ENE and PFE are all zero at $t = 0$. If set to `false` (default value), then the collateral balance at $t = 0$ will be set to zero.

The two cube file outputs `rawCubeOutputFile` and `netCubeOutputFile` are provided for interactive analysis and visualisation purposes, see section 6.

The `sensitivity` and `stress` 'analytics' provide computation of bump and revalue (zero rate resp. optionlet) sensitivities and NPV changes under user defined stress scenarios. Listing 6 shows a typical configuration for sensitvity calculation.

Listing 6: ORE analytic: sensitivity

```xml
<Analytics>
 <Analytic type="sensitivity">
   <Parameter name="active">Y</Parameter>
   <Parameter name="marketConfigFile">simulation.xml</Parameter>
   <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
   <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
   <Parameter name="scenarioOutputFile">scenario.csv</Parameter>
   <Parameter name="sensitivityOutputFile">sensitivity.csv</Parameter>
   <Parameter name="crossGammaOutputFile">crossgamma.csv</Parameter>
   <Parameter name="outputSensitivityThreshold">0.000001</Parameter>
   <Parameter name="recalibrateModels">Y</Parameter>
 </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `marketConfigFile:` Configuration file defining the simulation market under which sensitivities are computed, see 7.4. Only a subset of the specification is needed (the one given under `Market`, see 7.4.3 for a detailed description).
- `sensitivityConfigFile:` Configuration file for the sensitivity calculation, see section 7.5.
- `pricingEnginesFile:` Configuration file for the pricing engines to be used for sensitivity calculation.
- `scenarioOutputFile:` File containing the results of the sensitivity calculation in terms of the base scenario NPV, the scenario NPV and their difference.
- `sensitivityOutputFile:` File containing the results of the sensitivity calculation in terms of the base scenario NPV, the shift size and the resulting first and (pure) second order finite differences
- `crossGammaOutputFile:` File containing the results of the sensitivity calculation in terms of the base scenario NPV, two shift sizes and a (mixed) second order finite difference associated to a cross gamma calculation
- `outputSensitivityThreshold:` Only finite differences with absolute value greater than this number are written to the output files.
- `recalibrateModels:` If set to Y, then recalibrate pricing models after each shift of relevant term structures; otherwise do not recalibrate

The stress analytics configuration is similar to the one of the sensitivity calculation. Listing 7 shows an example.

Listing 7: ORE analytic: stress

```
<Analytics>
 <Analytic type="stress">
   <Parameter name="active">Y</Parameter>
   <Parameter name="marketConfigFile">simulation.xml</Parameter>
   <Parameter name="stressConfigFile">stresstest.xml</Parameter>
   <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
   <Parameter name="scenarioOutputFile">stresstest.csv</Parameter>
   <Parameter name="outputThreshold">0.000001</Parameter>
 </Analytic>
</Analytics>
```

The parameters have the same interpretation as for the sensitivity analytic. The configuration file for the stress scenarios is described in more detail in section 7.6.

The `VaR` 'analytics' provide computation of Value-at-Risk measures based on the sensitivity (delta, gamma, cross gamma) data above. Listing 8 shows a configuration example.

Listing 8: ORE analytic: VaR

```
<Analytics>
    <Analytic type="parametricVar">
      <Parameter name="active">Y</Parameter>
      <Parameter name="portfolioFilter">PF1|PF2</Parameter>
      <Parameter name="sensitivityInputFile">
         ../Output/sensitivity.csv,../Output/crossgamma.csv
      </Parameter>
      <Parameter name="covarianceInputFile">covariance.csv</Parameter>
      <Parameter name="salvageCovarianceMatrix">N</Parameter>
      <Parameter name="quantiles">0.01,0.05,0.95,0.99</Parameter>
      <Parameter name="breakdown">Y</Parameter>
      <!-- Delta, DeltaGammaNormal, MonteCarlo -->
      <Parameter name="method">DeltaGammaNormal</Parameter>
      <Parameter name="mcSamples">100000</Parameter>
      <Parameter name="mcSeed">42</Parameter>
      <Parameter name="outputFile">var.csv</Parameter>
    </Analytic> </Analytics>
```

The parameters have the following interpretation:

- portfolioFilter: Regular expression used to filter the portfolio for which VaR is computed; if the filter is not provided, then the full portfolio is processed
- `sensitivityInputFile:` Reference to the sensitivity (deltas, vegas, gammas) and cross gamma input as generated by ORE in a comma separated list
- `covarianceFile:` Reference to the covariances input data; these are currently not calculated in ORE and need to be provided externally, in a blank/tab/comma separated file with three columns (factor1, factor2, covariance), where factor1 and factor2 follow the naming convention used in ORE's sensitivity and cross gamma output files. Covariances need to be consistent with the sensitivity data provided. For example, if sensitivity to factor1 is computed by absolute shifts and expressed in basis points, then the covariances with factor1 need to be based on absolute

basis point shifts of factor1; if sensitivity is due to a relative factor1 shift of 1%, then covariances with factor1 need to be based on relative shifts expressed in percentages to, etc. Also note that covariances are expected to include the desired holding period, i.e. no scaling with square root of time etc is performed in ORE;

- `salvageCovarianceMatrix`: If set to Y, turn the input covariance matrix into a valid (positive definite) matrix applying a Salvaging algorithm; if set to N, throw an exception if the matrix is not positive definite
- `quantiles`: Several desired quantiles can be specified here in a comma separated list; these lead to several columns of results in the output file, see below. Note that e.g. the 1% quantile corresponds to the lower tail of the P&L distribution (VaR), 99% to the upper tail.
- `breakdown`: If yes, VaR is computed by portfolio, risk class (All, Interest Rate, FX, Inflation, Equity, Credit) and risk type (All, Delta & Gamma, Vega)
- `method`: Choices are *Delta, DeltaGammaNormal, MonteCarlo*, see appendix A.14
- `mcSamples`: Number of Monte Carlo samples used when the *MonteCarlo* method is chosen
- `mcSeed`: Random number generator seed when the *MonteCarlo* method is chosen
- `outputFile`: Output file name

7.2 Market: `todaysmarket.xml`

This configuration file determines the subset of the 'market' universe which is going to be built by ORE. It is the user's responsibility to make sure that this subset is sufficient to cover the portfolio to be analysed. If it is not, the application will complain at run time and exit.

We assume that the market configuration is provided in file `todaysmarket.xml`, however, the file name can be chosen by the user. The file name needs to be entered into the master configuration file `ore.xml`, see section 7.1.

The file starts and ends with the opening and closing tags `<TodaysMarket>` and `</TodaysMarket>`. The file then contains configuration blocks for

- Discounting curves
- Index curves (to project index fixings)
- Yield curves (for other purposes, e.g. as benchmark curve for bond pricing)
- Swap index curves (to project Swap rates)
- FX spot rates
- Inflation index curves (to project zero or yoy inflation fixings)
- Equity curves (to project forward prices)
- Default curves

- Swaption volatility structures
- Cap/Floor volatility structures
- FX volatility structures
- Inflation Cap/Floor price surfaces
- Equity volatility structures
- CDS volatility structures
- Base correlation structures
- Correlation structures
- Securities

There can be alternative versions of each block each labeled with a unique identifier (e.g. Discount curve block with ID 'default', discount curve block with ID 'ois', another one with ID 'xois', etc). The purpose of these IDs will be explained at the end of this section. We now discuss each block's layout.

7.2.1 Discounting Curves

We pick one discounting curve block as an example here (see `Examples/Input/todaysmarket.xml`), the one with ID 'ois'

Listing 9: Discount curve block with ID 'ois'

```
<DiscountingCurves Id="ois">
  <DiscountingCurve Currency="EUR">Yield/EUR/EUR1D</DiscountingCurve>
  <DiscountingCurve Currency="USD">Yield/USD/USD1D</DiscountingCurve>
  <DiscountingCurve Currency="GBP">Yield/GBP/GBP1D</DiscountingCurve>
  <DiscountingCurve Currency="CHF">Yield/CHF/CHF6M</DiscountingCurve>
  <DiscountingCurve Currency="JPY">Yield/JPY/JPY6M</DiscountingCurve>
  <!-- ... -->
</DiscountingCurves>
```

This block instructs ORE to build five discount curves for the indicated currencies. The string within the tags, e.g. Yield/EUR/EUR1D, uniquely identifies the curve to be built. Curve Yield/EUR/EUR1D is defined in the curve configuration file explained in section 7.7 below. In this case ORE is instructed to build an Eonia Swap curve made of Overnight Deposit and Eonia Swap quotes. The right most token of the string Yield/EUR/EUR1D (EUR1D) is user defined, the first two tokens Yield/EUR have to be used to point to a yield curve in currency EUR.

7.2.2 Index Curves

See an excerpt of the index curve block with ID 'default' from the same example file:

Listing 10: Index curve block with ID 'default'

```
<IndexForwardingCurves Id="default">
  <Index Name="EUR-EURIBOR-3M">Yield/EUR/EUR3M</Index>
  <Index Name="EUR-EURIBOR-6M">Yield/EUR/EUR6M</Index>
  <Index Name="EUR-EURIBOR-12M">Yield/EUR/EUR6M</Index>
  <Index Name="EUR-EONIA">Yield/EUR/EUR1D</Index>
  <Index Name="USD-LIBOR-3M">Yield/USD/USD3M</Index>
  <!-- ... -->
</IndexForwardingCurves>
```

This block of curve specifications instructs ORE to build another set of yield curves, unique strings (e.g. Yield/EUR/EUR6M etc.) point to the `curveconfig.xml` file where these curves are defined. Each curve is then associated with an index name (of format Ccy-IndexName-Tenor, e.g. EUR-EURIBOR-6M) so that ORE will project the respective index using the selected curve (e.g. Yield/EUR/EUR6M).

7.2.3 Yield Curves

See an excerpt of the yield curve block with ID 'default' from the same example file:

Listing 11: Yield curve block with ID 'default'

```
<YieldCurves id="default">
  <YieldCurve name="BANK_EUR_LEND">Yield/EUR/BANK_EUR_LEND</YieldCurve>
  <YieldCurve name="BANK_EUR_BORROW">Yield/EUR/BANK_EUR_BORROW</YieldCurve>
  <!-- ... -->
</YieldCurves>
```

This block of curve specifications instructs ORE to build another set of yield curves, unique strings (e.g. Yield/EUR/EUR6M etc.) point to the `curveconfig.xml` file where these curves are defined. Other than discounting and index curves the yield curves in this block are not tied to a particular purpose. The curves defined in this block typically include

- additional curves needed in the XVA post processor, e.g. for the FVA calculation
- benchmark curves used for bond pricing

7.2.4 Swap Index Curves

The following is an excerpt of the swap index curve block with ID 'default' from the same example file:

Listing 12: Swap index curve block with ID 'default'

```
<SwapIndexCurves Id="default">
  <SwapIndex Name="EUR-CMS-1Y">
    <Index>EUR-EURIBOR-6M</Index>
    <Discounting>EUR-EONIA</Discounting>
  </SwapIndex>
  <SwapIndex Name="EUR-CMS-30Y">
    <Index>EUR-EURIBOR-6M</Index>
    <Discounting>EUR-EONIA</Discounting>
  </SwapIndex>
  <!-- ... -->
</SwapIndexCurves>
```

These instructions do not build any additional curves. They only build the respective swap index objects and associate them with the required index forwarding and discounting curves already built above. This enables a swap index to project the fair rate of forward starting Swaps. Swap indices are also containers for conventions. Swaption volatility surfaces require two swap indices each available in the market object, a long term and a short term swap index. The curve configuration file below will show that in particular the required short term index has term 1Y, and the required long term index has 30Y term. This is why we build these two indices at this point.

7.2.5 FX Spot

The following is an excerpt of the FX spot block with ID 'default' from the same example file:

Listing 13: FX spot block with ID 'default'

```
<FxSpots Id="default">
  <FxSpot Pair="EURUSD">FX/EUR/USD</FxSpot>
  <FxSpot Pair="EURGBP">FX/EUR/GBP</FxSpot>
  <FxSpot Pair="EURCHF">FX/EUR/CHF</FxSpot>
  <FxSpot Pair="EURJPY">FX/EUR/JPY</FxSpot>
  <!-- ... -->
</FxSpots>
```

This block instructs ORE to provide four FX quotes, all quoted with target currency EUR so that foreign currency amounts can be converted into EUR via multiplication with that rate.

7.2.6 FX Volatilities

The following is an excerpt of the FX Volatilities block with ID 'default' from the same example file:

Listing 14: FX volatility block with ID 'default'

```xml
<FxVolatilities Id="default">
  <FxVolatility Pair="EURUSD">FXVolatility/EUR/USD/EURUSD</FxVolatility>
  <FxVolatility Pair="EURGBP">FXVolatility/EUR/GBP/EURGBP</FxVolatility>
  <FxVolatility Pair="EURCHF">FXVolatility/EUR/CHF/EURCHF</FxVolatility>
  <FxVolatility Pair="EURJPY">FXVolatility/EUR/JPY/EURJPY</FxVolatility>
  <!-- ... -->
</FxVolatilities>
```

This instructs ORE to build four FX volatility structures for all FX pairs with target currency EUR, see curve configuration file for the definition of the volatility structure.

7.2.7 Swaption Volatilities

The following is an excerpt of the Swaption Volatilities block with ID 'default' from the same example file:

Listing 15: Swaption volatility block with ID 'default'

```xml
<SwaptionVolatilities Id="default">
  <SwaptionVolatility Currency="EUR">SwaptionVolatility/EUR/EUR_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="USD">SwaptionVolatility/USD/USD_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="GBP">SwaptionVolatility/GBP/GBP_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="CHF">SwaptionVolatility/CHF/CHF_SW_N</SwaptionVolatility>
  <SwaptionVolatility Currency="JPY">SwaptionVolatility/CHF/JPY_SW_N</SwaptionVolatility>
</SwaptionVolatilities>
```

This instructs ORE to build five Swaption volatility structures, see the curve configuration file for the definition of the volatility structure. The latter token (e.g. EUR_SW_N) is user defined and will be found in the curve configuration's CurveId tag.

7.2.8 Cap/Floor Volatilities

The following is an excerpt of the Cap/Floor Volatilities block with ID 'default' from the same example file:

Listing 16: Cap/Floor volatility block with ID 'default'

```xml
<CapFloorVolatilities id="default">
  <CapFloorVolatility currency="EUR">CapFloorVolatility/EUR/EUR_CF_N</CapFloorVolatility>
  <CapFloorVolatility currency="USD">CapFloorVolatility/USD/USD_CF_N</CapFloorVolatility>
  <CapFloorVolatility currency="GBP">CapFloorVolatility/GBP/GBP_CF_N</CapFloorVolatility>
</CapFloorVolatilities>
```

This instructs ORE to build three Cap/Floor volatility structures, see the curve configuration file for the definition of the volatility structure. The latter token (e.g. EUR_CF_N) is user defined and will be found in the curve configuration's CurveId tag.

7.2.9 Default Curves

The following is an excerpt of the Default Curves block with ID 'default' from the same example file:

Listing 17: Default curves block with ID 'default'

```
<DefaultCurves Id="default">
  <DefaultCurve Name="BANK">Default/USD/BANK_SR_USD</DefaultCurve>
  <DefaultCurve Name="CPTY_A">Default/USD/CUST_A_SR_USD</DefaultCurve>
  <DefaultCurve Name="CPTY_B">Default/USD/CUST_A_SR_USD</DefaultCurve>
  <!-- ... -->
</DefaultCurves>
```

This instructs ORE to build a set of default probability curves, again defined in the curve configuration file. Each curve is then associated with a name (BANK, CUST_A) for subsequent lookup. As before, the last token (e.g. BANK_SR_USD) is user defined and will be found in the curve configuration's CurveId tag.

7.2.10 Securities

The following is an excerpt of the Security block with ID 'default' from the same example file:

Listing 18: Securities block with ID 'default'

```
<Securities id="default">
  <Security name="SECURITY_1">Security/SECURITY_1</Security>
</Securities>
```

The pricing of bonds includes (among other components) a security specific spread and rate. This block links a security name to a spread and rate pair defined in the curve configuration file. This name may then be referenced as the security id in the bond trade definition.

7.2.11 Equity Curves

The following is an excerpt of the Equity curves block with ID 'default' from the same example file:

Listing 19: Equity curves block with ID 'default'

```
<EquityCurves id="default">
  <EquityCurve name="SP5">Equity/USD/SP5</EquityCurve>
  <EquityCurve name="Lufthansa">Equity/EUR/Lufthansa</EquityCurve>
</EquityCurves>
```

This instructs ORE to build a set of equity curves, again defined in the curve configuration file. Each equity curve after construction will consist of a spot equity price, as well as a term structure of dividend yields, which can be used to determine forward prices. This object is then associated with a name (e.g. SP5) for subsequent lookup.

7.2.12 Equity Volatilities

The following is an excerpt of the equity volatilities block with ID 'default' from the same example file:

Listing 20: EQ volatility block with ID 'default'

```
<EquityVolatilities id="default">
  <EquityVolatility name="SP5">EquityVolatility/USD/SP5</EquityVolatility>
  <EquityVolatility name="Lufthansa">EquityVolatility/EUR/Lufthansa</EquityVolatility>
</EquityVolatilities>
```

This instructs ORE to build two equity volatility structures for SP5 and Lufthansa, respectively. See the curve configuration file for the definition of the equity volatility structure.

7.2.13 Inflation Index Curves

The following is an excerpt of the Zero Inflation Index Curves block with ID 'default' from the sample example file:

Listing 21: Zero Inflation Index Curves block with ID 'default'

```
<ZeroInflationIndexCurves id="default">
    <ZeroInflationIndexCurve name="EUHICPXT">
        Inflation/EUHICPXT/EUHICPXT_ZC_Swaps
    </ZeroInflationIndexCurve>
    <ZeroInflationIndexCurve name="FRHICP">
        Inflation/FRHICP/FRHICP_ZC_Swaps
    </ZeroInflationIndexCurve>
    <ZeroInflationIndexCurve name="UKRPI">
        Inflation/UKRPI/UKRPI_ZC_Swaps
    </ZeroInflationIndexCurve>
    <ZeroInflationIndexCurve name="USCPI">
        Inflation/USCPI/USCPI_ZC_Swaps
    </ZeroInflationIndexCurve>
    ...
</ZeroInflationIndexCurves>
```

This instructs ORE to build a set of zero inflation index curves, which are defined in the curve configuration file. Each curve is then associated with an index name (like e.g. EUHICPXT or UKRPI). The last token (e.g. EUHICPXT_ZC_Swap) is user defined and will be found in the curve configuration's CurveId tag.
In a similar way, Year on Year index curves are specified:

Listing 22: YoY Inflation Index Curves block with ID 'default'

```
<YYInflationIndexCurves id="default">
    <YYInflationIndexCurve name="EUHICPXT">
        Inflation/EUHICPXT/EUHICPXT_YY_Swaps
    </YYInflationIndexCurve>
    ...
</YYInflationIndexCurves>
```

Note that the index name is the same as in the corresponding zero index curve definition, but the token corresponding to the CurveId tag is different. This is because the actual underlying index (and in particular its fixings) are shared between the two index types, while different projection curves are used to forecast future index realisations.

7.2.14 Inflation Cap/Floor Price Surfaces

The following is an excerpt of the Inflation Cap/Floor Price Surfaces block with ID 'default' from the sample example file:

Listing 23: Inflation Cap/Floor Price Surfaces block with ID 'default'

```xml
<InflationCapFloorPriceSurfaces id="default">
    <InflationCapFloorPriceSurface name="EUHICPXT">
        InflationCapFloorPrice/EUHICPXT/EUHICPXT_ZC_CF
    </InflationCapFloorPriceSurface>
    <InflationCapFloorPriceSurface name="USCPI">
        InflationCapFloorPrice/USCPI/USCPI_ZC_CF
    </InflationCapFloorPriceSurface>
    <InflationCapFloorPriceSurface name="UKRPI">
        InflationCapFloorPrice/UKRPI/UKRPI_ZC_CF
    </InflationCapFloorPriceSurface>
</InflationCapFloorPriceSurfaces>
```

This instructs ORE to build a set of zero inflation cap floor price surfaces, which are defined in the curve configuration file. Each surface is associated with an idnex name. The last token (e.g. EUHICPXT_ZC_CF) is user defined and will be found in the curve configuration's CurveId tag.
Currently only zero coupon surfaces are supported, year on year surfaces are not supported.

7.2.15 CDS Volatility Structures

CDS volatility structures are configured as follows

Listing 24: CDS volatility structure block with ID 'default'

```xml
<CDSVolatilities id="default">
 <CDSVolatility name="CDSVOL_A">CDSVolatility/CDXIG</CDSVolatility>
 <CDSVolatility name="CDSVOL_B">CDSVolatility/CDXHY</CDSVolatility>
</CDSVolatilities>
```

The composition of the CDS volatility structures is defined in the curve configuration.

7.2.16 Base Correlation Structures

Base correlation structures are configured as follows

Listing 25: Base Correlations block with ID 'default'

```xml
<BaseCorrelations id="default">
 <BaseCorrelation name="CDXIG">BaseCorrelation/CDXIG</BaseCorrelation>
</BaseCorrelations>
```

The composition of the base correlation structure is defined in the curve configuration.

7.2.17 Correlation Structures

Correlation structures are configured as follows

Listing 26: Correlations block with ID 'default'

```xml
<Correlations id="default">
     <Correlation name="EUR-CMS-10Y:EUR-CMS-1Y">Correlation/EUR-CORR</Correlation>
     <Correlation name="USD-CMS-10Y:USD-CMS-1Y">Correlation/USD-CORR</Correlation>
</Correlations>
```

The composition of the correlation structure is defined in the curve configuration.

7.2.18 Market Configurations

Finally, representatives of each type of block (Discount Curves, Index Curves, Volatility structures, etc, up to Inflation Cap/Floor Price Surfaces) can be bundled into a market configuration. This is done by adding the following to the `todaysmarket.xml` file:

Listing 27: Market configurations

```xml
<Configuration Id="default">
  <DiscountingCurvesId>xois_eur</DiscountingCurvesId>
</Configuration>
<Configuration Id="collateral_inccy">
  <DiscountingCurvesId>ois</DiscountingCurvesId>
</Configuration>
<Configuration Id="collateral_eur">
  <DiscountingCurvesId>xois_eur</DiscountingCurvesId>
</Configuration>
<Configuration Id="libor">
  <DiscountingCurvesId>inccy_swap</DiscountingCurvesId>
</Configuration>
```

When ORE constructs the market object, all market configurations will be build and labelled using the 'Configuration Id'. This allows configuring a market setup for different alternative purposes side by side in the same `todaysmarket.xml` file. Typical use cases are

- different discount curves needed for model calibration and risk factor evolution, respectively
- different discount curves needed for collateralised and uncollateralised derivatives pricing.

The former is actually used throughout the `Examples` section. Each master input file `ore.xml` has a Markets section (see 7.1) where four market configuration IDs have to be provided - the ones used for 'lgmcalibration', 'fxcalibration', 'pricing' and 'simulation' (i.e. risk factor evolution).

The configuration ID concept extends across all curve and volatility objects though currently used only to distinguish discounting.

7.3 Pricing Engines: pricingengine.xml

The pricing engine configuration file is provided to select pricing models and pricing engines by product type. The following is an overview over the Example section's pricingengine.xml. Further below we discuss the Bermudan Swaption engine parametrisation in more detail.

```xml
<PricingEngines>
  <Product type="Swap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingSwapEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="CrossCurrencySwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingCrossCurrencySwapEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="FxForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingFxForwardEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="FxOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="EuropeanSwaption">
    <Model>BlackBachelier</Model> <!-- depends on input vol -->
    <ModelParameters/>
    <Engine>BlackBachelierSwaptionEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="Bond">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingRiskyBondEngine</Engine>
    <EngineParameters>
      <Parameter name="TimestepPeriod">6M</Parameter>
    </EngineParameters>
  </Product>
  <Product type="BermudanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
      <Parameter name="Calibration">Bootstrap</Parameter>
      <Parameter name="BermudanStrategy">CoterminalATM</Parameter>
      <!-- ccy specific reversions -->
      <Parameter name="Reversion_EUR">0.03</Parameter>
      <Parameter name="Reversion_USD">0.04</Parameter>
      <!-- reversion to use if no ccy specific value is given -->
      <Parameter name="Reversion">0.02</Parameter>
      <Parameter name="ReversionType">HullWhite</Parameter>
      <Parameter name="Volatility">0.01</Parameter>
```

```xml
      <Parameter name="VolatilityType">Hagan</Parameter>
      <Parameter name="ShiftHorizon">0.5</Parameter>
      <Parameter name="Tolerance">0.0001</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
      <Parameter name="sy">3.0</Parameter>
      <Parameter name="ny">10</Parameter>
      <Parameter name="sx">3.0</Parameter>
      <Parameter name="nx">10</Parameter>
    </EngineParameters>
  </Product>
  <Product type="CapFloor">
    <Model>IborCapModel</Model>
    <ModelParameters/>
    <Engine>IborCapEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="CapFlooredIborLeg">
    <Model>BlackOrBachelier</Model>
    <ModelParameters/>
    <Engine>BlackIborCouponPricer</Engine>
    <EngineParameters>
      <!-- Black76 or BivariateLognormal -->
      <TimingAdjustment>Black76</TimingAdjustment>
      <!-- Correlation Parameter for BivariateLognormal -->
      <Correlation>1.0</Correlation>
    </EngineParameters>
  </Product>
  <Product type="EquityForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingEquityForwardEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="EquityOption">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="Bond">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingRiskyBondEngine</Engine>
    <EngineParameters>
      <Parameter name="TimestepPeriod">6M</Parameter>
    </EngineParameters>
  </Product>
  <Product type="CreditDefaultSwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>MidPointCdsEngine</Engine>
    <EngineParameters/>
  </Product>
  <Product type="CMS">
    <Model>Hagan</Model><!-- or LinearTSR -->
    <ModelParameters/>
```

```xml
    <Engine>Analytic</Engine> <!-- or Numerical -->
    <EngineParameters>
      <!-- Alternative Yield Curve Models: ExactYield, ParallelShifts, NonParallelShifts -->
      <Parameter name="YieldCurveModel">Standard</Parameter>
      <Parameter name="MeanReversion_EUR">0.01</Parameter>
      <Parameter name="MeanReversion_USD">0.02</Parameter>
      <Parameter name="MeanReversion">0.0</Parameter>
    </EngineParameters>
  </Product>
  <Product type="CMSSpread">
    <Model>BrigoMercurio</Model>
    <ModelParameters/>
    <Engine>Analytic</Engine>
    <EngineParameters>
      <Parameter name="IntegrationPoints">16</Parameter>
    </EngineParameters>
  </Product>
</PricingEngines>
```

Listing 28: Pricing engine configuration

These settings will be taken into account when the engine factory is asked to build the respective pricing engines and required models, and to calibrate the required model.

For example, in case of the Bermudan Swaption, the parameters are interpreted as follows:

- The only model currently supported for Bermudan Swaption pricing is the LGM selected here.

- The first block of model parameters then provides initial values for the model (Reversion, Volatility) and chooses the parametrisation of the LGM model with ReversionType and VolatilityType choices *HullWhite* and *Hagan*. Notice the possibility to specify a currency-specific reversion. Calibration and BermudanStrategy can be set to *None* in order to skip model calibration. Alternatively, Calibration is set to *Bootstrap* and BermudanStrategy to *CoterminalATM* in order to calibrate to instrument-specific co-terminal ATM Swaptions, i.e. chosen to match the instruments first expiry and final maturity. If *CoterminalDealStrike* is chosen, the co-terminal swaptions will match the fixed rate of the deal (if the deal has changing fixed rates, the first rate is matched). Finally if the ShiftHorizon parameter is given, its value times the remaining maturity time of the deal is chosen as the horizon shift parameter for the LGM model. If not given, this parameter defaults to 0.5.

- The second block of engine parameters specifies the Numerical Swaption engine parameters which determine the number of standard deviations covered in the probability density integrals (sy and sx), and the number of grid points used per standard deviation (ny and nx).

To see the configuration options for the alternative CMS engines (Hagan Numerical, LinearTSR) or the Black Ibor coupon pricer (CapFlooredIborLeg), please refer to the commented parts in `Examples/Input/pricingengine.xml`.

This file is relevant in particular for structured products which are on the roadmap of future ORE releases. But it is also intended to allow the selection of optimised pricing engines for vanilla products such as Interest Rate Swaps.

7.4 Simulation: `simulation.xml`

This file determines the behaviour of the risk factor simulation (scenario generation) module. It is structured in three blocks of data.

Listing 29: Simulation configuration

```
<Simulation>
  <Parameters> ... </Parameters>
  <CrossAssetModel> ... </CrossAssetModel>
  <Market> ... </Market>
</Simulation>
```

Each of the three blocks is sketched in the following.

7.4.1 Parameters

Let us discuss this section using the following example

Listing 30: Simulation configuration

```
<Parameters>
  <Discretization>Exact</Discretization>
  <Grid>80,3M</Grid>
  <Calendar>EUR,USD,GBP,CHF</Calendar>
  <Sequence>SobolBrownianBridge</Sequence>
  <Seed>42</Seed>
  <Samples>1000</Samples>
  <Ordering>Steps</Ordering>
  <DirectionIntegers>JoeKuoD7</DirectionIntegers>
</Parameters>
```

- `Discretization:` Chooses between time discretization schemes for the risk factor evolution. *Exact* means exploiting the analytcal tractability of the model to avoid any time discretization error. *Euler* uses a naive time discretization scheme which has numerical error and requires small time steps for accurate results (useful for testing purposes)
- `Grid:` Specifies the simulation time grid, here 80 quarterly steps.[6]
- `Calendar:` Calendar or combination of calendars used to adjust the dates of the grid. Date adjustment is required because the simulation must step over 'good' dates on which index fixings can be stored.
- `Sequence:` Choose random sequence generator (*MersenneTwister, MersenneTwister-Antithetic, Sobol, SobolBrownianBridge*).
- `Seed:` Random number generator seed

[6]For exposure calculation under DIM, the second parameter has to match the Margin Period of Risk, i.e. if `MarginPeriodOfRisk` is set to for instance `2W` in a netting set definition in `netting.xml`, then one has to set `Grid` to for instance `80,2W`.

- `Samples:` Number of Monte Carlo paths to be produced use (*Backward, Forward, BestOfForwardBackward, InterpolatedForwardBackward*), which number of forward horizon days to use if one of the *Forward* related methods is chosen.

- `Ordering:` If the sequence type *SobolBrownianBridge* is used, ordering of variates (*Factors, Steps, Diagonal*)

- `DirectionIntegers:` If the sequence type *SobolBrownianBridge* or *Sobol* is used, type of direction integers in Sobol generator (*Unit, Jaeckel, SobolLevitan, SobolLevitanLemieux, JoeKuoD5, JoeKuoD6, JoeKuoD7, Kuo, Kuo2, Kuo3*)

7.4.2 Model

The `CrossAssetModel` section determines the cross asset model's number of currencies covered, composition, and each component's calibration. It is currently made of

- a sequence of LGM models for each currency (say n_c currencies),
- $n_c - 1$ FX models for each exchange rate to the base currency,
- n_e equity models,
- n_i inflation models,
- a specification of the correlation structure between all components.

The simulated currencies are specified as follows, with clearly identifying the domestic currency which is also the target currency for all FX models listed subsequently. If the portfolio requires more currencies to be simulated, this will lead to an exception at run time, so that it is the user's responsibility to make sure that the list of currencies here is sufficient. The list can be larger than actually required by the portfolio. This will not lead to any exceptions, but add to the run time of ORE.

Listing 31: Simulation model currencies configuration

```xml
<CrossAssetModel>
  <DomesticCcy>EUR</DomesticCcy>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
    <Currency>GBP</Currency>
    <Currency>CHF</Currency>
    <Currency>JPY</Currency>
  </Currencies>
  <BootstrapTolerance>0.0001</BootstrapTolerance>
  <!-- ... -->
</CrossAssetModel>
```

Bootstrap tolerance is a global parameter that applies to the calibration of all model components. If the calibration error of any component exceeds this tolerance, this will trigger an exception at runtime, early in the ORE process.

Each interest rate model is specified by a block as follows

Listing 32: Simulation model IR configuration

```
<CrossAssetModel>
  <!-- ... -->
  <InterestRateModels>
    <LGM ccy="default">
      <CalibrationType>Bootstrap</CalibrationType>
      <Volatility>
        <Calibrate>Y</Calibrate>
        <VolatilityType>Hagan</VolatilityType>
        <ParamType>Piecewise</ParamType>
        <TimeGrid>1.0,2.0,3.0,4.0,5.0,7.0,10.0</TimeGrid>
        <InitialValue>0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01<InitialValue>
      </Volatility>
      <Reversion>
        <Calibrate>N</Calibrate>
        <ReversionType>HullWhite</ReversionType>
        <ParamType>Constant</ParamType>
        <TimeGrid/>
        <InitialValue>0.03</InitialValue>
      </Reversion>
      <CalibrationSwaptions>
        <Expiries>1Y,2Y,4Y,6Y,8Y,10Y,12Y,14Y,16Y,18Y,19Y</Expiries>
        <Terms>19Y,18Y,16Y,14Y,12Y,10Y,8Y,6Y,4Y,2Y,1Y</Terms>
        <Strikes/>
      </CalibrationSwaptions>
      <ParameterTransformation>
        <ShiftHorizon>0.0</ShiftHorizon>
        <Scaling>1.0</Scaling>
      </ParameterTransformation>
    </LGM>
    <LGM ccy="EUR">
      <!-- ... -->
    </LGM>
    <LGM ccy="USD">
      <!-- ... -->
    </LGM>
  </InterestRateModels>
  <!-- ... -->
</CrossAssetModel>
```

We have LGM sections by currency, but starting with a section for currency 'default'. As the name implies, this is used as default configuration for any currency in the currency list for which we do not provide an explicit parametrisation. Within each LGM section, the interpretation of elements is as follows:

- `CalibrationType`: Choose between *Bootstrap* and *BestFit*, where Bootstrap is chosen when we expect to be able to achieve a perfect fit (as with calibration of piecewise volatility to a series of co-terminal Swaptions)

- `Volatility/Calibrate`: Flag to enable/disable calibration of this particular parameter

- `Volatility/VolatilityType`: Choose volatility parametrisation a la *Hull-White* or *Hagan*

- `Volatility/ParamType`: Choose between *Constant* and *Piecewise*

- `Volatility/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Volatility/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `Reversion/Calibrate:` Flag to enable/disable calibration of this particular parameter
- `Reversion/VolatilityType:` Choose reversion parametrisation a la *HullWhite* or *Hagan*
- `Reversion/ParamType:` Choose between *Constant* and *Piecewise*
- `Reversion/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Reversion/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `CalibrationSwaptions:` Choice of calibration instruments by expiry, underlying Swap term and strike
- `ParameterTransformation:` LGM model prices are invariant under scaling and shift transformations [20] with advantages for numerical convergence of results in long term simulations. These transformations can be chosen here. Default settings are shiftHorizon 0 (time in years) and scaling factor 1.

Each FX model is specified by a block as follows

Listing 33: Simulation model FX configuration

```
<CrossAssetModel>
  <!-- ... -->
  <ForeignExchangeModels>
    <CrossCcyLGM foreignCcy="default">
      <DomesticCcy>EUR</DomesticCcy>
      <CalibrationType>Bootstrap</CalibrationType>
      <Sigma>
        <Calibrate>Y</Calibrate>
        <ParamType>Piecewise</ParamType>
        <TimeGrid>1.0,2.0,3.0,4.0,5.0,7.0,10.0</TimeGrid>
        <InitialValue>0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1</InitialValue>
      </Sigma>
      <CalibrationOptions>
        <Expiries>1Y,2Y,3Y,4Y,5Y,10Y</Expiries>
        <Strikes/>
      </CalibrationOptions>
    </CrossCcyLGM>
    <CrossCcyLGM foreignCcy="USD">
      <!-- ... -->
    </CrossCcyLGM>
    <CrossCcyLGM foreignCcy="GBP">
      <!-- ... -->
    </CrossCcyLGM>
    <!-- ... -->
  </ForeignExchangeModels>
  <!-- ... -->
<CrossAssetModel>
```

CrossCcyLGM sections are defined by foreign currency, but we also support a default configuration as above for the IR model parametrisations. Within each CrossCcyLGM section, the interpretation of elements is as follows:

- `DomesticCcy`: Domestic currency completing the FX pair
- `CalibrationType`: Choose between *Bootstrap* and *BestFit* as in the IR section
- `Sigma/Calibrate`: Flag to enable/disable calibration of this particular parameter
- `Sigma/ParamType`: Choose between *Constant* and *Piecewise*
- `Sigma/TimeGrid`: Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Sigma/InitialValue`: Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `CalibrationOptions`: Choice of calibration instruments by expiry and strike, strikes can be empty (implying the default, ATMF options), or explicitly specified (in terms of FX rates as absolute strike values, in delta notation such as $\pm 25D$, $ATMF$ for at the money)

Each equity model is specified by a block as follows

Listing 34: Simulation model equity configuration

```xml
<CrossAssetModel>
  <!-- ... -->
  <EquityModels>
    <CrossAssetLGM name="default">
      <Currency>EUR</Currency>
      <CalibrationType>Bootstrap</CalibrationType>
      <Sigma>
        <Calibrate>Y</Calibrate>
        <ParamType>Piecewise</ParamType>
        <TimeGrid>1.0,2.0,3.0,4.0,5.0,7.0,10.0</TimeGrid>
        <InitialValue>0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1</InitialValue>
      </Sigma>
      <CalibrationOptions>
        <Expiries>1Y,2Y,3Y,4Y,5Y,10Y</Expiries>
        <Strikes/>
      </CalibrationOptions>
    </CrossAssetLGM>
    <CrossAssetLGM name="SP5">
      <!-- ... -->
    </CrossAssetLGM>
    <CrossAssetLGM name="Lufthansa">
      <!-- ... -->
    </CrossAssetLGM>
      <!-- ... -->
  </EquityModels>
  <!-- ... -->
<CrossAssetModel>
```

CrossAssetLGM sections are defined by equity name, but we also support a default configuration as above for the IR and FX model parameterisations. Within each CrossAssetLGM section, the interpretation of elements is as follows:

- `Currency:` Currency of denomination
- `CalibrationType:` Choose between *Bootstrap* and *BestFit* as in the IR section
- `Sigma/Calibrate:` Flag to enable/disable calibration of this particular parameter
- `Sigma/ParamType:` Choose between *Constant* and *Piecewise*
- `Sigma/TimeGrid:` Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Sigma/InitialValue:` Vector of initial values, matching number of entries in time, or single value if the time grid is empty
- `CalibrationOptions:` Choice of calibration instruments by expiry and strike, strikes can be empty (implying the default, ATMF options), or explicitly specified (in terms of equity prices as absolute strike values)

The inflation model components are specified by a block as follows

Listing 35: Simulation model inflation component configuration

```
<CrossAssetModel>
  ...
  <InflationIndexModels>
    <LGM index="EUHICPXT">
      <Currency>EUR</Currency>
      <!-- As in the LGM parameterisation for any IR components -->
      <CalibrationType> ... </CalibrationType>
      <Volatility> ... </Volatility>
      <Reversion> ... </Reversion>
      <ParameterTransformation> ... </ParameterTransformation>
      <!-- Inflation model specific -->
      <CalibrationCapFloors>
        <!-- not used yet, as there is only one strategy so far -->
        <CalibrationStrategy> ... </CalibrationStrategy>
        <CapFloor> Floor </CapFloor> <!-- Cap, Floor -->
        <Expiries> 2Y, 4Y, 6Y, 8Y, 10Y </Expiries>
        <!-- can be empty, this will yield calibration to ATM -->
        <Strikes> 0.03, 0.03, 0.03, 0.03, 0.03 </Strikes>
      </CalibrationCapFloors>
    </LGM>
    <LGM index="USCPI">
      ...
    </LGM>
    ...
  </InflationIndexModels>
  ...
<CrossAssetModel>
```

The inflation model parameterisation inherits from the LGM parameterisation for interest rate components, in particular the CalibrationType, Volatility and Reversion elements. The additional elements specify the model's calibration to a selection of either Caps or Floors.

Finally, the instantaneous correlation structure is specified as follows.

Listing 36: Simulation model correlation configuration

```
<CrossAssetModel>
  <!-- ... -->
  <InstantaneousCorrelations>
    <Correlation factor1="IR:EUR" factor2="IR:USD">0.3</Correlation>
    <Correlation factor1="IR:EUR" factor2="IR:GBP">0.3</Correlation>
    <Correlation factor1="IR:USD" factor2="IR:GBP">0.3</Correlation>
    <Correlation factor1="IR:EUR" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:EUR" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="IR:GBP" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:GBP" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="IR:USD" factor2="FX:USDEUR">0</Correlation>
    <Correlation factor1="IR:USD" factor2="FX:GBPEUR">0</Correlation>
    <Correlation factor1="FX:USDEUR" factor2="FX:GBPEUR">0</Correlation>
    <!-- ... -->
  </InstantaneousCorrelations>
</CrossAssetModel>
```

Any risk factor pair not specified explicitly here will be assumed to have zero correlation.

7.4.3 Market

The last part of the simulation configuration file covers the specification of the simulated market. Note that the simulation model will yield the evolution of risk factors such as short rates which need to be translated into entire yield curves that can be 'understood' by the instruments which we want to price under scenarios.
Moreover we need to specify how volatility structures evolve even if we do not explicitly simulate volatility. This translation happens based on the information in the *simulation market* object, which is configured in the section within the enclosing tags `<Market>` and `</Market>`, as shown in the following small example.
It should be noted that equity volatilities are taken to be a curve by default. To simulate an equity volatility surface with smile the xml node `<Surface>` must be supplied. There are two methods in ORE for equity volatility simulation:

- Simulating ATM volatilities only (and shifting other strikes relative to this using the T_0 smile). In this case set `<SimulateATMOnly>` to true.
- Simulating the full volatility surface. The node `<SimulateATMOnly>` should be omitted or set to false, and explicit moneyness levels for simulation should be provided.

Swaption volatilities are taken to be a surface by default. To simulate a swaption volatility cube with smile the xml node `<Cube>` must be supplied. There are two methods in ORE for swaption volatility cube simulation:

- Simulating ATM volatilities only (and shifting other strikes relative to this using the T_0 smile). In this case set `<SimulateATMOnly>` to true.
- Simulating the full volatility cube. The node `<SimulateATMOnly>` should be omitted or set to false, and explicit strike spreads for simulation should be provided.

FX volatilities are taken to be a curve by default. To simulate an FX volatility cube with smile the xml node `<Surface>` must be supplied. The surface node contains the moneyness levels to be simulated.
For Yield Curves, Swaption Volatilities, CapFloor Volatilities, Default Curves, Base Correlations and Inflation Curves, a DayCounter may be specified for each riskfactor using the node `<DayCounter name="EXAMPLE_CURVE">`. If no day counter is specified for a given riskfactor then the default Actual365 is used. To specify a new default for a riskfactor type then use the daycounter node without any attribute, `<DayCounter>`.

```
<Market>
  <BaseCurrency>EUR</BaseCurrency>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </Currencies>
  <YieldCurves>
    <Configuration>
      <Tenors>3M,6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,12Y,15Y,20Y</Tenors>
      <Interpolation>LogLinear</Interpolation>
      <Extrapolation>Y</Extrapolation>
      <DayCounter>ACT/ACT</DayCounter> //Sets a new default for all yieldCurves
    </Configuration>
  </YieldCurves>
```

```xml
<Indices>
  <Index>EUR-EURIBOR-6M</Index>
  <Index>EUR-EURIBOR-3M</Index>
  <Index>EUR-EONIA</Index>
  <Index>USD-LIBOR-3M</Index>
</Indices>
<SwapIndices>
  <SwapIndex>
    <Name>EUR-CMS-1Y</Name>
    <ForwardingIndex>EUR-EURIBOR-6M</ForwardingIndex>
    <DiscountingIndex>EUR-EONIA</DiscountingIndex>
  </SwapIndex>
</SwapIndices>
<DefaultCurves>
    <Names>
      <Name>CPTY1</Name>
      <Name>CPTY2</Name>
    </Names>
    <Tenors>6M,1Y,2Y</Tenors>
    <SimulateSurvivalProbabilities>true</SimulateSurvivalProbabilities>
    <DayCounter name="CPTY1">ACT/ACT</DayCounter>
</DefaultCurves>
<SwaptionVolatilities>
  <ReactionToTimeDecay>ForwardVariance</ReactionToTimeDecay>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </Currencies>
  <Expiries>6M,1Y,2Y,3Y,5Y,10Y,12Y,15Y,20Y</Expiries>
  <Terms>1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,30Y</Terms>
  <Cube>
   <SimulateATMOnly>false</SimulateATMOnly>
   <StrikeSpreads>-0.02,-0.01,0.0,0.01,0.02</StrikeSpreads>
  </Cube>
  <!-- Sets a new daycounter for just the EUR swaptionVolatility surface -->
  <DayCounter ccy="EUR">ACT/ACT</DayCounter>
</SwaptionVolatilities>
<CapFloorVolatilities>
  <ReactionToTimeDecay>ConstantVariance</ReactionToTimeDecay>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </Currencies>
  <DayCounter ccy="EUR">ACT/ACT</DayCounter>
</CapFloorVolatilities>
<FxVolatilities>
  <ReactionToTimeDecay>ForwardVariance</ReactionToTimeDecay>
  <CurrencyPairs>
    <CurrencyPair>EURUSD</CurrencyPair>
  </CurrencyPairs>
  <Expiries>6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y</Expiries>
  <Surface>
   <Moneyness>0.5,0.6,0.7,0.8,0.9</Moneyness>
  </Surface>
</FxVolatilities>
<EquityVolatilities>
    <Simulate>true</Simulate>
    <ReactionToTimeDecay>ForwardVariance</ReactionToTimeDecay>
```

```
      <!-- Alternative: ConstantVariance -->
      <Names>
        <Name>SP5</Name>
        <Name>Lufthansa</Name>
      </Names>
      <Expiries>6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y</Expiries>
      <Surface>
        <SimulateATMOnly>false</SimulateATMOnly><!-- false -->
        <Moneyness>0.1,0.5,1.0,1.5,2.0,3.0</Moneyness><!-- omitted if SimulateATMOnly true -->
      </Surface>
    </EquityVolatilities>
    ...
    <BenchmarkCurves>
  ^^I<BenchmarkCurve>
      <Currency>EUR</Currency>
      <Name>BENCHMARK_EUR</Name>
  ^^I</BenchmarkCurve>
  ^^I...
    </BenchmarkCurves>
    <Securities>
      <Simulate>true</Simulate>
      <Names>
        <Name>SECURITY_1</Name>
        ...
      </Names>
    </Securities>
    <ZeroInflationIndexCurves>
      <Names>
        <Name>EUHICP</Name>
        <Name>UKRPI</Name>
        <Name>USCPI</Name>
        ...
      </Names>
      <Tenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</Tenors>
    </ZeroInflationIndexCurves>
    <YYInflationIndexCurves>
      <Names>
        <Name>EUHICPXT</Name>
        ...
      </Names>
      <Tenors>1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</Tenors>
    </YYInflationIndexCurves>
    <DefaultCurves>
      <Names>
        <Name>ItraxxEuropeCrossoverS26V1</Name>
        ...
      </Names>
      <Tenors>1Y,2Y,3Y,5Y,10Y</Tenors>
      <SimulateSurvivalProbabilities>true</SimulateSurvivalProbabilities>
    </DefaultCurves>
    <BaseCorrelations/>
    <CDSVolatilities/>
    <Correlations>
      <Simulate>true</Simulate>
      <Pairs>
        <Pair>EUR-CMS-10Y,EUR-CMS-2Y</Pair>
      </Pairs>
      <Expiries>1Y,2Y</Expiries>
```

```
  </Correlations>
  <AdditionalScenarioDataCurrencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </AdditionalScenarioDataCurrencies>
  <AdditionalScenarioDataIndices>
    <Index>EUR-EURIBOR-3M</Index>
    <Index>EUR-EONIA</Index>
    <Index>USD-LIBOR-3M</Index>
  </AdditionalScenarioDataIndices>
</Market>
```

Listing 37: Simulation market configuration

7.5 Sensitivity Analysis: `sensitivity.xml`

ORE currently supports sensitivity analysis with respect to

- Discount curves (in the zero rate domain)
- Index curves (in the zero rate domain)
- Yield curves including e.g. equity forecast yield curves (in the zero rate domain)
- FX Spots
- FX volatilities
- Swaption volatilities, ATM matrix or cube
- Cap/Floor volatility matrices (in the caplet/floorlet domain)
- Default probability curves (in the "zero rate" domain, expressing survival probabilities $S(t)$ in term of zero rates $z(t)$ via $S(t) = \exp(-z(t) \times t)$ with Actual/365 day counter)
- Equity spot prices
- Equity volatilities, ATM or including strike dimension
- Zero inflation curves
- Year-on-Year inflation curves
- CDS volatilities
- Bond credit spreads
- Base correlation curves
- Correlation termstructures

The `sensitivity.xml` file specifies how sensitivities are computed for each market component. The general structure is shown in listing 38, for a more comprehensive case see `Examples/Example_15`. A subset of the following parameters is used in each market component to specify the sensitivity analysis:

- `ShiftType:` Both absolute or relative shifts can be used to compute a sensitivity, specified by the key words `Absolute` resp. `Relative`.
- `ShiftSize:` The size of the shift to apply.
- `ShiftTenors:` For curves, the tenor buckets to apply shifts to, given as a comma separated list of periods.
- `ShiftExpiries:` For volatility surfaces, the option expiry buckets to apply shifts to, given as a comma separated list of periods.
- `ShiftTerms:` For swaption volatility surfaces, the underlying terms to apply shifts to, given as a comma separated list of periods.
- `ShiftStrikes:` For cap/floor, FX option and equity option volatility surfaces, the strikes to apply shifts to, given as a comma separated list of absolute strikes
- `Index:` For cap / floor volatility surfaces, the index which together with the currency defines the surface. list of absolute strikes
- `CurveType:` In the context of Yield Curves used to identify an equity "risk free" rate forecasting curve; set to `EquityForecast` in this case

The cross gamma filter section contains a list of pairs of sensitivity keys. For each possible pair of sensitivity keys matching the given strings, a cross gamma sensitivity is computed. The given pair of keys can be (and usually are) shorter than the actual sensitivity keys. In this case only the prefix of the actual key is matched. For example, the pair `DiscountCurve/EUR,DiscountCurve/EUR` matches all actual sensitivity pairs belonging to a cross sensitivity by one pillar of the EUR discount curve and another (different) pillar of the same curve. We list the possible keys by giving an example in each category:

- `DiscountCurve/EUR/5/7Y`: 7y pillar of discounting curve in EUR, the pillar is at position 5 in the list of all pillars (counting starts with zero)
- `YieldCurve/BENCHMARK_EUR/0/6M`: 6M pillar of yield curve "BENCHMARK_EUR", the index of the 6M pillar is zero (i.e. it is the first pillar)
- `IndexCurve/EUR-EURIBOR-6M/2/2Y`: 2Y pillar of index forwarding curve for the Ibor index "EUR-EURIBOR-6M", the pillar index is 2 in this case
- `OptionletVolatility/EUR/18/5Y/0.04`: EUR caplet volatility surface, at 5Y option expiry and 4% strike, the running index for this expiry - strike pair is 18; the index counts the points in the surface in lexical order w.r.t. the dimensions option expiry, strike
- `FXSpot/USDEUR/0/spot`: FX spot USD vs EUR (with EUR as base ccy), the index is always zero for FX spots, the pillar is labelled as "spot" always
- `SwaptionVolatility/EUR/11/10Y/10Y/ATM`: EUR Swaption volatility surface at 10Y option expiry and 10Y underlying term, ATM level, the running index for this expiry, term, strike triple has running index 11; the index counts the points in the surface in lexical order w.r.t. the dimensions option expiry, underlying term and strike

```xml
<SensitivityAnalysis>
  <DiscountCurves>
    <DiscountCurve ccy="EUR">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
      <ShiftTenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</ShiftTenors>
    </DiscountCurve>
    ...
  </DiscountCurves>
  ...
  <IndexCurves>
    <IndexCurve index="EUR-EURIBOR-6M">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
      <ShiftTenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</ShiftTenors>
    </IndexCurve>
  </IndexCurves>
  ...
  <YieldCurves>
    <YieldCurve name="BENCHMARK_EUR">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
      <ShiftTenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</ShiftTenors>
    </YieldCurve>
  </YieldCurves>
  ...
  <FxSpots>
    <FxSpot ccypair="USDEUR">
      <ShiftType>Relative</ShiftType>
      <ShiftSize>0.01</ShiftSize>
    </FxSpot>
  </FxSpots>
  ...
  <FxVolatilities>
    <FxVolatility ccypair="USDEUR">
      <ShiftType>Relative</ShiftType>
      <ShiftSize>0.01</ShiftSize>
      <ShiftExpiries>1Y,2Y,3Y,5Y</ShiftExpiries>
      <ShiftStrikes/>
    </FxVolatility>
  </FxVolatilities>
  ...
  <SwaptionVolatilities>
    <SwaptionVolatility ccy="EUR">
      <ShiftType>Relative</ShiftType>
      <ShiftSize>0.01</ShiftSize>
      <ShiftExpiries>1Y,5Y,7Y,10Y</ShiftExpiries>
      <ShiftTerms>1Y,5Y,10Y</ShiftTerms>
      <ShiftStrikes/>
    </SwaptionVolatility>
  </SwaptionVolatilities>
  ...
  <CapFloorVolatilities>
    <CapFloorVolatility ccy="EUR">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
      <ShiftExpiries>1Y,2Y,3Y,5Y,7Y,10Y</ShiftExpiries>
      <ShiftStrikes>0.01,0.02,0.03,0.04,0.05</ShiftStrikes>
      <Index>EUR-EURIBOR-6M</Index>
    </CapFloorVolatility>
  </CapFloorVolatilities>
  <SecuritySpreads>
    <SecuritySpread security="SECURITY_1">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
    </SecuritySpread>
  </SecuritySpreads>
  ...
  <Correlations>
    <Correlation index1="EUR-CMS-10Y" index2="EUR-CMS-2Y">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.01</ShiftSize>
```

```
      <ShiftExpiries>1Y,2Y</ShiftExpiries>
      <ShiftStrikes>0</ShiftStrikes>
    </Correlation>
  </Correlations>
  ...
  <CrossGammaFilter>
    <Pair>DiscountCurve/EUR,DiscountCurve/EUR</Pair>
    <Pair>IndexCurve/EUR,IndexCurve/EUR</Pair>
    <Pair>DiscountCurve/EUR,IndexCurve/EUR</Pair>
  </CrossGammaFilter>
</SensitivityAnalysis>
```

Listing 38: Sensitivity configuration

7.6 Stress Scenario Analysis: `stressconfig.xml`

Stress tests can be applied in ORE to the same market segments and with same granularity as described in the sensitivity section 7.5.

This file `stressconfig.xml` specifies how stress tests can be configured. The general structure is shown in listing 39.
In this example, two stress scenarios "parallel_rates" and "twist" are defined. Each scenario definition contains the market components to be shifted in this scenario in a similar syntax that is also used for the sensitivity configuration, see 7.5. Components that should not be shifted, can just be omitted in the definition of the scenario.
However, instead of specifying one shift size per market component, here a whole vector of shifts can be given, with different shift sizes applied to each point of the curve (or surface / cube).
In case of the swaption volatility shifts, the single value given as `Shift` (without the attributes `expiry` and `term`) represents a default value that is used whenever no explicit value is given for a expiry / term pair.

```
<StressTesting>
  <StressTest id="parallel_rates">
    <DiscountCurves>
      <DiscountCurve ccy="EUR">
        <ShiftType>Absolute</ShiftType>
        <ShiftTenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</ShiftTenors>
        <Shifts>0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01</Shifts>
      </DiscountCurve>
      ...
    </DiscountCurves>
    <IndexCurves>
      ...
    </IndexCurves>
    <YieldCurves>
      ...
    </YieldCurves>
    <FxSpots>
      <FxSpot ccypair="USDEUR">
        <ShiftType>Relative</ShiftType>
        <ShiftSize>0.01</ShiftSize>
      </FxSpot>
    </FxSpots>
    <FxVolatilities>
      ...
    </FxVolatilities>
    <SwaptionVolatilities>
      <SwaptionVolatility ccy="EUR">
        <ShiftType>Absolute</ShiftType>
        <ShiftExpiries>1Y,10Y</ShiftExpiries>
        <ShiftTerms>5Y</ShiftTerms>
        <Shifts>
          <Shift>0.0010</Shift>
```

```
        <Shift expiry="1Y" term="5Y">0.0010</Shift>
        <Shift expiry="1Y" term="5Y">0.0010</Shift>
        <Shift expiry="1Y" term="5Y">0.0010</Shift>
        <Shift expiry="10Y" term="5Y">0.0010</Shift>
        <Shift expiry="10Y" term="5Y">0.0010</Shift>
        <Shift expiry="10Y" term="5Y">0.0010</Shift>
      </Shifts>
    </SwaptionVolatility>
  </SwaptionVolatilities>
  <CapFloorVolatilities>
    <CapFloorVolatility ccy="EUR">
      <ShiftType>Absolute</ShiftType>
      <ShiftExpiries>6M,1Y,2Y,3Y,5Y,10Y</ShiftExpiries>
      <Shifts>0.001,0.001,0.001,0.001,0.001,0.001</Shifts>
    </CapFloorVolatility>
  </CapFloorVolatilities>
 </StressTest>
 <StressTest id="twist">
  ...
 </StressTest>
</StressTesting>
```

Listing 39: Stress configuration

7.7 Curves: `curveconfig.xml`

The configuration of various term structures required to price a portfolio is covered in a single configuration file which we will label `curveconfig.xml` in the following though the file name can be chosen by the user. This configuration determines the composition of

- Yield curves
- Default curves
- Inflation curves
- Equity forward price curves
- Swaption volatility structures
- Cap/Floor volatility structures
- FX Option volatility structures
- CDS volatility structures
- Inflation Cap/Floor price surfaces
- Equity volatility structures
- Security spreads and recovery rates
- Base correlation curves
- Correlation termstructures

This file also contains other market objects such as FXSpots, Security Spreads and Security Rates which are necessary for the construction of a market.

7.7.1 Yield Curves

The top level XML elements for each `YieldCurve` node are shown in Listing 40.

Listing 40: Top level yield curve node

```
<YieldCurve>
  <CurveId> </CurveId>
  <CurveDescription> </CurveDescription>
  <Currency> </Currency>
  <DiscountCurve> </DiscountCurve>
  <Segments> </Segments>
  <InterpolationVariable> </InterpolationVariable>
  <InterpolationMethod> </InterpolationMethod>
  <ZeroDayCounter> </ZeroDayCounter>
  <Tolerance> </Tolerance>
  <Extrapolation> </Extrapolation>
</YieldCurve>
```

The meaning of each of the top level elements in Listing 40 is given below. If an element is labelled as 'Optional', then it may be excluded or included and left blank.

- CurveId: Unique identifier for the yield curve.
- CurveDescription: A description of the yield curve. This field may be left blank.
- Currency: The yield curve currency.
- DiscountCurve: If the yield curve is being bootstrapped from market instruments, this gives the CurveId of the yield curve used to discount cash flows during the bootstrap procedure. If this field is left blank or set equal to the current CurveId, then this yield curve itself is used to discount cash flows during the bootstrap procedure.
- Segments: This element contains child elements and is described in the following subsection.
- InterpolationVariable [Optional]: The variable on which the interpolation is performed. The allowable values are given in Table 6. If the element is omitted or left blank, then it defaults to *Discount*.
- InterpolationMethod [Optional]: The interpolation method to use. The allowable values are given in Table 7. If the element is omitted or left blank, then it defaults to *LogLinear*.
- ZeroDayCounter [Optional]: The day count basis used internally by the yield curve to calculate the time between dates. In particular, if the curve is queried for a zero rate without specifying the day count basis, the zero rate that is returned has this basis. If the element is omitted or left blank, then it defaults to *A365*.
- Tolerance [Optional]: The tolerance used by the root finding procedure in the bootstrapping algorithm. If the element is omitted or left blank, then it defaults to 1.0×10^{-12}.
- Extrapolation [Optional]: Set to *True* or *False* to enable or disable extrapolation respectively. If the element is omitted or left blank, then it defaults to *True*.

Variable	Description
Zero	The continuously compounded zero rate
Discount	The discount factor
Forward	The instantaneous forward rate

Table 6: Allowable interpolation variables.

Method	Description
Linear	Linear interpolation
LogLinear	Linear interpolation on the natural log of the interpolation variable
NaturalCubic	Monotonic Kruger cubic interpolation with second derivative at left and right
FinancialCubic	Monotonic Kruger cubic interpolation with second derivative at left and first derivative at right
ConvexMonotone	Convex Monotone Interpolation (Hagan, West)

Table 7: Allowable interpolation methods.

Segments Node

The `Segments` node gives the zero rates, discount factors and instruments that comprise the yield curve. This node consists of a number of child nodes where the node name depends on the segment being described. Each node has a `Type` that determines its structure. The following sections describe the type of child nodes that are available.

Direct Segment

When the node name is `Direct`, the `Type` node has the value *Zero* or *Discount* and the node has the structure shown in Listing 41. We refer to this segment here as a direct segment because the discount factors, or equivalently the zero rates, are given explicitly and do not need to be bootstrapped. The `Quotes` node contains a list of `Quote` elements. Each `Quote` element contains an ID pointing to a line in the `market.txt` file, i.e. in this case, pointing to a particular zero rate or discount factor. The `Conventions` node contains the ID of a node in the `conventions.xml` file described in section 7.8. The `Conventions` node associates conventions with the quotes.

Listing 41: Direct yield curve segment

```
<Direct>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
     <!--...-->
  </Quotes>
  <Conventions> </Conventions>
</Direct>
```

Simple Segment

When the node name is `Simple`, the `Type` node has the value *Deposit*, *FRA*, *Future*, *OIS* or *Swap* and the node has the structure shown in Listing 42. This segment holds quotes for a set of deposit, FRA, Future, OIS or swap instruments corresponding to the value in the `Type` node. These quotes will be used by the bootstrap algorithm to imply a discount factor, or equivalently a zero rate, curve. The only difference between this segment and the direct segment is that there is a `ProjectionCurve` node. This node allows us to specify the CurveId of another curve to project floating rates on the instruments underlying the quotes listed in the `Quote` nodes during the bootstrap procedure. This is an optional node. If it is left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId.

Listing 42: Simple yield curve segment

```
<Simple>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurve> </ProjectionCurve>
</Simple>
```

Average OIS Segment

When the node name is `AverageOIS`, the `Type` node has the value *Average OIS* and the node has the structure shown in Listing 43. This segment is used to hold quotes for Average OIS swap instruments. The `Quotes` node has the structure shown in Listing 44. Each quote for an Average OIS instrument (a typical example in a USD Overnight Index Swap) consists of two quotes, a vanilla IRS quote and an OIS-LIBOR basis swap spread quote. The IDs of these two quotes are stored in the `CompositeQuote` node. The `RateQuote` node holds the ID of the vanilla IRS quote and the `SpreadQuote` node holds the ID of the OIS-LIBOR basis swap spread quote.

Listing 43: Average OIS yield curve segment

```
<AverageOIS>
  <Type> </Type>
  <Quotes>
    <CompositeQuote> </CompositeQuote>
    <CompositeQuote> </CompositeQuote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurve> </ProjectionCurve>
</AverageOIS>
```

Listing 44: Average OIS segment's quotes section

```xml
<Quotes>
  <CompositeQuote>
    <SpreadQuote> </SpreadQuote>
    <RateQuote> </RateQuote>
  </CompositeQuote>
  <!--...-->
</Quotes>
```

Tenor Basis Segment

When the node name is `TenorBasis`, the `Type` node has the value *Tenor Basis Swap* or *Tenor Basis Two Swaps* and the node has the structure shown in Listing 45. This segment is used to hold quotes for tenor basis swap instruments. The quotes may be for a conventional tenor basis swap where Ibor of one tenor is swapped for Ibor of another tenor plus a spread. In this case, the `Type` node has the value *Tenor Basis Swap*. The quotes may also be for the difference in fixed rates on two fair swaps where one swap is against Ibor of one tenor and the other swap is against Ibor of another tenor. In this case, the `Type` node has the value *Tenor Basis Two Swaps*. Again, the structure is similar to the simple segment in Listing 42 except that there are two projection curve nodes. There is a `ProjectionCurveShort` node for the index with the shorter tenor. This node holds the CurveId of a curve for projecting the floating rates on the short tenor index. Similarly, there is a `ProjectionCurveLong` node for the index with the longer tenor. This node holds the CurveId of a curve for projecting the floating rates on the long tenor index. These are optional nodes. If they are left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId. However, at least one of the nodes needs to be populated to allow the bootstrap to proceed.

Listing 45: Tenor basis yield curve segment

```xml
<TenorBasis>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <ProjectionCurveLong> </ProjectionCurveLong>
  <ProjectionCurveShort> </ProjectionCurveShort>
</TenorBasis>
```

Cross Currency Segment

When the node name is `CrossCurrency`, the `Type` node has the value *FX Forward* or *Cross Currency Basis Swap*. When the `Type` node has the value *FX Forward*, the node has the structure shown in Listing 46. This segment is used to hold quotes for FX forward instruments. The `DiscountCurve` node holds the CurveId of a curve used to discount cash flows in the other currency i.e. the currency in the currency pair that is not equal to the currency in Listing 40. The `SpotRate` node holds the ID of a spot FX quote for the currency pair that is looked up in the `market.txt` file.

Listing 46: FX forward yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <DiscountCurve> </DiscountCurve>
  <SpotRate> </SpotRate>
</CrossCurrency>
```

When the `Type` node has the value *Cross Currency Basis Swap* then the node has the structure shown in Listing 47. This segment is used to hold quotes for cross currency basis swap instruments. The `DiscountCurve` node holds the CurveId of a curve used to discount cash flows in the other currency i.e. the currency in the currency pair that is not equal to the currency in Listing 40. The `SpotRate` node holds the ID of a spot FX quote for the currency pair that is looked up in the `market.txt` file. The `ProjectionCurveDomestic` node holds the CurveId of a curve for projecting the floating rates on the index in this currency i.e. the currency in the currency pair that is equal to the currency in Listing 40. It is an optional node and if it is left blank or omitted, then the projection curve is assumed to equal the curve being bootstrapped i.e. the current CurveId. Similarly, the `ProjectionCurveForeign` node holds the CurveId of a curve for projecting the floating rates on the index in the other currency. If it is left blank or omitted, then it is assumed to equal the CurveId provided in the `DiscountCurve` node in this segment.

Listing 47: Cross currency basis yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <DiscountCurve> </DiscountCurve>
  <SpotRate> </SpotRate>
  <ProjectionCurveDomestic> </ProjectionCurveDomestic>
  <ProjectionCurveForeign> </ProjectionCurveForeign>
</CrossCurrency>
```

Zero Spread Segment

When the node name is `ZeroSpread`, the `Type` node has the only allowable value *Zero Spread*, and the node has the structure shown in Listing 48. This segment is used to build yield curves which are expressed as a spread over some reference yield curve.

Listing 48: Zero spread yield curve segment

```xml
<ZeroSpread>
      <Type>Zero Spread</Type>
      <Quotes>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/2Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/5Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/10Y</Quote>
        <Quote>ZERO/YIELD_SPREAD/EUR/BANK_EUR_LEND/A365/20Y</Quote>
      </Quotes>
      <Conventions>EUR-ZERO-CONVENTIONS-TENOR-BASED</Conventions>
      <ReferenceCurve>EUR1D</ReferenceCurve>
</ZeroSpread>
```

7.7.2 Default Curves

Listing 49 shows the configuration of Default curves built from CDS spread quotes. Alternatively default curves can be set up as a difference curve of two yield curves as shown in listing 50. If $P_B(0,t)$ and $P_S(0,t)$ denote the discount factors of the given benchmark and source curve respectively the resulting default term structures has survival probabilities

$$S(0,t) = P_S(0,t)/P_B(0,t)$$

on the given pillars, and w.r.t. a a zero recovery rate. The interpolation is backward flat in the hazard rate.

```xml
<DefaultCurves>
  <DefaultCurve>
    <CurveId>BANK_SR_USD</CurveId>
    <CurveDescription>BANK SR CDS USD</CurveDescription>
    <Currency>USD</Currency>
    <Type>SpreadCDS</Type>
    <DiscountCurve>Yield/USD/USD3M</DiscountCurve>
    <DayCounter>A365</DayCounter>
    <RecoveryRate>RECOVERY_RATE/RATE/BANK/SR/USD</RecoveryRate>
    <Quotes>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/1Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/2Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/3Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/4Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/5Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/7Y</Quote>
      <Quote>CDS/CREDIT_SPREAD/BANK/SR/USD/10Y</Quote>
    </Quotes>
    <Conventions>CDS-STANDARD-CONVENTIONS</Conventions>
  </DefaultCurve>
  <DefaultCurve>
  ...
  </DefaultCurve>
</DefaultCurves>
```

Listing 49: Default curve configuration based on CDS quotes

```xml
    <DefaultCurve>
      <CurveId>BOND_YIELD_EUR_OVER_OIS</CurveId>
      <CurveDescription>Default curve derived as bond yield curve over Eonia</CurveDescription>
      <Currency>EUR</Currency>
      <Type>Benchmark</Type>
      <DayCounter>A365</DayCounter>
      <BenchmarkCurve>Yield/EUR/EUR1D</BenchmarkCurve>
      <SourceCurve>Yield/EUR/BOND_YIELD_EUR</SourceCurve>
      <Pillars>1Y,2Y,3Y,4Y,5Y,7Y,10Y</Pillars>
      <SpotLag>0</SpotLag>
      <Calendar>TARGET</Calendar>
    </DefaultCurve>
  </DefaultCurves>
```

Listing 50: Default curve as a difference of two yield curves

7.7.3 Swaption Volatility Structures

Listing 51 shows the configuration of Swaption volatility structures.

```xml
<SwaptionVolatilities>
  <SwaptionVolatility>
    <CurveId>EUR_SW_N</CurveId>
    <CurveDescription>EUR normal swaption volatilities</CurveDescription>
    <!-- ATM (Smile not yet supported) -->
    <Dimension>ATM</Dimension>
    <!-- Normal or Lognormal or ShiftedLognormal -->
    <VolatilityType>Normal</VolatilityType>
    <!-- Flat, Linear, None -->
    <Extrapolation>Flat</Extrapolation>
    <!-- Day counter for date to time conversion -->
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <!--Calendar and Business day convention for option tenor to date conversion -->
    <Calendar>TARGET</Calendar>
    <BusinessDayConvention>Following</BusinessDayConvention>
    <!-- ATM matrix specification -->
    <OptionTenors>1M,3M,6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</OptionTenors>
    <SwapTenors>1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</SwapTenors>
    <ShortSwapIndexBase>EUR-CMS-1Y</ShortSwapIndexBase>
    <SwapIndexBase>EUR-CMS-30Y</SwapIndexBase>
    <!-- Smile specification (optional) -->
    <SmileOptionTenors>6M,1Y,10Y</SmileOptionTenors>
    <SmileSwapTenors>2Y,5Y</SmileSwapTenors>
    <SmileSpreads>-0.02,-0.01,0.01,0.02</SmileSpreads> <!-- i.e. strike spreads -->
  </SwaptionVolatility>
  <SwaptionVolatility>
    ...
  </SwaptionVolatility>
</SwaptionVolatilities>
```

Listing 51: Swaption volatility configuration

7.7.4 Cap/Floor Volatility Structures

Listing 52 shows the configuration of Cap/Floor volatility structures.

```xml
  <CapFloorVolatilities>
    <CapFloorVolatility>
      <CurveId>EUR_CF_N</CurveId>
      <CurveDescription>EUR normal cap floor volatilities</CurveDescription>
      <!-- Normal, Lognormal or ShiftedLognormal -->
      <VolatilityType>Normal</VolatilityType>
      <!-- Linear, Flat, None -->
      <Extrapolation>Linear</Extrapolation>
      <!-- Include ATM vol quotes also: True or False -->
      <IncludeAtm>FALSE</IncludeAtm>
      <!-- Day counter for date to time conversion -->
      <DayCounter>Actual/365 (Fixed)</DayCounter>
      <!--Calendar and Business day convention for cap/floor term to date conversion -->
      <Calendar>TARGET</Calendar>
      <BusinessDayConvention>Following</BusinessDayConvention>
      <Tenors>1Y,2Y,3Y,4Y,5Y,6Y,7Y,8Y,9Y,10Y,15Y,20Y</Tenors>
      <Strikes>-0.01,0.0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1</Strikes>
      <IborIndex>EUR-EURIBOR-6M</IborIndex>
```

```
      <DiscountCurve>Yield/EUR/EUR1D</DiscountCurve>
    </CapFloorVolatility>
    ...
  </CapFloorVolatilities>
```

Listing 52: Cap/Floor volatility configuration

7.7.5 FX Volatility Structures

Listing 53 shows the configuration of FX volatility structures.

```
<FXVolatilities>
  <FXVolatility>
    <CurveId>EURUSD</CurveId>
    <CurveDescription />
    <Dimension>ATM</Dimension> <!-- ATM, Smile -->
    <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
    <FXSpotID>FX/EUR/USD</FXSpotID>
    <FXForeignCurveID>Yield/USD/USD1D</FXForeignCurveID>
    <FXDomesticCurveID>Yield/EUR/EUR1D</FXDomesticCurveID>
  </FXVolatility>
  <FXVolatility>
    ...
  </FXVolatility>
</FXVolatilities>
```

Listing 53: FX option volatility configuration

7.7.6 Equity Curve Structures

Listing 54 shows the configuration of equity forward price curves.

```
<EquityCurves>
  <EquityCurve>
    <CurveId>SP5</CurveId>
    <CurveDescription>SP 500 equity price projection curve</CurveDescription>
    <Currency>USD</Currency>
    <ForecastingCurve>EUR1D</ForecastingCurve>
    <Type>DividendYield</Type> <!-- DividendYield, ForwardPrice -->
    <!-- Spot quote from the market data file -->
    <SpotQuote>EQUITY/PRICE/SP5/USD</SpotQuote>
    <Quotes>
      <Quote>EQUITY_DIVIDEND/RATE/SP5/USD/3M</Quote>
      <Quote>EQUITY_DIVIDEND/RATE/SP5/USD/20160915</Quote>
      <Quote>EQUITY_DIVIDEND/RATE/SP5/USD/1Y</Quote>
      <Quote>EQUITY_DIVIDEND/RATE/SP5/USD/20170915</Quote>
    </Quotes>
    <DayCounter>A365</DayCounter>
  </EquityCurve>
  <EquityCurve>
    ...
  </EquityCurve>
  </EquityCurves>
```

Listing 54: Equity curve configuration

The equity curves here consists of a spot equity price, as well as a set of either forward prices or else dividend yields. Upon construction, ORE stores internally an equity spot price quote, a forecasting curve and a dividend yield term structure, which are then used together for projection of forward prices.

7.7.7 Equity Volatility Structures

Listing 55 shows the configuration of equity volatility structures.

```
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <Currency>USD</Currency>
    <Dimension>Smile</Dimension><!-- Alternative: ATM -->
    <Expiries>1M,5Y,10Y</Expiries>
    <!-- If Dimension is Smile: -->
    <Strikes>1932.8,2147.56,2254.939</Strikes>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 55: Equity option volatility configuration

7.7.8 Inflation Curves

Listing 56 shows the configuration of an inflation curve. The inflation curve specific elements are the following:

```
<InflationCurves>
   <InflationCurve>
       <CurveId>USCPI_ZC_Swaps</CurveId>
       <CurveDescription>Estimation Curve for USCPI</CurveDescription>
       <NominalTermStructure>Yield/USD/USD1D</NominalTermStructure>
       <Type>ZC</Type>
       <Quotes>
           <Quote>ZC_INFLATIONSWAP/RATE/USCPI/1Y</Quote>
           <Quote>ZC_INFLATIONSWAP/RATE/USCPI/2Y</Quote>
           ...
           <Quote>ZC_INFLATIONSWAP/RATE/USCPI/30Y</Quote>
           <Quote>ZC_INFLATIONSWAP/RATE/USCPI/40Y</Quote>
       </Quotes>
       <Conventions>USCPI_INFLATIONSWAP</Conventions>
       <Extrapolation>true</Extrapolation>
       <Calendar>US</Calendar>
       <DayCounter>A365</DayCounter>
       <Lag>3M</Lag>
       <Frequency>Monthly</Frequency>
       <BaseRate>0.01</BaseRate>
       <Tolerance>0.000000000001</Tolerance>
       <Seasonality>
           <BaseDate>20160101</BaseDate>
           <Frequency>Monthly</Frequency>
           <Factors>
```

```xml
            <Factor>SEASONALITY/RATE/MULT/USCPI/JAN</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/FEB</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/MAR</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/APR</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/MAY</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/JUN</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/JUL</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/AUG</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/SEP</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/OCT</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/NOV</Factor>
            <Factor>SEASONALITY/RATE/MULT/USCPI/DEC</Factor>
        </Factors>
      </Seasonality>
   </InflationCurve>
</InflationCurves>
```

Listing 56: Inflation Curve Configuration

- `NominalTermStructure`: The interest rate curve to be used to strip the inflation curve.
- `Type`: The type of the curve, `ZC` for zero coupon, `YY` for year on year.
- `Quotes`: The instruments' market quotes from which to bootstrap the curve.
- `Conventions`: The conventions applicable to the curve instruments.
- `Lag`: The observation lag used in the term structure.
- `Frequency`: The frequency of index fixings.
- `BaseRate`: The rate at $t = 0$, this introduces an additional degree of freedom to get a smoother curve. If not given, it is defaulted to the first market rate.

The optional seasonality block defines a multiplicative seasonality and contains the following elements:

- `BaseDate`: Defines the first inflation period to which to apply the seasonality correction, only day and month matters, the year is ignored.
- `Frequency:` Defines the frequency of the factors (usually identical to the index's fixing frequency).
- `Factors:` Multiplicative seasonality correction factors, must be part of the market data.

We note that if zero coupon swap market quotes are given, but the type is set to YY, the zero coupon swap quotes will be converted to year on year swap quotes on the fly, using the plain forward rates, i.e. no convexity adjustment is applied.

7.7.9 Inflation Cap/Floor Price Surfaces

Listing 57 shows the configuration of an zero coupon inflation cap floor price surface.

```
<InflationCapFloorPriceSurfaces>
  <InflationCapFloorPriceSurface>
      <CurveId>EUHICPXT_ZC_CF</CurveId>
      <CurveDescription>Price Surface ZC CapFloor EUHICPXT</CurveDescription>
      <Type>ZC</Type>
      <StartRate>0.10</StartRate>
      <ObservationLag>3M</ObservationLag>
      <Calendar>TARGET</Calendar>
      <BusinessDayConvention>MF</BusinessDayConvention>
      <DayCounter>A365</DayCounter>
      <Index>EUHICPXT</Index>
      <IndexCurve>Inflation/EUHICPXT/EUHICPXT_ZC_Swaps</IndexCurve>
      <IndexInterpolated>false</IndexInterpolated>
      <YieldTermStructure>Yield/EUR/EUR1D</YieldTermStructure>
      <CapStrikes>0.01,0.015,0.02,0.025,0.03</CapStrikes>
      <FloorStrikes>-0.02,-0.01,-0.005,0.00,0.01,0.015,0.02,0.025,0.03</FloorStrikes>
      <Maturities>1Y,2Y,3Y,4Y,5Y,6Y,7Y,8Y,9Y,10Y,12Y,15Y,20Y,30Y</Maturities>
  </InflationCapFloorPriceSurface>
</InflationCapFloorPriceSurfaces>
```

Listing 57: Inflation zc cap floor price surface configuration

- `Type`: The type of the surface, `ZC` for zero coupon, `YY` for year on year. Only zero coupon surfaces are supported currently.
- `StartRate`: A fall back value used in case the inflation index does not have a term structure attached. Should not be relevant.
- `Observation Lag`: The observation lag applicable to the term structure.
- `Index`: The underlying zero inflation index.
- `IndexCurve`: The curve id of the index's projection curve used to determine the ATM levels for the surface.
- `IndexInterpolated`: Flag indicating whether the index should be interpolating.
- `YieldTermStructure`: The nominal term structure.
- `CapStrikes`: The strikes for which cap prices are quoted (may (and will usually) overlap with the floor strike region).
- `FloorStrikes`: The strikes for which floor prices are quoted (may (and will usually) overlap with the cap strike region).
- `Maturities:` The maturities for which cap and floor prices are quoted

7.7.10 CDS Volatilities

Listing 58 shows the configuration of an ATM CDS volatility structure.

```
<CDSVolatilities>
  <CDSVolatility>
    <CurveId>CDXIG</CurveId>
    <CurveDescription>Lognormal option implied vols for CDX IG</CurveDescription>
    <Expiries>1M,3M,6M</Expiries>
  </CDSVolatility>
</CDSVolatilities>
```

Listing 58: CDS Volatility Configuration

7.7.11 Base Correlations

Listing 59 shows the configuration of a Base Correlation curve.

```
<BaseCorrelations>
  <BaseCorrelation>
    <CurveId>CDXIG</CurveId>
    <CurveDescription>CDX IG Base Correlations</CurveDescription>
    <Terms>1D</Terms>
    <DetachmentPoints>0.03, 0.06, 0.10, 0.20, 1.0</DetachmentPoints>
    <SettlementDays>0</SettlementDays>
    <Calendar>US</Calendar>
    <BusinessDayConvention>F</BusinessDayConvention>
    <DayCounter>A365</DayCounter>
    <Extrapolate>Y</Extrapolate>
  </BaseCorrelation>
</BaseCorrelations>
```

Listing 59: Base Correlation Configuration

7.7.12 FXSpots

Listing 60 shows the configuration of the fxSpots. It is assumed that each FXSpot CurveId is of the form "Ccy1Ccy2".

```
<FXSpots>
  <FXSpot>
    <CurveId>EURUSD</CurveId>
    <CurveDescription/>
  </FXSpot>
  <FXSpot>
    <CurveId>EURGBP</CurveId>
    <CurveDescription/>
  </FXSpot>
  <FXSpot>
    <CurveId>EURCHF</CurveId>
    <CurveDescription/>
  </FXSpot>
  <FXSpot>
    <CurveId>EURJPY</CurveId>
    <CurveDescription/>
  </FXSpot>
</FXSpots>
```

Listing 60: FXSpot Configuration

7.7.13 Securities

Listing 61 shows the configuration of the Securities. Each Security name is associated with a SpreadQuote and a RecoveryRateQuote.

```xml
<Securities>
  <Security>
    <CurveId>SECURITY_1</CurveId>
    <CurveDescription>Security</CurveDescription>
    <SpreadQuote>BOND/YIELD_SPREAD/SECURITY_1</SpreadQuote>
    <RecoveryRateQuote>RECOVERY_RATE/RATE/SECURITY_1</RecoveryRateQuote>
  </Security>
</Securities>
```

Listing 61: Security Configuration

7.7.14 Correlations

Listing 62 shows the configuration of the Correlations. The Correlation type can be either CMSSpread or Generic. The former one is to configure the correlation between two CMS indexes, the latter one is to generally configure the correlation between two indexes, e.g. between a CMS index and a IBOR index. Currently only ATM correlation curves or Flat correlation structures are supported. Correlation quotes may be loaded directly (by setting QuoteType to RATE) or calibrated from prices (set QuoteType to PRICE). Moreover a flat zero correlation curve can be loaded (by setting QuoteType to NULL). In this case market quotes are not needed to be provided. Currently only CMSSpread correlations can be calibrated. This is done using CMS Spread Options, and requires a CMSSpreadOption convention, SwaptionVolatility and DiscountCurve to be provided.

```xml
  <Correlations>
    <Correlation>
      <CurveId>EUR-CORR</CurveId>
      <CurveDescription>EUR CMS correlations</CurveDescription>
      <!--CMSSpread, Generic-->
      <CorrelationType>CMSSpread</CorrelationType>
      <Index1>EUR-CMS-10Y</Index1>
      <Index2>EUR-CMS-2Y</Index2>
      <!--Conventions, SwaptionVolatility and DiscountCurve only required when QuoteType = PRICE-->
      <Conventions>EUR-CMS-10Y-2Y-CONVENTION</Conventions>
      <SwaptionVolatility>EUR</SwaptionVolatility>
      <DiscountCurve>EUR-EONIA</DiscountCurve>
      <Currency>EUR</Currency>
      <!-- ATM, Constant -->
      <Dimension>ATM</Dimension>
      <!-- RATE, PRICE, NULL -->
      <QuoteType>PRICE</QuoteType>
      <Extrapolation>true</Extrapolation>
      <!-- Day counter for date to time conversion -->
      <DayCounter>Actual/365 (Fixed)</DayCounter>
      <!--Ccalendar and Business day convention for option tenor to date conversion -->
      <Calendar>TARGET</Calendar>
      <BusinessDayConvention>Following</BusinessDayConvention>
      <OptionTenors>1Y,2Y</OptionTenors>
    </Correlation>
```

Listing 62: Correlation Configuration

7.8 Conventions: `conventions.xml`

The conventions to associate with a set market quotes in the construction of termstructures are specified in another xml file which we will refer to as `conventions.xml` in the following though the file name can be chosen by the user. Each separate set of conventions is stored in an XML node. The type of conventions that a node holds is determined by the node name. Every node has an `Id` node that gives a unique identifier for the convention set. The following sections describe the type of conventions that can be created and the allowed values.

7.8.1 Zero Conventions

A node with name *Zero* is used to store conventions for direct zero rate quotes. Direct zero rate quotes can be given with an explicit maturity date or with a tenor and a set of conventions from which the maturity date is deduced. The node for a zero rate quote with an explicit maturity date is shown in Listing 63. The node for a tenor based zero rate is shown in Listing 64.

Listing 63: Zero conventions

```xml
<Zero>
  <Id> </Id>
  <TenorBased>False</TenorBased>
  <DayCounter> </DayCounter>
  <CompoundingFrequency> </CompoundingFrequency>
  <Compounding> </Compounding>
</Zero>
```

Listing 64: Zero conventions, tenor based

```xml
<Zero>
  <Id> </Id>
  <TenorBased>True</TenorBased>
  <DayCounter> </DayCounter>
  <CompoundingFrequency> </CompoundingFrequency>
  <Compounding> </Compounding>
  <TenorCalendar> </TenorCalendar>
  <SpotLag> </SpotLag>
  <SpotCalendar> </SpotCalendar>
  <RollConvention> </RollConvention>
  <EOM> </EOM>
</Zero>
```

The meanings of the various elements in this node are as follows:

- TenorBased: True if the conventions are for a tenor based zero quote and False if they are for a zero quote with an explicit maturity date.
- DayCounter: The day count basis associated with the zero rate quote (for choices see section 8.4)
- CompoundingFrequency: The frequency of compounding (Choices are *Once, Annual, Semiannual, Quarterly, Bimonthly, Monthly, Weekly, Daily*).

- Compounding: The type of compounding for the zero rate (Choices are *Simple, Compounded, Continuous, SimpleThenCompounded*).
- TenorCalendar: The calendar used to advance from the spot date to the maturity date by the zero rate tenor (for choices see section 8.4).
- SpotLag [Optional]: The number of business days to advance from the valuation date before applying the zero rate tenor. If not provided, this defaults to 0.
- SpotCalendar [Optional]: The calendar to use for business days when applying the `SpotLag`. If not provided, it defaults to a calendar with no holidays.
- RollConvention [Optional]: The roll convention to use when applying the zero rate tenor. If not provided, it defaults to Following (Choices are *Backward, Forward, Zero, ThirdWednesday, Twentieth, TwentiethIMM, CDS*).
- EOM [Optional]: Whether or not to use the end of month convention when applying the zero rate tenor. If not provided, it defaults to false.

7.8.2 Deposit Conventions

A node with name *Deposit* is used to store conventions for deposit or index fixing quotes. The conventions can be index based, in which case all necessary conventions are deduced from a given index family. The structure of the index based node is shown in Listing 65. Alternatively, all the necessary conventions can be given explicitly without reference to an index family. The structure of this node is shown in Listing 66.

Listing 65: Deposit conventions

```
<Deposit>
  <Id> </Id>
  <IndexBased>True</IndexBased>
  <Index> </Index>
</Deposit>
```

Listing 66: Deposit conventions

```
<Deposit>
  <Id> </Id>
  <IndexBased>False</IndexBased>
  <Calendar> </Calendar>
  <Convention> </Convention>
  <EOM> </EOM>
  <DayCounter> </DayCounter>
</Deposit>
```

The meanings of the various elements in this node are as follows:

- IndexBased: *True* if the deposit conventions are index based and *False* if the conventions are given explicitly.
- Index: The index family from which to imply the conventions for the deposit quote. For example, this could be EUR-EURIBOR, USD-LIBOR etc.

- Calendar: The business day calendar for the deposit quote.
- Convention: The roll convention for the deposit quote.
- EOM: *True* if the end of month roll convention is to be used for the deposit quote and *False* if not.
- DayCounter: The day count basis associated with the deposit quote.

7.8.3 Future Conventions

A node with name *Future* is used to store conventions for IMM Future quotes. The structure of this node is shown in Listing 67. The only piece of information needed is the underlying money market index name and this is given in the `Index` node. For example, this could be EUR-EURIBOR-3M, USD-LIBOR-3M etc.

Listing 67: Future conventions

```
<Future>
  <Id> </Id>
  <Index> </Index>
</Future>
```

7.8.4 FRA Conventions

A node with name *FRA* is used to store conventions for FRA quotes. The structure of this node is shown in Listing 68. The only piece of information needed is the underlying index name and this is given in the `Index` node. For example, this could be EUR-EURIBOR-6M, CHF-LIBOR-6M etc.

Listing 68: FRA conventions

```
<FRA>
  <Id> </Id>
  <Index> </Index>
</FRA>
```

7.8.5 OIS Conventions

A node with name *OIS* is used to store conventions for Overnight Indexed Swap (OIS) quotes. The structure of this node is shown in Listing 69.

Listing 69: OIS conventions

```
<OIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <Index> </Index>
  <FixedDayCounter> </FixedDayCounter>
  <PaymentLag> </PaymentLag>
  <EOM> </EOM>
  <FixedFrequency> </FixedFrequency>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <Rule> </Rule>
  <PaymentCalendar> </PaymentCalendar>
</OIS>
```

The meanings of the various elements in this node are as follows:

- SpotLag: The number of business days until the start of the OIS.
- Index: The name of the overnight index. For example, this could be EUR-EONIA, USD-FedFunds etc.
- FixedDayCounter: The day count basis on the fixed leg of the OIS.
- PaymentLag [Optional]: The payment lag, as a number of business days, on both legs. If not provided, this defaults to 0.
- EOM [Optional]: *True* if the end of month roll convention is to be used when generating the OIS schedule and *False* if not. If not provided, this defaults to *False*.
- FixedFrequency [Optional]: The frequency of payments on the fixed leg. If not provided, this defaults to *Annual*.
- FixedConvention [Optional]: The roll convention for accruals on the fixed leg. If not provided, this defaults to *Following*.
- FixedPaymentConvention [Optional]: The roll convention for payments on the fixed leg. If not provided, this defaults to *Following*.
- Rule [Optional]: The rule used for generating the OIS dates schedule i.e. *Backward* or *Forward*. If not provided, this defaults to *Backward*.
- PaymentCalendar [Optional]: The business day calendar used for determining coupon payment dates. If not specified, this defaults to the fixing calendar defined on the overnight index.

7.8.6 Swap Conventions

A node with name *Swap* is used to store conventions for vanilla interest rate swap (IRS) quotes. The structure of this node is shown in Listing 70.

Listing 70: Swap conventions

```
<Swap>
  <Id> </Id>
  <FixedCalendar> </FixedCalendar>
  <FixedFrequency> </FixedFrequency>
  <FixedConvention> </FixedConvention>
  <FixedDayCounter> </FixedDayCounter>
  <Index> </Index>
  <FloatFrequency> </FloatFrequency>
  <SubPeriodsCouponType> </SubPeriodsCouponType>
</Swap>
```

The meanings of the various elements in this node are as follows:

- FixedCalendar: The business day calendar on the fixed leg.
- FixedFrequency: The frequency of payments on the fixed leg.
- FixedConvention: The roll convention on the fixed leg.
- FixedDayCounter: The day count basis on the fixed leg.
- Index: The Ibor index on the floating leg.
- FloatFrequency [Optional]: The frequency of payments on the floating leg, to be used if the frequency is different to the tenor of the index (e.g. CAD swaps for BA-3M have a 6M or 1Y payment frequency with a Compounding coupon)
- SubPeriodsCouponType [Optional]: Defines how coupon rates should be calculated when the float frequency is different to that of the index. Possible values are "Compounding" and "Averaging".

7.8.7 Average OIS Conventions

A node with name *AverageOIS* is used to store conventions for average OIS quotes. An average OIS is a swap where a fixed rate is swapped against a daily averaged overnight index plus a spread. The structure of this node is shown in Listing 71.

Listing 71: Average OIS conventions

```
<AverageOIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <FixedTenor> </FixedTenor>
  <FixedDayCounter> </FixedDayCounter>
  <FixedCalendar> </FixedCalendar>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <Index> </Index>
  <OnTenor> </OnTenor>
  <RateCutoff> </RateCutoff>
</AverageOIS>
```

The meanings of the various elements in this node are as follows:

- SpotLag: Number of business days until the start of the average OIS.
- FixedTenor: The frequency of payments on the fixed leg.
- FixedDayCounter: The day count basis on the fixed leg.
- FixedCalendar: The business day calendar on the fixed leg.
- FixedFrequency: The frequency of payments on the fixed leg.
- FixedConvention: The roll convention for accruals on the fixed leg.
- FixedPaymentConvention: The roll convention for payments on the fixed leg.
- Index: The name of the overnight index.
- OnTenor: The frequency of payments on the overnight leg.
- RateCutoff: The rate cut-off on the overnight leg. Generally, the overnight fixing is only observed up to a certain number of days before the payment date and the last observed rate is applied for the remaining days in the period. This rate cut-off gives the number of days e.g. 2 for Fed Funds average OIS.

7.8.8 Tenor Basis Swap Conventions

A node with name *TenorBasisSwap* is used to store conventions for tenor basis swap quotes. The structure of this node is shown in Listing 72.

Listing 72: Tenor basis swap conventions

```
<TenorBasisSwap>
  <Id> </Id>
  <LongIndex> </LongIndex>
  <ShortIndex> </ShortIndex>
  <ShortPayTenor> </ShortPayTenor>
  <SpreadOnShort> </SpreadOnShort>
  <IncludeSpread> </IncludeSpread>
  <SubPeriodsCouponType> </SubPeriodsCouponType>
</TenorBasisSwap>
```

The meanings of the various elements in this node are as follows:

- LongIndex: The name of the long tenor Ibor index.
- ShortIndex: The name of the short tenor Ibor index.
- ShortPayTenor [Optional]: The frequency of payments on the short tenor Ibor leg. This is usually the same as the short tenor Ibor index's tenor. However, it can also be longer e.g. USD tenor basis swaps where the short tenor Ibor index is compounded and paid on the same frequency as the long tenor Ibor index. If not provided, this defaults to the short tenor Ibor index's tenor.
- SpreadOnShort [Optional]: *True* if the tenor basis swap quote has the spread on the short tenor Ibor index leg and *False* if not. If not provided, this defaults to *True*.

- IncludeSpread [Optional]: *True* if the tenor basis swap spread is to be included when compounding is performed on the short tenor Ibor index leg and *False* if not. If not provided, this defaults to *False*.

- SubPeriodsCouponType [Optional]: This field can have the value *Compounding* or *Averaging* and it only applies when the frequency of payments on the short tenor Ibor leg does not equal the short tenor Ibor index's tenor. If *Compounding* is specified, then the short tenor Ibor index is compounded and paid on the frequency specified in the `ShortPayTenor` node. If *Averaging* is specified, then the short tenor Ibor index is averaged and paid on the frequency specified in the `ShortPayTenor` node. If not provided, this defaults to *Compounding*.

7.8.9 Tenor Basis Two Swap Conventions

A node with name *TenorBasisTwoSwap* is used to store conventions for tenor basis swap quotes where the quote is the spread between the fair fixed rate on two swaps against Ibor indices of different tenors. We call the swap against the Ibor index of longer tenor the long swap and the remaining swap the short swap. The structure of the tenor basis two swap conventions node is shown in Listing 73.

Listing 73: Tenor basis two swap conventions

```xml
<TenorBasisTwoSwap>
  <Id> </Id>
  <Calendar> </Calendar>
  <LongFixedFrequency> </LongFixedFrequency>
  <LongFixedConvention> </LongFixedConvention>
  <LongFixedDayCounter> </LongFixedDayCounter>
  <LongIndex> </LongIndex>
  <ShortFixedFrequency> </ShortFixedFrequency>
  <ShortFixedConvention> </ShortFixedConvention>
  <ShortFixedDayCounter> </ShortFixedDayCounter>
  <ShortIndex> </ShortIndex>
  <LongMinusShort> </LongMinusShort>
</TenorBasisTwoSwap>
```

The meanings of the various elements in this node are as follows:

- Calendar: The business day calendar on both swaps.
- LongFixedFrequency: The frequency of payments on the fixed leg of the long swap.
- LongFixedConvention: The roll convention on the fixed leg of the long swap.
- LongFixedDayCounter: The day count basis on the fixed leg of the long swap.
- LongIndex: The Ibor index on the floating leg of the long swap.
- ShortFixedFrequency: The frequency of payments on the fixed leg of the short swap.
- ShortFixedConvention: The roll convention on the fixed leg of the short swap.
- ShortFixedDayCounter: The day count basis on the fixed leg of the short swap.

- ShortIndex: The Ibor index on the floating leg of the short swap.

- LongMinusShort [Optional]: *True* if the basis swap spread is to be interpreted as the fair rate on the long swap minus the fair rate on the short swap and *False* if the basis swap spread is to be interpreted as the fair rate on the short swap minus the fair rate on the long swap. If not provided, it defaults to *True*.

7.8.10 FX Conventions

A node with name *FX* is used to store conventions for FX spot and forward quotes for a given currency pair. The structure of this node is shown in Listing 74.

Listing 74: FX conventions

```
<FX>
  <Id> </Id>
  <SpotDays> </SpotDays>
  <SourceCurrency> </SourceCurrency>
  <TargetCurrency> </TargetCurrency>
  <PointsFactor> </PointsFactor>
  <AdvanceCalendar> </AdvanceCalendar>
  <SpotRelative> </SpotRelative>
  <AdditionalSettleCalendar> </AdditionalSettleCalendar>
</FX>
```

The meanings of the various elements in this node are as follows:

- SpotDays: The number of business days to spot for the currency pair.

- SourceCurrency: The source currency of the currency pair. The FX quote is assumed to give the number of units of target currency per unit of source currency.

- TargetCurrency: The target currency of the currency pair.

- PointsFactor: The number by which a points quote for the currency pair should be divided before adding it to the spot quote to obtain the forward rate.

- AdvanceCalendar [Optional]: The business day calendar(s) used for advancing dates for both spot and forwards. If not provided, it defaults to a calendar with no holidays.

- SpotRelative [Optional]: *True* if the forward tenor is to be interpreted as being relative to the spot date. *False* if the forward tenor is to be interpreted as being relative to the valuation date. If not provided, it defaults to *True*.

- AdditionalSettleCalendar [Optional]: In some cases, when the spot date is calculated using the values in the `AdvanceCalendar` and `SpotDays` nodes, it is checked against an additional settlement calendar(s) and if it is not a good business day then it is moved forward until it is a good business day on both the additional settlement calendar(s) and the AdvanceCalendar. This additional settlement calendar(s) can be specified here. If not provided, it defaults to a calendar with no holidays.

7.8.11 Cross Currency Basis Swap Conventions

A node with name *CrossCurrencyBasis* is used to store conventions for cross currency basis swap quotes. The structure of this node is shown in Listing 75.

Listing 75: Cross currency basis swap conventions

```
<CrossCurrencyBasis>
  <Id> </Id>
  <SettlementDays> </SettlementDays>
  <SettlementCalendar> </SettlementCalendar>
  <RollConvention> </RollConvention>
  <FlatIndex> </FlatIndex>
  <SpreadIndex> </SpreadIndex>
  <EOM> </EOM>
  <IsResettable> </IsResettable>
  <FlatIndexIsResettable> </FlatIndexIsResettable>>
</CrossCurrencyBasis>
```

The meanings of the various elements in this node are as follows:

- SettlementDays: The number of business days to the start of the cross currency basis swap.
- SettlementCalendar: The business day calendar(s) for both legs and to arrive at the settlement date using the SettlementDays above.
- RollConvention: The roll convention for both legs.
- FlatIndex: The name of the index on the leg that does not have the cross currency basis spread.
- SpreadIndex: The name of the index on the leg that has the cross currency basis spread.
- EOM [Optional]: *True* if the end of month convention is to be used when generating the schedule on both legs, and *False* if not. If not provided, it defaults to *False*.
- IsResettable [Optional]: *True* if the swap is mark-to-market resetting, and *False* otherwise. If not provided, it defaults to *False*.
- FlatIndexIsResettable [Optional]: *True* if it is the notional on the leg paying the flat index that resets, and *False* otherwise. If not provided, it defaults to *True*.

7.8.12 Inflation Conventions

A node with name *InflationSwap* is used to store conventions for zero or year on year inflation swap quotes. The structure of this node is shown in Listing 77

Listing 76: Inflation swap conventions

```
<InflationSwap>
    <Id>EUHICPXT_INFLATIONSWAP</Id>
    <FixCalendar>TARGET</FixCalendar>
    <FixConvention>MF</FixConvention>
    <DayCounter>30/360</DayCounter>
    <Index>EUHICPXT</Index>
    <Interpolated>false</Interpolated>
    <ObservationLag>3M</ObservationLag>
    <AdjustInflationObservationDates>false</AdjustInflationObservationDates>
    <InflationCalendar>TARGET</InflationCalendar>
    <InflationConvention>MF</InflationConvention>
</InflationSwap>
```

The meaning of the elements is as follows:

- FixCalendar: The calendar for the fixed rate leg of the swap.
- FixConvention: The rolling convention for the fixed rate leg of the swap.
- DayCounter: The payoff / coupon day counter (applied to both legs).
- Index: The underlying inflation index.
- Interpolated: Flag indicating interpolation of the index in the swap's payoff calculation.
- ObservationLag: The index observation lag to be applied.
- AdjustInfObsDates: Flag indicating whether index observation dates should be adjusted or not.
- InfCalendar: The calendar for the inflation leg of the swap.
- InfConvention: The rolling convention for the inflation leg of the swap.

7.8.13 CMS Spread Option Conventions

A node with name *InflationSwap* is used to store conventions for zero or year on year inflation swap quotes. The structure of this node is shown in Listing 77

Listing 77: Inflation swap conventions

```
<CmsSpreadOption>
  <Id>EUR-CMS-10Y-2Y-CONVENTION</Id>
  <ForwardStart>0M</ForwardStart>
  <SpotDays>2D</SpotDays>
  <SwapTenor>3M</SwapTenor>
  <FixingDays>2</FixingDays>
  <Calendar>TARGET</Calendar>
  <DayCounter>A360</DayCounter>
  <RollConvention>MF</RollConvention>
</CmsSpreadOption>
```

The meaning of the elements is as follows:

- ForwardStart: The calendar for the fixed rate leg of the swap.
- SpotDays: The number of business days to spot for the CMS Spread Index.
- SwapTenor: The frequency of payments on the CMS Spread leg.
- FixingDays: The number of fixing days.
- Calendar: The calendar for the CMS Spread leg.
- DayCounter: The day counter for the CMS Spread leg.
- RollConvention: The rolling convention for the CMS Spread Leg.

8 Trade Data

The trades that make up the portfolio are specified in an XML file where the portfolio data is specified in a hierarchy of nodes and sub-nodes. The nodes containing individual trade data are referred to as elements or XML elements. These are generally the lowest level nodes.

The top level portfolio node is delimited by an opening `<Portfolio>` and a closing `</Portfolio>` tag. Within the portfolio node, each trade is defined by a starting `<Trade id="[Tradeid]">` and a closing `</Trade>` tag. Further, the trade type is set by the TradeType XML element. Each trade has an Envelope node that includes the same XML elements for all trade types (Id, Type, Counterparty, Rating, NettingSetId) plus the Additional fields node, and after that, a node containing trade specific data.

An example of a `portfolio.xml` file with one Swap trade including the full envelope node is shown in Listing 78.

Listing 78: Portfolio

```
<Portfolio>
  <Trade id="Swap#1">
    <TradeType> Swap </TradeType>
    <Envelope>
      <CounterParty> Counterparty#1 </CounterParty>
      <NettingSetId> NettingSet#2 </NettingSetId>
      <PortfolioIds>
          <PortoliodId> PF#1 </PortfolioId>
          <PortoliodId> PF#2 </PortfolioId>
      </PortfolioIds>
      <AdditionalFields>
        <Sector> SectorA </Sector>
        <Book> BookB </Book>
        <Rating> A1 </Rating>
      </AdditionalFields>
    </Envelope>
    <SwapData>
        ...
        [Trade specific data for a Swap]
        ...
    </SwapData>
  </Trade>
</Portfolio>
```

A description of all portfolio data, i.e. of each node and XML element in the portfolio file, with examples and allowable values follows below. There is only one XML elements directly under the top level `Portfolio` node:

- `TradeType`: ORE currently supports 14 trade types.

  Allowable values: *ForwardRateAgreement, Swap, CapFloor, Swaption, FxForward, FxSwap, FxOption, EquityForward, EquityOption, VarianceSwap, CommodityForward, CommodityOption, CreditDefaultSwap, Bond*

8.1 Envelope

The envelope node contains basic identifying details of a trade (`Id`, `Type`, `Counterparty`, `NettingSetId`), a `PortfolioIds` node containing a list of portfolio assignments, plus an `AdditionalFields` node where custom elements can be added for informational purposes such as `Book` or `Sector`. Beside the custom elements within the `AdditionalFields` node, the envelope contains the same elements for all Trade types. The `Id`, `Type`, `Counterparty` and `NettingSetId` elements must have non-blank entries for ORE to run. The meanings and allowable values of the various elements in the `Envelope` node follow below.

- `Id`: The `Id` element in the envelope is used to identify trades within a portfolio. It should be set to identical values as the `Trade id=" "` element.

  Allowable values: Any alphanumeric string. The underscore (_) sign may be used as well.

- `Counterparty`: Specifies the name of the counterparty of the trade. It is used to show exposure analytics by counterparty.

  Allowable values: Any alphanumeric string. Underscores (_) and blank spaces may be used as well.

- `NettingSetId` [Optional]: The `NettingSetId` element specifies the identifier for a netting set. If a `NettingSetId` is specified, the trade is eligible for close-out netting under the terms of an associated ISDA agreement. The specified `NettingSetId` must be defined within the netting set definitions file (see section 9). If left blank or omitted the trade will not belong to any netting set, and thus not be eligible for netting.

  Allowable values: Any alphanumeric string. Underscores (_) and blank spaces may be used as well.

- `PortfolioIds` [Optional]: The PortfolioIds node allows the assignment of a given trade to several portfolios, each enclosed in its own pair of tags `<PortfolioId>` and `</PortfolioId>` . Note that ORE does not assume a hierarchical organisation of such portfolios. If present, the portfolio IDs will be used in the generation of some ORE reports such as the VaR report which provides breakdown by any portfolio id that occurs in the trades' envelopes.

  Allowable values for each PortfolioId: Any string.

- `AdditionalFields` [Optional]: The AdditionalFields node allows the insertion of additional trade information using custom XML elements. For example, elements such as Sector, Desk or Folder can be used. The elements within the `AdditionalFields` node are used for informational purposes only, and do not affect any analytics in ORE.

  Allowable values: Any custom element.

8.2 Trade Specific Data

After the envelope node, trade-specific data for each trade type supported by ORE is included. Each trade type has its own trade data container which is defined by an XML node containing a trade-specific configuration of individual XML tags - called elements - and trade components. The trade components are defined by XML sub-nodes that can be used within multiple trade data containers, i.e. by multiple trade types.

Details of trade-specific data for all trade types follow below.

8.2.1 Swap

The `SwapData` node is the trade data container for the Swap trade type. A Swap must have at least one leg, and can have an unlimited number of legs. Each leg is represented by a `LegData` trade component sub-node, described in section 8.3.2. An example structure of a two-legged `SwapData` node is shown in Listing 79.

Listing 79: Swap data

```
<SwapData>
  <LegData>
    ...
  </LegData>
  <LegData>
    ...
  </LegData>
</SwapData>
```

8.2.2 Cap/Floor

The `CapFloorData` node is the trade data container for trade type CapFloors. It's a cap, floor or collar (i.e. a portfolio of a long cap and a short floor for a long position in the collar) on a series of Ibor or CMS rates. The `CapFloorData` node contains a `LongShort` sub-node which indicates whether the cap (floor, collar) is long or short, and a `LegData` sub-node where the LegType element must be set to *Floating* or *CMS*, plus elements for the Cap and Floor rates. An example structure with Cap rates is shown in in Listing 80. A `CapFloorData` node must have either `Caps` or `Floors` elements, or both.

Listing 80: Cap/Floor data

```
<CapFloorData>
  <LongShort>Long</LongShort>
  <LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
     ...
  </LegData>
  <Caps>
    <Cap>0.05</Cap>
  </Caps>
</CapFloorData>
```

The meanings and allowable values of the elements in the `CapFloorData` node follow below.

- LongShort: This node defines the position in the cap (floor, collar) and can take values *Long* or *Short*.

- LegData: This is a trade component sub-node outlined in section 8.3.2. Exactly one `LegData` node is allowed and the LegType element must be set to *Floating* or *CMS*.

- Caps: This node has child elements of type `Cap` capping the floating leg. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed cap rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Cap` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 5%

- Floors: This node has child elements of type `Floor` flooring the floating leg. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed floor rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Floor` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 5%

8.2.3 Forward Rate Agreement

A forward rate agreement is set up using a `ForwardRateAgreementData` block as shown in listing 81. The forward rate agreement specific elements are:

- StartDate: A FRA expires/settles on the startDate.
  Allowable values: See `Date` in Table 11.

- EndDate: EndDate is the date when the forward loan or deposit ends. It follows that (EndDate - StartDate) is the tenor/term of the underlying loan or deposit.

  Allowable values: See `Date` in Table 11.

- Currency: The currency of the FRA notional.
  Allowable values: See `Currency` in Table 11.

- Index: The name of the interest rate index the FRA is benchmarked against.

  Allowable values: An alphanumeric string of the form CCY-INDEX-TERM. CCY, INDEX and TERM must be separated by dashes (-). CCY and INDEX must be among the supported currency and index combinations. TERM must be an integer followed by D, W, M or Y. See Table 14.

- LongShort: Specifies whether the FRA position is long (one receives the agreed rate) or short (one pays the agreed rate).

  Allowable values: *Long*, *Short*.

- Strike: The agreed forward interest rate.

  Allowable values: Any real number. The strike rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.

- Notional: No accretion or amortisation, just a constant notional.
  Allowable values: Any positive real number.

Listing 81: Forward Rate Agreement Data

```
<ForwardRateAgreementData>
    <StartDate>20161028</StartDate>
    <EndDate>20351028</EndDate>
    <Currency>EUR</Currency>
    <Index>EUR-EURIBOR-6M</Index>
    <LongShort>Long</LongShort>
    <Strike>0.001</Strike>
    <Notional>1000000000</Notional>
</ForwardRateAgreementData>
```

8.2.4 Swaption

The `SwaptionData` node is the trade data container for the Swaption trade type. The `SwaptionData` node has one and exactly one `OptionData` trade component sub-node, and at least one `LegData` trade component sub-node. These trade components are outlined in section 8.3.1 and section 8.3.2.

Supported swaption exercise styles are European and Bermudan. Swaptions of both exercise styles must have two legs, with each leg represented by a `LegData` sub-node. Cross currency swaptions are not supported for either exercise style, i.e. the Currency element must have the same value for all `LegData` sub-nodes of a swaption. See Table 8 for further details on requirements for swaptions.

The structure of an example `SwaptionData` node of a European swaption is shown in Listing 82.

Listing 82: Swaption data

```
<SwaptionData>
    <OptionData>
        <Style>European</Style>
        ...
    </OptionData>
    <LegData>
        <Currency>GBP</Currency>
        ...
    </LegData>
    <LegData>
        <Currency>GBP</Currency>
        ...
    </LegData>
</SwaptionData>
```

	A Swaption requires:
`OptionData`	One `OptionData` sub-node
`Style`	*Bermudan* or *European*
`ExerciseDates`	*Bermudan* swaptions require at least two `ExerciseDate` child elements. *European* swaptions can only have one `ExerciseDate` child element.
`LegData`	Two `LegData` sub-nodes
`Currency`	The same currency for both nodes.
`LegType`	*Bermudan* swaptions must have: *Fixed* on one node and *Floating* on the other. No such requirement for *European* swaptions.

Table 8: Requirements for Swaptions

8.2.5 FX Forward

The `FXForwardData` node is the trade data container for the FX Forward trade type. The structure - including example values - of the `FXForwardData` node is shown in Listing 83. It contains no sub-nodes.

Listing 83: FX Forward data

```
<FxForwardData>
    <ValueDate>2023-04-09</ValueDate>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1500000</SoldAmount>
</FxForwardData>
```

The meanings and allowable values of the various elements in the `FXForwardData` node follow below. All elements are required.

- ValueDate: The value date of the FX Forward.
  Allowable values: See `Date` in Table 11.
- BoughtCurrency: The currency to be bought on value date.
  Allowable values: See `Currency` in Table 11.
- BoughtAmount: The amount to be bought on value date.
  Allowable values: Any positive real number.
- SoldCurrency: The currency to be sold on value date.
  Allowable values: See `Currency` in Table 11.
- SoldAmount: The amount to be sold on value date.
  Allowable values: Any positive real number.
- Settlement [Optional]: Delivery type. Note that Non-Deliverable Forwards can be represented by *Cash* settlement. Delivery type does not impact pricing in ORE.

  Allowable values: *Cash* or *Physical*. Defaults to *Physical* if left blank or omitted.

8.2.6 FX Swap

The `FXSwapData` node is the trade data container for the FX Swap trade type. The structure - including example values - of the `FXSwapData` node is shown in Listing 84. It contains no sub-nodes.

Listing 84: FX Swap data

```
<FxSwapData>
    <NearDate>2018-09-01</NearDate>
    <NearBoughtCurrency>EUR</NearBoughtCurrency>
    <NearBoughtAmount>1000000</NearBoughtAmount>
    <NearSoldCurrency>USD</NearSoldCurrency>
    <NearSoldAmount>1140000</NearSoldAmount>
    <FarDate>2028-09-01</FarDate>
    <FarBoughtAmount>1300000</FarBoughtAmount>
    <FarSoldAmount>1000000</FarSoldAmount>
</FxSwapData>
```

The meanings and allowable values of the various elements in the `FXSwapData` node follow below. All elements are required.

- NearDate: The date of the initial fx exchange of the FX Swap.
  Allowable values: See `Date` in Table 11.
- NearBoughtCurrency: The currency to be bought in the initial exchange at near date, and sold in the final exchange at far date.
  Allowable values: See `Currency` in Table 11.
- NearBoughtAmount: The amount to be bought on near date.
  Allowable values: Any positive real number.
- NearSoldCurrency: The currency to be sold in the initial fx exchange at near date, and bought in the final exchange at far date.
  Allowable values: See `Currency` in Table 11.
- NearSoldAmount: The amount to be sold on near date.
  Allowable values: Any positive real number.
- FarDate: The date of the final fx exchange of the FX Swap.
  Allowable values: Any date further into the future than NearDate. See `Date` in Table 11.
- FarBoughtAmount: The amount to be bought on far date.
  Allowable values: Any positive real number.
- FarSoldAmount: The amount to be sold on far date.
  Allowable values: Any positive real number.

8.2.7 FX Option

The `FXOptionData` node is the trade data container for the FX Option trade type. Vanilla FX options are supported, the exercise style must be *European*. The strike rate of an FX option is: SoldAmount / BoughtAmount. The `FXOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the FX Option. The structure of an example `FXOptionData` node for a FX Option is shown in Listing 85.

Listing 85: FX Option data

```
<FxOptionData>
    <OptionData>
     ...
    </OptionData>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1350000</SoldAmount>
</FxOptionData>
```

The meanings and allowable values of the elements in the `FXOptionData` node follow below.

- OptionData: This is a trade component sub-node outlined in section 8.3.1. Note that the FX option type allows for *European* option style only.
- BoughtCurrency: The bought currency of the FX option.

  Allowable values: See Currency in Table 11.
- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.
- SoldCurrency: The sold currency of the FX option.

  Allowable values: See Currency in Table 11.
- SoldAmount: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

8.2.8 Equity Option

The `EquityOptionData` node is the trade data container for the equity option trade type. Vanilla equity options are supported, the exercise style must be *European*. The `EquityOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the equity option. The structure of an example `EquityOptionData` node for an equity option is shown in Listing 86.

Listing 86: Equity Option data

```
<EquityOptionData>
    <OptionData>
      ...
    </OptionData>
    <Name>ISIN:US78378X1072</Name>
    <Currency>USD</Currency>
    <Strike>2147.56</Strike>
    <Quantity>17000</Quantity>
</EquityOptionData>
```

The meanings and allowable values of the elements in the `EquityOptionData` node follow below.

- OptionData: This is a trade component sub-node outlined in section 8.3.1 Option Data. Note that the equity option type allows for *European* option style only.
- Name: The identifier of the underlying equity or equity index.
  Allowable values: Any string (provided it is the ID of an equity in the market configuration). Typically an ISIN-code with the *ISIN:* prefix.
- Currency: The currency of the equity option.
  Allowable values: See `Currency` in Table 11.
- Strike: The option strike price.
  Allowable values: Any positive real number.
- Quantity: The number of units of the underlying covered by the transaction.
  Allowable values: Any positive real number.

8.2.9 Equity Forward

The `EquityForwardData` node is the trade data container for the equity forward trade type. Vanilla equity forwards are supported. The structure of an example `EquityForwardData` node for an equity option is shown in Listing 87.

Listing 87: Equity Forward data

```
<EquityForwardData>
  <LongShort>Long</LongShort>
  <Maturity>2018-06-30</Maturity>
  <Name>ISIN:US78378X1072</Name>
  <Currency>USD</Currency>
  <Strike>2147.56</Strike>
  <Quantity>17000</Quantity>
</EquityForwardData>
```

The meanings and allowable values of the elements in the `EquityForwardData` node follow below.

- LongShort: Defines whether the underlying equity will be bought (long) or sold (short).
  Allowable values: *Long*, *Short*.

- Maturity: The maturity date of the forward contract, i.e. the date when the underlying equity will be bought/sold.
  Allowable values: Any date string, see `Date` in Table 11.
- Name: The identifier of the underlying equity or equity index.

  Allowable values: Any string (provided it is the ID of an equity in the market configuration). Typically an ISIN-code with the *ISIN:* prefix.
- Currency: The currency of the equity forward.
  Allowable values: See `Currency` in Table 11.
- Strike: The agreed buy/sell price of the equity forward.
  Allowable values: Any positive real number.
- Quantity: The number of units of the underlying equity to be bought/sold.
  Allowable values: Any positive real number.

8.2.10 Equity Swap

An Equity Swap is set up as a swap using the `SwapData` node, with one leg of type *Equity* and one more leg that can be either *Fixed* or *Floating*. Listing 88 shows an example. The Equity leg contains an additional `EquityLegData` block. See 8.3.10 for details on the Equity leg specification.

Listing 88: Equity Swap Data

```
<SwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Payer>true</Payer>
    ...
  </LegData>
  <LegData>
    <LegType>Equity</LegType>
    <Payer>false</Payer>
    ...
    <EquityLegData>
    ...
    </EquityLegData>
  </LegData>
</SwapData>
```

8.2.11 CPI Swap

A CPI swap is set up as a swap, with one leg of type `CPI`. Listing 89 shows an example. The CPI leg contains an additional `CPILegData` block. See 8.3.11 for details on the CPI leg specification.

Listing 89: CPI Swap Data

```
<SwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Payer>true</Payer>
    ...
  </LegData>
  <LegData>
    <LegType>CPI</LegType>
    <Payer>false</Payer>
    ...
    <CPILegData>
    ...
    </CPILegData>
  </LegData>
</SwapData>
```

8.2.12 Year on Year Inflation Swap

A Year on Year inflation swap is set up as a swap, with one leg of type `YY`. Listing 90 shows an example. The YY leg contains an additional `YYLegData` block. See 8.3.12 for details on the YY leg specification.

Listing 90: Year on Year Swap Data

```
<SwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Payer>true</Payer>
    ...
  </LegData>
  <LegData>
    <LegType>YY</LegType>
    <Payer>false</Payer>
    ...
    <YYLegData>
    ...
    </YYLegData>
  </LegData>
</SwapData>
```

8.2.13 Bond

A vanilla Bond is set up using a `BondData` block as shown in listing 91. The bond specific elements are

- IssuerId: A unique identifier for the issuer of the bond.
- CreditCurveId: The identifier of the default curve used for pricing, which must match a default curve id in the market data configuration.
- SecurityId: A unique identifier for the security, this defines the security specific spread to be used for pricing.

- ReferenceCurveId: The benchmark curve to be used for pricing, this must match a name of a yield curve in the market data configuration.
- SettlementDays: The settlement delay applicable to the security.
- Calendar: The calendar associated to the settlement lag.
- IssueDate: The issue date of the security.

A LegData block then defines the cashflow structure of the bond, this can be of type fixed, floating etc.

Listing 91: Bond Data

```
<BondData>
    <IssuerId>CPTY_C</IssuerId>
    <CreditCurveId>ISIN:XS0982710740</CreditCurveId>
    <SecurityId>SECURITY_1</SecurityId>
    <ReferenceCurveId>EUR-EURIBOR-6M</ReferenceCurveId>
    <SettlementDays>2</SettlementDays>
    <Calendar>TARGET</Calendar>
    <IssueDate>20160203</IssueDate>
    <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        ...
    </LegData>
</BondData>
```

The bond pricing requires a recovery rate that can be specified per SecurityId in the market data configuration.

8.2.14 Credit Default Swap

A credit default swap is set up using a `CreditDefaultSwapData` block as shown in listing 92. The CDS specific elements are

- IssuerId: An identifier for the reference entity of the CDS. For informational purposes and not used for pricing.
- CreditCurveId: Typically the ISIN-code of the reference entity defining the default curve used for pricing. Other identifiers may be used as well, provided they are supported in the market data configuration.
- SettlesAccrual: Whether or not the accrued coupon is due in the event of a default.
- PaysAtDefaultTime: If set to true, any payments triggered by a default event are due at default time. If set to false, they are due at the end of the accrual period.
- ProtectionStart: The first date where a default event will trigger the contract.
- UpfrontDate [Optional]: Settlement date for the upfront payment.
- UpfrontFee [Optional]: The upfront payment, expressed as a rate, to be multiplied by notional amount.

A LegData block then defines the cashflow structure of the credit default swap, this must be be of type *Fixed*.

Listing 92: CreditDefaultSwap Data

```
<CreditDefaultSwapData>
  <IssuerId>CPTY_A</IssuerId>
  <CreditCurveId>ISIN:XS0982710740</CreditCurveId>
  <SettlesAccrual>Y</SettlesAccrual>
  <PaysAtDefaultTime>Y</PaysAtDefaultTime>
  <ProtectionStart>20160206</ProtectionStart>
  <UpfrontDate>20160208</UpfrontDate>
  <UpfrontFee>0.0</UpfrontFee>
  <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        ...
  </LegData>
</CreditDefaultSwapData>
```

8.2.15 Commodity Option

The `CommodityOptionData` node is the trade data container for the commodity option trade type. Vanilla commodity options are supported, the exercise style must be *European*. The `CommodityOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the commodity option. The structure of an example `CommodityOptionData` node for a commodity option is shown in Listing 93.

Listing 93: Commodity Option data

```
<CommodityOptionData>
    <OptionData>
      ...
    </OptionData>
    <Name>COMDTY_WTI_USD</Name>
    <Currency>USD</Currency>
    <Strike>70</Strike>
    <Quantity>500000</Quantity>
</CommodityOptionData>
```

The meanings and allowable values of the elements in the `CommodityOptionData` node follow below.

- OptionData: This is a trade component sub-node outlined in section 8.3.1 Option Data. Note that the commodity option type allows for *European* option style only.
- Name: The name of the underlying commodity.
  Allowable values: Any string (provided it is the ID of a commodity in the market configuration).
- Currency: The currency of the commodity option.
  Allowable values: See `Currency` in Table 11.

- Strike: The option strike price.
  Allowable values: Any positive real number.

- Quantity: The number of units of the underlying commodity covered by the transaction.
  Allowable values: Any positive real number.

8.2.16 Commodity Forward

The `CommodityForwardData` node is the trade data container for the commodity forward trade type. The structure of an example `CommodityForwardData` node for an equity option is shown in Listing 94.

Listing 94: Commodity Forward data

```
<CommodityForwardData>
  <LongShort>Long</LongShort>
  <Maturity>2018-06-30</Maturity>
  <Name>COMDTY_GOLD_USD</Name>
  <Currency>USD</Currency>
  <Strike>1355</Strike>
  <Quantity>1000</Quantity>
</CommodityForwardData>
```

The meanings and allowable values of the elements in the `CommodityForwardData` node follow below.

- LongShort: Defines whether the underlying commodity will be bought (long) or sold (short).
  Allowable values: *Long, Short*

- Maturity: The maturity date of the forward contract, i.e. the date when the underlying commodity will be bought/sold.
  Allowable values: Any date string, see `Date` in Table 11.

- Name: The name of the underlying commodity.
  Allowable values: Any string (provided it is the ID of an equity in the market configuration).

- Currency: The currency of the commodity forward.
  Allowable values: See `Currency` in Table 11.

- Strike: The agreed buy/sell price of the commodity forward.
  Allowable values: Any positive real number.

- Quantity: The number of units of the underlying commodity to be bought/sold.
  Allowable values: Any positive real number.

8.3 Trade Components

Trade components are XML sub-nodes used within the trade data containers to define sets of trade data that more than one trade type can have in common, such as a leg or a schedule. A trade data container can include multiple trade components such as a swap with multiple legs, and a trade component can itself contain further trade components in a nested way.

An example of a `SwapData` trade data container, including two `LegData` trade components which in turn include further trade components such as `FixedLegData`, `ScheduleData` and `FloatingLegData` is shown in Listing 95.

Listing 95: Trade Components Example

```
<SwapData>
    <LegData>
        <Payer>true</Payer>
        <LegType>Fixed</LegType>
        <Currency>EUR</Currency>
        <PaymentConvention>Following</PaymentConvention>
        <DayCounter>30/360</DayCounter>
        <Notionals>
            <Notional>1000000</Notional>
        </Notionals>
        <ScheduleData>
        ...
        </ScheduleData>
        <FixedLegData>
            <Rates>
                <Rate>0.035</Rate>
            </Rates>
        </FixedLegData>
    </LegData>
    <LegData>
        ...
        <ScheduleData>
            ...
        </ScheduleData>
        <FloatingLegData>
            ...
        </FloatingLegData>
    </LegData>
</SwapData>
```

Descriptions of all trade components supported in ORE follow below.

8.3.1 Option Data

This trade component node is used within the `SwaptionData` and `FXOptionData` trade data containers. It contains the `ExerciseDates` sub-node which includes `ExerciseDate` child elements. An example structure of the `OptionData` trade component node is shown in Listing 96.

Listing 96: Option data

```xml
<OptionData>
    <LongShort>Long</LongShort>
    <OptionType>Call</OptionType>
    <Style>Bermudan</Style>
    <Settlement>Cash</Settlement>
    <SettlementMethod>CollateralizedCashPrice</SettlementMethod>
    <PayOffAtExpiry>true</PayOffAtExpiry>
    <ExerciseDates>
        <ExerciseDate>2016-04-20</ExerciseDate>
        <ExerciseDate>2017-04-20</ExerciseDate>
    </ExerciseDates>
</OptionData>
```

The meanings and allowable values of the elements in the `OptionData` node follow below.

- LongShort: Specifies whether the option position is long or short.

  Allowable values: *Long, LONG, long, L* or *Short, SHORT, short, S*

- OptionType: Specifies whether it is a call or a put option.

  Allowable values: *Call* or *Put*

  The meaning of Call and Put values depend on the asset class of the option, see Table 9.

Asset Class	Call / Put Specifications
Equity/ Commodity Option	*Call*: The right to buy the underlying equity/commodity at the strike price. *Put*: The right to sell the underlying equity/commodity at the strike price.
IR Swaption	*Call/Put* values are ignored. Payer/Receiver swaption is determined in the Leg Data nodes of the underlying swap.
FX Option	*Call*: Bought and Sold currencies/amounts stay as determined in the trade data node. *Put*: Bought and Sold currencies/amounts are switched compared to the trade data node.

Table 9: Specification of Option Type Call / Put

- Style: The exercise style of the option.

  Allowable values: *European* or *American* or *Bermudan*. Note that trade type Swaption can have style *European* or *Bermudan*, but not *American*. The FX Option trade type can have style *European* or *American*, but not *Bermudan*.

- Settlement: Delivery type.

  Allowable values: *Cash* or *Physical*

- SettlementMethod [Optional]: Specifies the method to calculate the settlement amount for swaptions.

  Allowable values: *PhysicalOTC*, *PhysicalCleared*, *CollateralizedCashPrice*, *ParYieldCurve*.

  Defaults to *ParYieldCurve* if Settlement is *Cash* and defaults to *PhysicalOTC* if Settlement is *Physical*.

  PhysicalOTC = OTC traded swaptions with physical settlement
  PhysicalCleared = Cleared swaptions with physical settlement
  CollateralizedCashPrice = Cash settled swaptions with settlement price calculation using zero coupon curve discounting
  ParYieldCurve = Cash settled swaptions with settlement price calculation using par yield discounting [7] [8]

- PayOffAtExpiry [Optional]: Relevant for options with early exercise, i.e. the exercise occurs before expiry; *true* indicates payoff at expiry, whereas *false* indicates payoff at exercise. Defaults to *false* if left blank or omitted.

  Allowable values: *true*, *false*

- ExerciseDates: This node contains child elements of type `ExerciseDate`. Options of style *European* or *American* require a single exercise date expressed by one single `ExerciseDate` child element. *Bermudan* style options must have two or more `ExerciseDate` child elements.

- PremiumAmount [Optional]: Option premium amount paid by the option buyer to the option seller.

  Allowable values: Any positive real number.

- PremiumCurrency [Optional]: Currency of the option premium.

  Allowable values: See `Currency` in Table 11.

- PremiumPayDate [Optional]: Date of the option premium payment.

  Allowable values: See `Date` in Table 11.

8.3.2 Leg Data and Notionals

The `LegData` trade component node is used within the `CapFloorData`, `SwapData` and `SwaptionData` trade data containers. It contains a `ScheduleData` trade component sub-node, and a sub-node that depends on the value of the `LegType` element, e.g.: `FixedLegData` for `LegType` *Fixed* or `FloatingLegData` for `LegType` *Floating*. The `LegData` node also includes a `Notionals` sub-node with `Notional` child elements described below. An example structure of a `LegData` node of `LegType` *Floating* is shown in Listing 97.

[7] https://www.isda.org/book/2006-isda-definitions/
[8] https://www.isda.org/a/TlAEE/Supplement-No-58-to-ISDA-2006-Definitions.pdf

Listing 97: Leg data

```xml
<LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
    <Currency>EUR</Currency>
    <PaymentConvention>Following</PaymentConvention>
    <DayCounter>30/360</DayCounter>
    <Notionals>
        <Notional>1000000</Notional>
    </Notionals>
    <ScheduleData>
        ...
    </ScheduleData>
    <FloatingLegData>
        ...
    </FloatingLegData>
</LegData>
```

The meanings and allowable values of the elements in the `LegData` node follow below.

- LegType: Determines which of the available sub-nodes must be used.

  Allowable values: *Fixed, Floating, Cashflow, CMS, CMSSpread, DigitalCMSSpread, Equity, YY, CPI*

- Payer: The flows of the leg are paid to the counterparty if *true*, and received if *false*.

  Allowable values: *true, false*

- Currency: The currency of the leg.

  Allowable values: See `Currency` in Table 11.

- PaymentConvention: The payment convention of the leg coupons.

  Allowable values: See `Roll Convention` in Table 11.

- PaymentLag [optional]: The payment lag given as as number business days, applies to fixed, Ibor and OIS legs.

  Allowable values: A non-negative whole number, if not given it defaults to zero.

- DayCounter: The day count convention of the leg coupons.

  Allowable values: See `DayCount Convention` in Table 13.

- Notionals: This node contains child elements of type `Notional`. If the notional is fixed over the life of the leg only one notional value should be entered. If the notional is amortising or accreting, this is represented by entering multiple notional values, each represented by a `Notional` child element. The first notional value corresponds to the first coupon, the second notional value corresponds to the second coupon, etc. If the number of coupons exceeds the number of notional values, the notional will be kept flat at the value of last entered notional for the remaining coupons. The number of entered notional values cannot exceed the number of coupons.

Allowable values: Each child element can take any positive real number.

An example of a `Notionals` element for an amortising leg with four coupons is shown in Listing 98.

Listing 98: Notional list

```
<Notionals>
    <Notional>65000000</Notional>
    <Notional>65000000</Notional>
    <Notional>55000000</Notional>
    <Notional>45000000</Notional>
</Notionals>
```

Another allowable specification of the notional schedule is shown in Listing 99.

Listing 99: Notional list with dates

```
<Notionals>
    <Notional>65000000</Notional>
    <Notional startDate='2016-01-02'>65000000</Notional>
    <Notional startDate='2017-01-02'>55000000</Notional>
    <Notional startDate='2021-01-02'>45000000</Notional>
</Notionals>
```

The first notional must not have a start date, it will be associated with the schedule's start, The subsequent notionals can have a start date specified from which date onwards the new notional is applied. This allows specifying notionals only for dates where the notional changes.

In case of exchange of currencies an initial exchange, a final exchange and an amortising exchange can be specified using an `Exchanges` child element with `NotionalInitialExchange`, `NotionalFinalExchange` and `NotionalAmortizingExchange` as subelements, see Listing 100.

Listing 100: Notional list with exchange

```
<Notionals>
    <Notional>65000000</Notional>
    <Exchanges>
      <NotionalInitialExchange>true</NotionalInitialExchange>
      <NotionalFinalExchange>true</NotionalFinalExchange>
      <NotionalAmortizingExchange>true</NotionalAmortizingExchange>
    </Exchanges>
</Notionals>
```

FX Resets, used for Rebalancing Cross-currency swaps, can be specified using an `FXReset` child element with the following subelements: See Listing 101 for an example.

- ForeignCurrency: The foreign currency the notional of the leg resets to.

  Allowable values: See `Currency` in Table 11.

- ForeignAmount: The notional amount in the foreign currency that the notional of the leg resets to.

  Allowable values: Any positive real number.

- FXIndex: A reference to an FX Index source for the FX reset fixing.

  Allowable values: A string on the form FX-SOURCE-CCY1-CCY2.

- FixingDays: The FX fixing lag in business days

  Allowable values: Any integer.

- FixingCalendar[Optional]: The calendar associated with the FX Index.

  Allowable values: See Table 12 Calendar. Defaults to the null calendar if left blank or omitted.

Listing 101: Notional list with fx reset

```
<Currency>USD</Currency>
<Notionals>
    <Notional>65000000</Notional> <!-- in USD -->
    <FXReset>
      <ForeignCurrency> EUR </ForeignCurrency>
      <ForeignAmount> 60000000 </ForeignAmount>
      <FXIndex> FX-ECB-USD-EUR </FXIndex>
      <FixingDays> 2 </FixingDays>
      <FixingCalendar>TARGET</FixingCalendar>
    </FXReset>
</Notionals>
```

After the `Notional` sub-node the `LegData` node includes a `ScheduleData` sub-node, and a sub-node based on the choice of `LegType` as per below:

- ScheduleData: This is a trade component sub-node outlined in section 8.3.3 Schedule Data and Dates.
- FixedLegData: This trade component sub-node is required if `LegType` is set to *fixed* It is outlined in section 8.3.4.
- FloatingLegData: This trade component sub-node is required if `LegType` is set to *floating* It is outlined in section 8.3.5 Floating Leg Data and Spreads.
- CMSLegData: This trade component sub-node is required if `LegType` is set to *CMS*. It is outlined in section 8.3.7.
- CMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *CMSSpread*. It is outlined in section 8.3.8.
- DigitalCMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *DigitalCMSSpread*. It is outlined in section 8.3.9.

- EquityLegData: This trade component sub-node is required if `LegType` is set to *Equity*. It is outlined in section 8.3.10.
- CPILegData: This trade component sub-node is required if `LegType` is set to *CPI*. It is outlined in section 8.3.11.
- YYLegData: This trade component sub-node is required if `LegType` is set to *YY*. It is outlined in section 8.3.12.
- ZeroCouponFixedLegData: This trade component sub-node is required if `LegType` is set to *ZeroCouponFixed*. It is outlined in section 8.3.13.

8.3.3 Schedule Data (Rules and Dates)

The `ScheduleData` trade component node is used within the `LegData` trade component. The Schedule can both be rules based (at least one sub-node Rules exists) and dates based (at least one `Dates` sub-node exists, where the schedule is determined directly by `Date` child elements). In rules based schedules, the schedule dates are generated from a set of rules based on the entries of the sub-node Rules, having the elements StartDate, EndDate, Tenor, Calendar, Convention, TermConvention, and Rule. Example structures of `ScheduleData` nodes based on rules respectively dates are shown in Listing 102 and Listing 103, respectively.

Listing 102: Schedule data, rules based

```xml
<ScheduleData>
      <Rules>
          <StartDate>2013-02-01</StartDate>
          <EndDate>2030-02-01</EndDate>
          <Tenor>1Y</Tenor>
          <Calendar>UK</Calendar>
          <Convention>MF</Convention>
          <TermConvention>MF</TermConvention>
          <Rule>Forward</Rule>
      </Rules>
</ScheduleData>
```

Listing 103: Schedule data, date based

```xml
<ScheduleData>
 <Dates>
  <Calendar>NYB</Calendar>
  <Convention>Following</Convention>
  <Tenor>3M</Tenor>
     <Dates>
         <Date>2012-01-06</Date>
         <Date>2012-04-10</Date>
         <Date>2012-07-06</Date>
         <Date>2012-10-08</Date>
         <Date>2013-01-07</Date>
         <Date>2013-04-08</Date>
     </Dates>
  </Dates>
 </ScheduleData>
```

The ScheduleData section can contain any number and combination of `<Dates>` and `<Rules>` sections. The resulting schedule will then be an ordered concatenation of individual schedules.

The meanings and allowable values of the elements in a `<Rules>` based section of the `ScheduleData` node follow below.

- Rules: a sub-node that determines whether the schedule is set by specifying rules that generate the schedule. If existing, the following entries are required: StartDate, EndDate, Tenor, Calendar, Convention, TermConvention, and Rule. If not existing, the Dates sub-node is required.

- StartDate: The schedule start date.

  Allowable values: See `Date` in Table 11.

- EndDate: The schedule end date.

  Allowable values: See `Date` in Table 11.

- Tenor: The tenor used to generate schedule dates.

  Allowable values: A string where the last character must be D or W or M or Y. The characters before that must be a positive integer.
  D = Day, W = Week, M = Month, Y = Year

- Calendar: The calendar used to generate schedule dates.

  Allowable values: See Table 12 Calendar.

- Convention: Determines the adjustment of the schedule dates with regards to the selected calendar.

  Allowable values: See `Roll Convention` in Table 11.

- TermConvention [Optional]: Determines the adjustment of the final schedule date with regards to the selected calendar.

  Allowable values: See `Roll Convention` in Table 11.

- Rule [Optional]: Rule for the generation of the schedule using given start and end dates, tenor, calendar and business day conventions.

  Allowable values and descriptions: See Table 10 Rule. Defaults to *Forward* if left blank or omitted.

- EndOfMonth [Optional]: Specifies whether the date generation rule is different for end of month.

  Allowable values: *True, False*. Defaults to *False* if left blank or omitted. Must be set to *False* or omitted if the date generation Rule is set to *CDS* or *CDS2015*.

- FirstDate [Optional]: Date for initial stub period.

  Allowable values: See `Date` in Table 11. Incompatible with date generation Rule *CDS* and *CDS2015*.

- LastDate [Optional]: Date for final stub period.

  Allowable values: See `Date` in Table 11.

The meanings and allowable values of the elements in a `<Dates>` based section of the `ScheduleData` node follow below.

- Dates: a sub-node that determines that the schedule is set by specifying schedule dates explicitly.
- Calendar [Optional]: Calendar used to determine payment dates, and also to compute day count fractions for irregular periods when day count convention is ActActISMA and the schedule is dates based.

  Allowable values: See Table 12 Calendar. Defaults to *NullCalendar* if omitted, i.e. no holidays at all, not even on weekends.
- Convention [Optional]: Convention used to compute day count fractions for irregular periods when day count convention is ActActISMA and the schedule is dates based.

  Allowable values: See `Roll Convention` in Table 11. Defaults to *Unadjusted* if omitted.
- Tenor [Optional]: Tenor used to compute day count fractions for irregular periods when day count convention is ActActISMA and the schedule is dates based.

  Allowable values: A string where the last character must be D or W or M or Y. The characters before that must be a positive integer.
  D = Day, W = Week, M = Month, Y = Year

  Defaults to *null* if omitted.
- Dates: This is a sub-sub-node and contains child elements of type `Date`. In this case the schedule dates are determined directly by the `Date` child elements. At least two `Date` child elements must be provided.

  Allowable values: Each `Date` child element can take the allowable values listed in `Date` in Table 11.

`Rule`	
Allowable Values	**Effect**
Backward	Backward from termination date to effective date.
Forward	Forward from effective date to termination date.
Zero	No intermediate dates between effective date and termination date.
ThirdWednesday	All dates but effective date and termination date are taken to be on the third Wednesday of their month (with forward calculation.)
Twentieth	All dates but the effective date are taken to be the twentieth of their month (used for CDS schedules in emerging markets.) The termination date is also modified.
TwentiethIMM	All dates but the effective date are taken to be the twentieth of an IMM month (used for CDS schedules.) The termination date is also modified.
OldCDS	Same as TwentiethIMM with unrestricted date ends and long/short stub coupon period (old CDS convention).
CDS	Credit derivatives standard rule defined in 'Big Bang' changes in 2009.
CDS2015	Credit derivatives standard rule updated in 2015. Same as *CDS* but with termination dates adjusted to 20th June and 20th December.

Table 10: Allowable Values for Rule

8.3.4 Fixed Leg Data and Rates

The `FixedLegData` trade component node is used within the `LegData` trade component when the `LegType` element is set to *Fixed*. The `FixedLegData` node only includes the `Rates` sub-node which contains the rates of the fixed leg as child elements of type `Rate`. An example of a `FixedLegData` node for a fixed leg with constant notional is shown in Listing 104.

Listing 104: Fixed leg data

```xml
<FixedLegData>
      <Rates>
          <Rate>0.05</Rate>
      </Rates>
</FixedLegData>
```

The meanings and allowable values of the elements in the `FixedLegData` node follow below.

- Rates: This node contains child elements of type `Rate`. If the rate is constant over the life of the fixed leg, only one rate value should be entered. If two or more coupons have different rates, multiple rate values are required, each represented by a `Rate` child element. The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. The number of entered rate values cannot exceed the number of coupons.

  Allowable values: Each child element can take any real number. The rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.

  As in the case of notionals, the rate schedule can be specified with dates as shown in Listing 105.

Listing 105: Fixed leg data with 'dated' rates

```xml
<FixedLegData>
      <Rates>
          <Rate>0.05</Rate>
          <Rate startDate='2016-02-04'>0.05</Rate>
          <Rate startDate='2019-02-05'>0.05</Rate>
      </Rates>
</FixedLegData>
```

8.3.5 Floating Leg Data, Spreads, Gearings, Caps and Floors

The `FloatingLegData` trade component node is used within the `LegData` trade component when the `LegType` element is set to *Floating*. It is also used directly within the `CapFloor` trade data container. The `FloatingLegData` node includes elements specific to a floating leg as well as the `Spreads` sub-node which contains the spreads of the floating leg as child elements of type `Spread`.
An example of a `FloatingLegData` node is shown in Listing 106.

Listing 106: Floating leg data

```
<FloatingLegData>
    <Index>USD-LIBOR-3M</Index>
    <IsInArrears>false</IsInArrears>
    <FixingDays>2</FixingDays>
    <Spreads>
        <Spread>0.005</Spread>
    </Spreads>
    <Gearings>
        <Gearing>2.0</Gearing>
    </Gearings>
    <Caps>
        <Cap>0.05</Cap>
    </Caps>
    <Floors>
        <Floor>0.01</Floor>
    </Floors>
    <NakedOption>N</NakedOption>
</FloatingLegData>
```

The meanings and allowable values of the elements in the `FloatingLegData` node follow below.

- Index: The combination of currency, index and term that identifies the relevant fixings and yield curve of the floating leg.

  Allowable values: An alphanumeric string of the form CCY-INDEX-TERM. CCY, INDEX and TERM must be separated by dashes (-). CCY and INDEX must be among the supported currency and index combinations. TERM must be an integer followed by D, W, M or Y. See Table 14.

- IsAveraged [Optional]: For cases where there are multiple index fixings over a period *true* indicates that the average of the fixings is used to calculate the coupon. *false* indicates that the coupon is calculated by compounding the fixings. IsAveraged only applies to Overnight indices and Sub Periods Coupons.

  Allowable values: *true, false*. Defaults to *false* if left blank or omitted.

- HasSubPeriods [Optional]: For cases where several Ibor fixings result in a single payment for a period, e.g. if the Ibor tenor is 3M and the schedule tenor is 6M, two fixings are used to compute the amount of the semiannual coupon payments. *true* indicates that an average (IsAveraged = true) or a compounded (IsAveraged=false) value of the fixings is used to determine the payment rate.

  Allowable values: *true, false*. Defaults to *false* if left blank or omitted.

- IncludeSpread [Optional]: Only applies to Sub Periods Coupons. If *true* the spread is included in the compounding, otherwise it is excluded.

  Allowable values: *true, false*. Defaults to *false* if left blank or omitted.

- IsInArrears [Optional]: *true* indicates that fixing is in arrears, i.e. the fixing gap is calculated in relation to the current period end date.
  false indicates that fixing is in advance, i.e. the fixing gap is calculated in relation to the previous period end date.

Allowable values: *true, false*. Defaults to *false* if left blank or omitted.

- FixingDays [Optional]: This is the fixing gap, i.e. the number of days before the period end date an index fixing is taken.

  Allowable values: Positive integers. Defaults to *0* if left blank or omitted.

- Spreads [Optional]: This node contains child elements of type `Spread`. If the spread is constant over the life of the floating leg, only one spread value should be entered. If two or more coupons have different spreads, multiple spread values are required, each represented by a `Spread` child element. The first spread value corresponds to the first coupon, the second spread value corresponds to the second coupon, etc. If the number of coupons exceeds the number of spread values, the spread will be kept flat at the value of last entered spread for the remaining coupons. The number of entered spread values cannot exceed the number of coupons.

  Allowable values: Each child element can take any real number. The spread is expressed in decimal form, e.g. 0.005 is a spread of 0.5% or 50 bp.

  For the `<Spreads>` section, the same applies as for notionals and rates - a list of changing spreads can be specified without or with individual starte dates as shown in Listing 107.

Listing 107: 'Dated' spreads

```
<Spreads>
    <Spread>0.005</Spread>
    <Spread startDate='2017-03-05'>0.007</Spread>
    <Spread startDate='2019-03-05'>0.009</Spread>
</Spreads>
```

- Gearings [Optional]: This node contains child elements of type `Gearing` indicating that the coupon rate is multiplied by the given factors. The mode of specification is analogous to spreads, see above.

- Caps [Optional]: This node contains child elements of type `Cap` indicating that the coupon rate is capped at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above.

- Floors [Optional]: This node contains child elements of type `Floor` indicating that the coupon rate is floored at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above.

- NakedOption [Optional]: Optional node (defaults to N), if Y the leg represents only the embedded floor, cap or collar. By convention these embedded options are considered long if the leg is a receiver leg, otherwise short.

8.3.6 Leg Data with Amortisation Structures

Amortisation structures can (optionally) be added to a leg as indicated in the following listing 108, within a block of information enclosed by `<Amortizations>` and `</Amortizations>` tags.

Listing 108: Amortisation data

```
<LegData>
  <LegType> ... </LegType>
  <Payer> ... </Payer>
  <Currency> ... </Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <Amortizations>
    <AmortizationData>
      <Type>FixedAmount</Type>
      <Value>1000000</Value>
      <StartDate>20170203</StartDate>
      <Frequency>1Y</Frequency>
      <Underflow>false</Underflow>
    </AmortizationData>
    <AmortizationData>
      ...
    </AmortizationData>
  </Amortizations>
  ...
</LegData>
```

The user can specify a sequence of `AmortizationData` items in order to switch from one kind of amortisation to another etc. Within each `AmortisationData` block the meaning of elements is

- Type: Amortisation type with allowable values *FixedAmount, RelativeToInitialNotional, RelativeToPreviousNotional, Annuity.*
- Value: Interpreted depending on `Type`, see below.
- StartDate: Amortisation starts on first schedule date on or beyond StartDate.
- Frequency, entered as a period: Frequency of amortisations.
- Underflow: Allow amortisation below zero notional if `true`, otherwise amortisation stops at zero notional.

The amortisation data block's `Value` element is interpreted depending on the chosen `Type`:

- FixedAmount: The value is interpreted as a notional amount to be subtracted from the current notional on each amortisation date.
- RelativeToInitialNotional: The value is interpreted as a fraction of the **initial** notional to be subtraced from the current notional on each amortisation date.
- RelativeToPreviousNotional: The value is interpreted as a fraction of the **previous** notional to be subtraced from the current notional on each amortisation date.
- Annuity: The value is interpreted as annuity amount (redemption plus coupon).

Annuity type amortisation is supported for fixed rate legs as well as floating (ibor) legs. Note:

- Floating annuities require at least one previous vanilla coupon in order to work out the first amortisation amount.
- Floating legs with annuity amortisation currently do not allow switching the amortisation type, i.e. only a single block of `AmortizationData`.

8.3.7 CMS Leg Data

Listing 109 shows an example for a leg of type CMS.

Listing 109: CMS leg data

```xml
<LegData>
  <LegType>CMS</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <CMSLegData>
    <Index>EUR-CMS-10Y</Index>
    <Spreads>
      <Spread>0.0010</Spread>
    </Spreads>
    <Gearings>
      <Gearing>2.0</Gearing>
    </Gearings>
    <Caps>
      <Cap>0.05</Cap>
    </Caps>
    <Floors>
      <Floor>0.01</Floor>
    </Floors>
  </CMSLegData>
  <NakedOption>N</NakedOption>
</LegData>
```

The CMSLegData block contains the following elements:

- Index: The underlying CMS index.
- Spreads: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.
- IsInArrears: *true* indicates that fixing is in arrears, i.e. the fixing gap is calculated in relation to the current period end date.
  false indicates that fixing is in advance, i.e. the fixing gap is calculated in relation to the previous period end date.
- FixingDays: This is the fixing gap, i.e. the number of days before the period end date an index fixing is taken.

- Gearings: This node contains child elements of type `Gearing` indicating that the coupon rate is multiplied by the given factors. The mode of specification is analogous to spreads, see above.

- Caps: This node contains child elements of type `Cap` indicating that the coupon rate is capped at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above.

- Floors: This node contains child elements of type `Floor` indicating that the coupon rate is floored at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above.

- NakedOption: Optional node (defaults to N), if Y the leg represents only the embedded floor, cap or collar. By convention these embedded options are considered long if the leg is a receiver leg, otherwise short.

8.3.8 CMS Spread Leg Data

Listing 110 shows an example for a leg of type CMSSpread.

Listing 110: CMS Spread leg data

```
<LegData>
  <LegType>CMSSpread</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <CMSSpreadLegData>
    <Index1>EUR-CMS-10Y</Index1>
    <Index2>EUR-CMS-2Y</Index2>
    <Spreads>
      <Spread>0.0010</Spread>
    </Spreads>
    <Gearings>
      <Gearing>8.0</Gearing>
    </Gearings>
    <Caps>
      <Cap>0.05</Cap>
    </Caps>
    <Floors>
      <Floor>0.01</Floor>
    </Floors>
  </CMSSpreadLegData>
  <NakedOption>N</NakedOption>
</LegData>
```

The elements of the CMSSpreadLegData block are identical to those of the CMSLegData (see 8.3.7), except for the index which is defined by two CMS indices as the difference between `Index1` and `Index2`.

8.3.9 Digital CMS Spread Leg Data

Listing 111 shows an example for a leg of type DigitalCMSSpread.

Listing 111: Digital CMS Spread leg data

```xml
<LegData>
  <LegType>DigitalCMSSpread</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <DigitalCMSSpreadLegData>
      <CMSSpreadLegData>
        <Index1>EUR-CMS-10Y</Index1>
        <Index2>EUR-CMS-2Y</Index2>
        <Spreads>
          <Spread>0.0010</Spread>
        </Spreads>
        <Gearings>
          <Gearing>8.0</Gearing>
        </Gearings>
        <NakedOption>N</NakedOption>
      </CMSSpreadLegData>
      <CallPosition>Long</CallPosition>
      <IsCallATMIncluded>false</IsCallATMIncluded>
      <CallStrikes>
          <Strike>0.0001</Strike>
      </CallStrikes>
      <CallPayoffs>
          <Payoff>0.0001</Payoff>
      </CallPayoffs>
      <PutPosition>Long</PutPosition>
      <IsPutATMIncluded>false</IsPutATMIncluded>
      <PutStrikes>
          <Strike>0.001</Strike>
      </PutStrikes>
      <PutPayoffs>
          <Payoff>0.001</Payoff>
      </PutPayoffs>
  </DigitalCMSSpreadLegData>
</LegData>
```

The DigitalCMSLegData block contains the following elements:

- DigitalCMSLegData: a DigitalCMSSpreadLegData block describing the underlying digital cms spread leg (see 8.3.9.)
- CallPosition: Specifies whether the call option position is long or short.

- IsCallATMIncluded: inclusion flag on the call payoff if the call option ends at-the-money

- CallStrikes: strike rate for the the call option

- CallPayoffs: digital call option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing

- PutPosition: Specifies whether the put option position is long or short.

- IsPutATMIncluded: inclusion flag on the put payoff if the put option ends at-the-money

- PutStrikes: strike rate for the the put option

- PutPayoffs: digital put option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing

8.3.10 Equity Leg Data

Listing 112 shows an example of a leg of type Equity. The EquityLegData block contains the following elements:

- ReturnType: *Price* indicates that the coupons on the equity leg are determined by the price movement of the underlying equity, whereas *Total* indicates that coupons are determined by the total return of the underlying equity including dividends.

  Allowable values: *Price* or *Total*

- Name: The identifier of the underlying equity or equity index.

  Allowable values: Any string (provided it is an ID of an equity in the market configuration). Typically an ISIN-code with the *ISIN:* prefix.

- InitialPrice [Optional]: Initial Price of the equity, if not present, the first valuation date is used to determine the initial price.

  Allowable values: Any positive real number

- NotionalReset [Optional]: Defaults to *false*. If set to *true*, the Notional is reset at the beginning of each coupon period. Notional resets only affect the equity leg.

  Allowable values: *true* or *false*

- DividendFactor [Optional]: Factor of dividend to be included in return. Note that the DividendFactor is only relevant when the ReturnType is set to *Total*. It is not used if the ReturnType is set to *Price*.

  Allowable values: $0 \leq$ DividendFactor ≤ 1. Defaults to *1* if left blank or omitted.

- ValuationSchedule [Optional]: Schedule of dates for equity valuation.

  Allowable values: A node on the same form as `ScheduleData`, (see 8.3.3). If omitted, equity valuation dates follow the schedule of the equity leg adjusted for FixingDays.

- FixingDays [Optional]: The number of days before payment date for equity valuation. *N.B.* Only used when no valuation schedule present. Defaults to *0*.

Listing 112: Equity leg data

```xml
<LegData>
  <LegType>Equity</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-03-01</StartDate>
      <EndDate>2018-03-01</EndDate>
      <Tenor>3M</Tenor>
      <Calendar>TARGET</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <EquityLegData>
    <ReturnType>Price</ReturnType>
    <Name>ISIN:US78378X1072</Name>
    <InitialPrice>100</InitialPrice>
    <NotionalReset>true</NotionalReset>
    <DividendFactor>1</DividendFactor>
    <ValuationSchedule>
      <Dates>
        <Calendar>USD</Calendar>
        <Convention>ModifiedFollowing</Convention>
        <Dates>
          <Date>2016-03-01</Date>
          <Date>2016-06-01</Date>
          <Date>2016-09-01</Date>
          <Date>2016-12-01</Date>
          <Date>2017-03-01</Date>
          <Date>2017-06-01</Date>
          <Date>2017-09-01</Date>
          <Date>2017-12-01</Date>
          <Date>2018-03-01</Date>
        </Dates>
      </Dates>
     </ValuationSchedule>
     <FixingDays>0</FixingDays>
  </EquityLegData>
</LegData>
```

8.3.11 CPI Leg Data

Listing 113 shows an example for a leg of type CPI. The CPILegData block contains the following elements:

- Index: The underlying zero inflation index.

  Allowable values: Any string (provided it is the ID of an inflation index in the market configuration).

- Rates: The fixed real rate(s) of the leg. As usual, this can be a single value, a vector of values or a dated vector of values.

  Allowable values: Each rate element can take any real number. The rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.

- BaseCPI: The base CPI used to determine the lifting factor for the fixed coupons.

  Allowable values: Any positive real number.

- ObservationLag: The observation lag to be applied.

  Allowable values: An integer followed by *D*, *W*, *M* or *Y*. Interpolation lags are typically expressed in *M*, months.

- Interpolated: A flag indicating whether interpolation should be applied to inflation fixings.

  Allowable values: *true, false*

- SubtractInflationNotional [Optional]: A flag indicating whether the non-inflation adjusted notional amount should be subtracted from the the final inflation-adjusted notional exchange at maturity. Note that the final coupon payment is not affected by this flag.
  Final notional payment if *true*: $N\frac{CPI_T}{CPI_{Base}} - N$.
  If *false*: $N\frac{CPI_T}{CPI_{Base}}$
  Allowable values: *true, false*
  Defaults to *false* if left blank or omitted.

Listing 113: CPI leg data

```
<LegData>
  <LegType>CPI</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>1Y</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <CPILegData>
    <Index>UKRPI</Index>
    <Rates>
      <Rate>0.02</Rate>
    </Rates>
    <BaseCPI>210</BaseCPI>
    <ObservationLag>2M</ObservationLag>
    <Interpolated>false</Interpolated>
  </CPILegData>
</LegData>
```

8.3.12 YY Leg Data

Listing 114 shows an example for a leg of type YY. The YYLegData block contains the following elements:

- Index: The underlying zero inflation index.

  Allowable values: Any string (provided it is the ID of an inflation index in the market configuration).

- FixingDays: The number of fixing days.

  Allowable values: An integer followed by D,

- ObservationLag: The observation lag to be applied.

  Allowable values: An integer followed by D, W, M or Y. Interpolation lags are typically expressed in M, months.

- Caps: This node contains child elements of type `Cap` indicating that the coupon rate is capped at the given rate (after applying gearing and spread, if any).

- Floors: This node contains child elements of type `Floor` indicating that the coupon rate is floored at the given rate (after applying gearing and spread, if any).

- NakedOption [Optional]: Optional node (defaults to N), if Y the leg represents only the embedded floor, cap or collar. By convention these embedded options are considered long if the leg is a receiver leg, otherwise short.

Listing 114: YY leg data

```xml
<LegData>
  <LegType>YY</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>1Y</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <YYLegData>
    <Index>EUHICPXT</Index>
    <FixingDays>2</FixingDays>
    <ObservationLag>2M</ObservationLag>
    <Interpolated>true</Interpolated>
    <Caps>
      <Cap>0.05</Cap>
    </Caps>
    <Floors>
      <Floor>0.01</Floor>
    </Floors>
    <NakedOption>N</NakedOption>
  </YYLegData>
</LegData>
```

8.3.13 ZeroCouponFixed Leg Data

Listing 115 shows an example for a leg of type Zero Coupon Fixed. The ZeroCouponFixedLegData block contains the following elements:

- Rates: The fixed real rate(s) of the leg. While this can be a single value, a vector of values or a dated vector of values, the current ZeroCouponFixed leg requires only a single rate value to be passed.

Allowable values: Each rate element can take any real number. The rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.

- Compounding: The method of compounding applied to the rate.

  Allowable values: 'Simple' i.e. (1 + r * t), or 'Compounded' i.e. (1 + r)t̂

Listing 115: ZeroCouponFixed leg data

```
<LegData>
  <LegType>ZeroCouponFixed</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>5Y</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <ZeroCouponFixedLegData>
      <Rates>
          <Rate>0.02</Rate>
      </Rates>
      <Compounding>Simple</Compounding>
  </ZeroCouponFixedLegData>
</LegData>
```

8.4 Allowable Values for Standard Trade Data

Trade Data	Allowable Values
`Date`	The following date formats are supported: *yyyymmdd* *yyyy-mm-dd* *yyyy/mm/dd* *yyyy.mm.dd* *dd-mm-yy* *dd/mm/yy* *dd.mm.yy* *dd-mm-yyyy* *dd/mm/yyyy* *dd.mm.yyyy* and Dates as serial numbers, comparable to Microsoft Excel dates, with a minimum of 367 for Jan 1, 1901, and a maximum of 109574 for Dec 31, 2199.
`Currency`	*ATS, AUD, BEF, BRL, CAD, CHF, CNY, CZK, DEM, DKK, EUR, ESP, FIM, FRF, GBP, GRD, HKD, HUF, IEP, ITL, INR, ISK, JPY, KRW, LUF, NLG, NOK, NZD, PLN, PTE, RON, SEK, SGD, THB, TRY, TWD, USD, ZAR, ARS, CLP, COP, IDR, ILS, KWD, PEN, MXN, SAR, RUB, TND, MYR, UAH, KZT, QAR, MXV, CLF, EGP, BHD, OMR, VND, AED, PHP, NGN, MAD*, Note: Currency codes must also match available currencies in the `simulation.xml` file.
`Roll Convention`	*F, Following, FOLLOWING* *MF, ModifiedFollowing, Modified Following, MODIFIEDF* *P, Preceding, PRECEDING* *MP, ModifiedPreceding, Modified Preceding, MODIFIEDP* *U, Unadjusted, INDIFF*

Table 11: Allowable values for standard trade data.

Calendar	
Allowable Values	**Resulting Calendar**
TARGET, TGT, EUR	Target Calendar
CA,TRB, CAD	Canada Calendar
TKB, JP, JPY	Japan Calendar
ZUB, CHF	Switzerland Calendar
GB, LNB, UK	UK Calendar
London stock exchange	London Stock Exchange Calendar
US, NYB, USD	US Calendar
US-SET	US Settlement Calendar
US-GOV	US Government Bond Calendar
US-NYSE, New York stock exchange	US NYSE Calendar
US with Libor impact	US Calendar for Libor fixings
US-NERC	US NERC Calendar
AU, AUD	Australia Calendar
SA, ZAR	South Africa Calendar
SS, SEK	Sweden Calendar
ARS	Argentina Calendar
BRL	Brazil Calendar
CNY	China Calendar
CZK	Czech Republic Calendar
DEN, DKK	Denmark Calendar
FIN	Finland Calendar
HKD	HongKong Calendar
ISK	Iceland Calendar
INR	India Calendar
IDR	Indonesia Calendar
MXN	Mexico Calendar
NZD	New Zealand Calendar
NOK	Norway Calendar
PLN	Poland Calendar
RUB	Russia Calendar
SAR	Saudi Arabia
SGD	Singapore Calendar
KRW	South Korea Calendar
TWD	Taiwan Calendar
TRY	Turkey Calendar
UAH	Ukraine Calendar
WeekendsOnly	Weekends Only Calendar

Table 12: Allowable Values for Calendar. Combinations of up to four calendars can be provided using comma separated calendar names.

DayCount Convention	
Allowable Values	**Resulting DayCount Convention**
A360, Actual/360, ACT/360	Actual 360
A365, A365F, Actual/365, Actual/365 (fixed)	Actual 365 Fixed
T360, 30/360, 30/360 (Bond Basis), ACT/nACT	Thirty 360 (US)
30E/360, 30E/360 (Eurobond Basis)	Thirty 360 (European)
30/360 (Italian)	Thirty 360 (Italian)
ActActISDA, ActualActual (ISDA), ACT/ACT, ACT	Actual Actual (ISDA)
ActActISMA, ActualActual (ISMA)	Actual Actual (ISMA)
ActActAFB, Actual/Actual (AFB)	Actual Actual (AFB)

Table 13: Allowable Values for DayCount Convention

Index	
On form CCY-INDEX-TENOR, and matching available indices in the market data configuration.	
Index Component	**Allowable Values**
CCY-INDEX	*EUR-EONIA* *EUR-EURIBOR* *EUR-LIBOR* *EUR-CMS* *USD-FedFunds* *USD-LIBOR* *USD-SIFMA* *USD-CMS* *GBP-SONIA* *GBP-LIBOR* *GBP-CMS* *JPY-LIBOR* *JPY-TIBOR* *JPY-TONAR* *JPY-CMS* *CHF-LIBOR* *CHF-SARON* *AUD-LIBOR* *AUD-BBSW* *CAD-CDOR* *CAD-BA* *SEK-STIBOR* *SEK-LIBOR* *DKK-LIBOR* *DKK-CIBOR* *SGD-SIBOR* *SGD-SOR* *HKD-HIBOR* *NOK-NIBOR* *HUF-BUBOR* *IDR-IDRFIX* *INR-MIFOR* *MXN-TIIE* *PLN-WIBOR* *SKK-BRIBOR* *NZD-BKBM*
TENOR	An integer followed by *D, W, M or Y*

Table 14: Allowable values for Index.

9 Netting Set Definitions

The netting set definitions file - `netting.xml` - contains a list of definitions for various ISDA netting agreements. The file is written in XML format.

Each netting set is defined within its own `NettingSet` node. All of these `NettingSet` nodes are contained as children of a `NettingSetDefinitions` node.

There are two distinct cases to consider:

- An ISDA agreement which does not contain a *Credit Support Annex* (CSA).
- An ISDA agreement which does contain a CSA.

9.1 Uncollateralised Netting Set

If an ISDA agreement does not contain a Credit Support Annex, the portfolio exposures are not eligible for collateralisation. In such a case the netting set can be defined within the following XML template:

Listing 116: Uncollateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <Counterparty> </Counterparty>
    <ActiveCSAFlag> </ActiveCSAFlag>
    <CSADetails></CSADetails>
</NettingSet>
```

The meanings of the various elements are as follows:

- NettingSetId: The unique identifier for the ISDA netting set.
- Counterparty: The identifier for the counterparty to the ISDA agreement.
- ActiveCSAFlag: Boolean indicating whether the netting set is covered by a Credit Support Annex. For uncollateralised netting sets this flag should be *False*.
- CSADetails: Node containing as children details of the governing Credit Support Annex. For uncollateralised netting sets there is no need to store any information within this node.

9.2 Collateralised Netting Set

If an ISDA agreement contains a Credit Support Annex, the portfolio exposures are eligible for collateralisation. In such a case the netting set can be defined within the following XML template:

Listing 117: Collateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <Counterparty> </Counterparty>
    <ActiveCSAFlag> </ActiveCSAFlag>
    <CSADetails>
        <Bilateral> </Bilateral>
        <CSACurrency> </CSACurrency>
        <Index> </Index>
        <ThresholdPay> </ThresholdPay>
        <ThresholdReceive> </ThresholdReceive>
        <MinimumTransferAmountPay> </MinimumTransferAmountPay>
        <MinimumTransferAmountReceive> </MinimumTransferAmountReceive>
        <IndependentAmount>
            <IndependentAmountHeld> </IndependentAmountHeld>
            <IndependentAmountType> </IndependentAmountType>
        </IndependentAmount>
        <MarginingFrequency>
            <CallFrequency> </CallFrequency>
            <PostFrequency> </PostFrequency>
        </MarginingFrequency>
        <MarginPeriodOfRisk> </MarginPeriodOfRisk>
        <CollateralCompoundingSpreadReceive>
        </CollateralCompoundingSpreadReceive>
        <CollateralCompoundingSpreadPay> </CollateralCompoundingSpreadPay>
        <EligibleCollaterals>
            <Currencies>
                <Currency>USD</Currency>
                <Currency>EUR</Currency>
                <Currency>CHF</Currency>
                <Currency>GBP</Currency>
                <Currency>JPY</Currency>
                <Currency>AUD</Currency>
            </Currencies>
        </EligibleCollaterals>
    </CSADetails>
</NettingSet>
```

The first few nodes are shared with the template for uncollateralised netting sets:

- NettingSetId: The unique identifier for the ISDA netting set.
- Counterparty: The identifier for the counterparty to the ISDA agreement.
- ActiveCSAFlag: Boolean indicating whether the netting set is covered by a Credit Support Annex. For collateralised netting sets this flag should be *True*.
- CSADetails: Node containing as children details of the governing Credit Support Annex.

CSADetails

The `CSADetails` node contains details of the Credit Support Annex which are relevant for the purposes of exposure calculation. The meanings of the various elements are as follows:

Bilateral There are three possible values here:

- Bilateral: Both parties to the CSA are legally entitled to request collateral to cover their counterparty credit risk exposure on the underlying portfolio.
- CallOnly: Only we are entitled to hold collateral; the counterparty has no such entitlement.
- PostOnly: Only the counterparty is entitled to hold collateral; we have no such entitlement.

CSACurrency A three-letter ISO code specifying the master currency of the CSA. All monetary values specified within the CSA are assumed to be denominated in this currency.

Index The index is used to derive the fixing which is used for compounding cash collateral in the master currency of the CSA.

ThresholdPay A threshold amount above which the counterparty is entitled to request collateral to cover excess exposure.

ThresholdReceive A threshold amount above which we are entitled to request collateral from the counterparty to cover excess exposure.

MinimumTransferAmountPay Any margin calls issued by the counterparty must exceed this minimum transfer amount. If the collateral shortfall is less than this amount, the counterparty is not entitled to request margin.

MinimumTransferAmountReceive Any margin calls issued by us to the counterparty must exceed this minimum transfer amount. If the collateral shortfall is less than this amount, we are not entitled to request margin.

IndependentAmount This element contains two child nodes:

- IndependentAmountHeld: The netted sum of all independent amounts covered by this ISDA agreement/CSA. A negative number implies that the counterparty holds the independent amount.
- IndependentAmountType: The nature of the independent amount as defined within the Credit Support Annex. The only supported value here is *FIXED*.

This covers only the case where only one party has to post an independent amount. In a future release this will be extended to the situation prescribed by the Basel/IOSCO regulation (initial margin to be posted by both parties without netting).

MarginingFrequency This element contains two child nodes:

- CallFrequency: The frequency with which we are entitled to request additional margin from the counterparty (e.g. *1D*, *2W*, *1M*).
- PostFrequency: The frequency with which the counterparty is entitled to request additional margin from us (e.g. *1D*, *2W*, *1M*).

MarginPeriodOfRisk The length of time assumed necessary for closing out the portfolio position after a default event (e.g. *1D*, *2W*, *1M*).

CollateralCompoundingSpreadReceive The spread over the O/N interest accrual rate taken by the clearing house, when holding collateral.

CollateralCompoundingSpreadPay The spread over the O/N interest accrual rate taken by the clearing house, when collateral is held by the counterparty.

EligibleCollaterals For now the only supported type of collateral is cash. If the CSA specifies a set of currencies which are eligible as collateral, these can be listed using `Currency` nodes.

10 Market Data

In this section we discuss the market data, which enters into the calibration of OREs risk factor evolution models. Market data in the `market.txt` file is given in three columns; Date, Quote and Quote value.

- **Date**: The as of date of the market quote value.

  Allowable values: See `Date` in Table 11.

- **Quote**: A generic description that contains Instrument Type and Quote Type, followed by instrument specific descriptions (see 10.1 ff.). The base of a quote consists of InstType/QuoteType followed by instrument specific information separated by slashes "/".

  Allowable values for Instrument Types and Quote Types are given in Table 15.

- **Quote Value**: The market quote value in decimal form for the given quote on the given as of date. Quote values are assumed to be mid-market.

  Allowable values: Any real number.

Market Data Parameter	**Allowable Values**
Instrument Type	*ZERO, DISCOUNT, MM, MM_FUTURE, FRA, IMM_FRA, IR_SWAP, BASIS_SWAP, CC_BASIS_SWAP, CDS, CDS_INDEX, FX_SPOT, FX_FWD, SWAPTION, CAPFLOOR, FX_OPTION, HAZARD_RATE, RECOVERY_RATE, ZC_INFLATIONSWAP, YY_INFLATIONSWAP, ZC_INFLATIONCAPFLOOR, SEASONALITY, EQUITY_SPOT, EQUITY_FWD, EQUITY_DIVIDEND, EQUITY_OPTION, BOND, INDEX_CDS_OPTION, CPR*
Quote Type	*BASIS_SPREAD, CREDIT_SPREAD, YIELD_SPREAD, HAZARD_RATE, RATE, RATIO, PRICE, RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, BASE_CORRELATION, SHIFT*

Table 15: Allowable values for Instrument and Quote type market data.

An excerpt from a typical `market.txt` file is shown in Listing 118.

Listing 118: Excerpt of a market data file

```
2011-01-31 MM/RATE/EUR/0D/1D 0.013750
2011-01-31 MM/RATE/EUR/1D/1D 0.010500
2011-01-31 MM/RATE/EUR/2D/1D 0.010500
2011-01-31 MM/RATE/EUR/2D/1W 0.009500
2011-01-31 MM/RATE/EUR/2D/1M 0.008700
2011-01-31 MM/RATE/EUR/2D/2M 0.009100
2011-01-31 MM/RATE/EUR/2D/3M 0.010200
2011-01-31 MM/RATE/EUR/2D/4M 0.011000

2011-01-31 FRA/RATE/EUR/3M/3M 0.013080
2011-01-31 FRA/RATE/EUR/4M/3M 0.013890
2011-01-31 FRA/RATE/EUR/5M/3M 0.014630
2011-01-31 FRA/RATE/EUR/6M/3M 0.015230

2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y 0.014400
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y3M 0.015400
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/1Y6M 0.016500
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/2Y 0.018675
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/3Y 0.022030
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/4Y 0.024670
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/5Y 0.026870
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/6Y 0.028700
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/7Y 0.030125
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/8Y 0.031340
2011-01-31 IR_SWAP/RATE/EUR/2D/3M/9Y 0.032450
```

10.1 Zero Rate

The instrument specific information to be captured for quotes representing Zero Rates is shown in Table 16.

Property	**Allowable values**	**Description**
Instrument Type	*ZERO*	
Quote Type	*RATE, YIELD_SPREAD*	
Currency	See `Currency` in Table 11	Currency of the Zero rate
CurveId	A CCY concatenated with a Tenor. Should match CurveIds in the `yield-curves.xml` file	Unique identifier for the yield curve associated with the zero quote
DayCounter	See `DayCount Convention` in Table 13	The day count basis associated with the zero quote
Tenor or ZeroDate	Tenor: An integer followed by D, W, M or Y, ZeroDate: See `Date` in Table 11	Either a Tenor for tenor based zero quotes, or an explicit maturity date (ZeroDate)

Table 16: Zero Rate

Examples with a Tenor and with a ZeroDate:

- ZERO/RATE/USD/USD6M/A365F/6M
- ZERO/RATE/USD/USD6M/A365F/12-05-2018

10.2 Discount Factor

The instrument specific information to be captured for quotes representing Discount Factors is shown in Table 17.

Property	Allowable values	Description
Instrument Type	*DISCOUNT*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 11	Currency of the Discount rate
CurveId	A CCY concatenated with a Tenor. Should match CurveIds in the `yield-curves.xml` file	Unique identifier for the yield curve associated with the discount quote
Term or Discount-Date	Term: An integer followed by D, W, M or Y, Discount-Date: See `Date` in Table 11	Either a Term is used to determine the maturity date, or an explicit maturity date (Discount Date) is given.

Table 17: Discount Rate

If a Term is given in the last element of the quote, it is converted to a maturity date using a weekend only calendar.

Examples with a Term and with a DiscountDate:

- DISCOUNT/RATE/EUR/EUR3M/3Y
- DISCOUNT/RATE/EUR/EUR3M/A365F/12-05-2018

10.3 FX Spot Rate

Property	Allowable values	Description
Instrument Type	*FX*	
Quote Type	*RATE*	
Unit currency	See `Currency` in Table 11	Unit/Source currency
Target currency	See `Currency` in Table 11	Target currency

Table 18: FX Spot Rate

Example:

- FX/RATE/EUR/USD

10.4 FX Forward Rate

An FX Forward quote is expected in "forward points" quotation

$$\text{Forward Points} = \frac{\text{FX Forward} - \text{FX Spot}}{\text{Conversion Factor}}$$

with conversion factor set to 1.

Property	Allowable values	Description
Instrument Type	*FX_FWD*	
Quote Type	*RATE*	
Unit currency	See `Currency` in Table 11	Unit/Source currency
Target currency	See `Currency` in Table 11	Target currency
Term	An integer followed by D, W, M or Y.	Period from today to maturity

Table 19: FX Forward Rate

Example:

- FX_FWD/RATE/EUR/USD/1M

10.5 Deposit Rate

Property	Allowable values	Description
Instrument Type	*MM*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 11	Currency of the Deposit rate
Forward start	An integer followed by D, W, M or Y.	Period from today to start
Term	An integer followed by D, W, M or Y.	Period from start to maturity

Table 20: Deposit Rate

Deposits are usually quoted as ON (Overnight), TN (Tomorrow Next), SN (Spot Next), SW (Spot Week), 3W (3 Weeks), 6M (6 Months), etc.
Forward start for ON is today (i.e. forward start = 0D), for TN tomorrow (forward start = 1D), for SN two days from today (forward start = 2D). For longer term Deposits, forward start is derived from conventions, see 7.8, and is between 0D and 2D, i.e. "spot days" are between 0 and 2.

Example:

- MM/RATE/EUR/2D/3M

10.6 FRA Rate

Property	**Allowable values**	**Description**
Instrument Type	*FRA*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 11	Currency of the FRA rate
Forward start	An integer followed by D, W, M or Y	Period from today to start
Term	An integer followed by D, W, M or Y	Period from start to maturity

Table 21: FRA Rate

FRAs are typically quoted as e.g. 6x9 which means forward start 6M from today, maturity 9M from today, with appropriate adjustment of dates.

IMM FRA quotes are represented as follows.

Property	**Allowable values**	**Description**
Instrument Type	*IMM_FRA*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 11	Currency of the FRA rate
Start	An integer	Number of IMM dates from today to start
End	An integer	Number of IMM dates from today to maturity

Table 22: IMM FRA Rate

Example:

- FRA/RATE/EUR/9M/3M
- IMM_FRA/RATE/EUR/2/3

10.7 Money Market Futures Price

Property	Allowable values	Description
Instrument Type	*MM_FUTURE*	
Quote Type	*PRICE*	
Currency	See `Currency` in Table 11	Currency of the MM Future price
Expiry	Alphanumeric string of the form YYYY-MM	Expiry month and year
Contract	String	Contract name
Term	An integer followed by D, W, M or Y	Underlying Term

Table 23: Money Market Futures Price

Expiry month is quoted here as YYYY-MM. The exact expiry date follows from a date rule such as 3rd Wednesday of the specified month, adjusted to the following business day. The date rule is not quoted directly, but defined in the futures contract.

Example:

- MM_FUTURE/PRICE/EUR/2018-06/LIF3ME/3M

10.8 Swap Rate

Property	Allowable values	Description
Instrument Type	*IR_SWAP*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 11	Currency of the Swap rate
Forward start	An integer followed by D, W, M or Y	Generic period from today to start
Tenor	An integer followed by D, W, M or Y	Underlying index period
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 24: Swap Rate

Forward start is usually not quoted, but needs to be derived from conventions.

Example:

- IR_SWAP/RATE/EUR/2D/6M/10Y

10.9 Basis Swap Spread

Property	Allowable values	Description
Instrument Type	*BASIS_SWAP*	
Quote Type	*BASIS_SPREAD*	
Flat tenor	An integer followed by D, W, M or Y	Zero spread leg's index tenor
Tenor	An integer followed by D, W, M or Y	Non-zero spread leg's index tenor
Currency	See `Currency` in Table 11	Currency of the basis swap spread
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 25: Basis Swap Spread

Example:

- BASIS_SWAP/BASIS_SPREAD/6M/3M/CHF/10Y

10.10 Cross Currency Basis Swap Spread

Property	Allowable values	Description
Instrument Type	*CC_BASIS_SWAP*	
Quote Type	*BASIS_SPREAD*	
Flat currency	See `Currency` in Table 11	Currency for zero spread leg
Flat tenor	An integer followed by D, W, M or Y	Zero spread leg's index tenor
Currency	See `Currency` in Table 11	Currency for non-zero spread leg
Tenor	An integer followed by D, W, M or Y	Non-zero spread leg's index tenor
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 26: Cross Currency Basis Swap Spread

Example:

- CC_BASIS_SWAP/BASIS_SPREAD/USD/3M/JPY/6M/10Y

10.11 CDS Spread

Property	Allowable values	Description
Instrument Type	*CDS*	
Quote Type	*CREDIT_SPREAD*	
Issuer	String	Issuer name
Seniority	String	Seniority status
Currency	See `Currency` in Table 11	CDS Spread currency
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 27: CDS Spread

Example:

- CDS/CREDIT_SPREAD/GE/SeniorUnsec/EUR/5Y

10.12 CDS Recovery Rate

Property	Allowable values	Description
Instrument Type	*RECOVERY_RATE*	
Quote Type	*RATE*	
Issuer	String	Issuer name
Seniority	String	Seniority status
Currency	See `Currency` in Table 11	CDS Spread currency

Table 28: CDS Recovery Rate

Example:

- RECOVERY_RATE/RATE/GE/SeniorUnsec/EUR

10.13 Security Recovery Rate

Bond recovery rates can also be specified per security. This requires only one key, the security ID, no need to specify a seniority or currency as for CDS:

Property	Allowable values	Description
Instrument Type	*RECOVERY_RATE*	
Quote Type	*RATE*	
ID	String	Security ID

Table 29: Security Recovery Rate

Example:

- RECOVERY_RATE/RATE/SECURITY_1

10.14 Hazard Rate (Instantaneous Probability of Default)

This allows to directly pass hazard rates as instantaneous probabilities of default.

Property	Allowable values	Description
Instrument Type	*HAZARD_RATE*	
Quote Type	*RATE*	
Issuer	String	Issuer name
Seniority	String	Seniority status
Currency	See `Currency` in Table 11	Hazard rate currency
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 30: Hazard Rate

Example:

- HAZARD_RATE/RATE/CPTY_A/SR/USD/30Y
- HAZARD_RATE/RATE/CPTY_C/SR/EUR/0Y

10.15 FX Option Implied Volatility

Property	Allowable values	Description
Instrument Type	*FX_OPTION*	
Quote Type	*RATE_LNVOL*	
Unit currency	See `Currency` in Table 11	Unit/Source currency
Target currency	See `Currency` in Table 11	Target currency
Expiry	An integer followed by D, W, M or Y	Period from today to expiry
Strike	*ATM, RR, BF*	ATM (Straddle), RR (Risk Reversal), BF (Butterfly)

Table 31: FX Option Implied Volatility

Volatilities are quoted in terms of strategies - at-the-money straddle, risk reversal and butterfly.

Example:

- FX_OPTION/RATE_LNVOL/EUR/USD/3M/ATM

10.16 Cap/Floor Implied Volatility

Property	Allowable values	Description
Instrument Type	*CAPFLOOR*	
Quote Type	*RATE_LNVOL*	
Currency	See `Currency` in Table 11	Currency of the Cap/Floor volatility
Term	An integer followed by D, W, M or Y	Period from start to expiry
IndexTenor	An integer followed by D, W, M or Y	Underlying index tenor
Atm	*1, 0*	ATM volatility quote if true (1), otherwise (0) smile quote
Relative	*1, 0*	Relative quote (to be added to atm vol) if true (1), otherwise (0) absolute quote
Strike	Real number	Strike rate

Table 32: Cap/Floor Implied Volatility

Examples:

- CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/0/0/0.0350 (smile, absolute, strike 3.5%)
- CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/1/0/0.0000 (atm, absolute, irrelevant strike)

10.17 Swaption Implied Volatility

Property	Allowable values	Description
Instrument Type	*SWAPTION*	
Quote Type	*RATE_LNVOL, RATE_NVOL, RATE_SLNVOL*	Lognormal quoted volatility, Normal quoted volatility, shifted lognormal quoted volatility
Currency	See `Currency` in Table 11	Currency of the Swaption volatility
Expiry	An integer followed by D, W, M or Y	Period from start to expiry
Term	An integer followed by D, W, M or Y	Underlying Swap term
Dimension	*Smile, ATM*	Whether volatility quote is a Smile or ATM
Strike	Real number	Strike rate - (not required for ATM), as deviation from the ATM strike

Table 33: Swaption Implied Volatility

Note: The volatility quote is expected to be an absolute volatility, and not the deviation from the at-the-money volatility (the latter is e.g. the quotation convention used by BGC partners).

Examples:

- SWAPTION/RATE_LNVOL/EUR/5Y/10Y/ATM (absolute ATM vol quote)
- SWAPTION/RATE_LNVOL/EUR/5Y/10Y/Smile/0.0050 (absolute vol quote for ATM strike plus 50bp)

10.18 Equity Spot Price

Property	**Allowable values**	**Description**
Instrument Type	*EQUITY_SPOT*	
Quote Type	*PRICE*	
Name	String	Identifying name of the equity
Currency	See `Currency` in Table 11	Currency of the equity

Table 34: Equity Spot Price

10.19 Equity Forward Price

Property	**Allowable values**	**Description**
Instrument Type	*EQUITY_FWD*	
Quote Type	*PRICE*	
Name	String	Identifying name of the equity
Currency	See `Currency` in Table 11	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 35: Equity Forward Price

Examples:

- EQUITY_FWD/PRICE/SP5/USD/2016-06-16
- EQUITY_FWD/PRICE/SP5/USD/2Y

10.20 Equity Dividend Yield

Property	Allowable values	Description
Instrument Type	*EQUITY_DIVIDEND*	
Quote Type	*RATE*	
Name	String	Identifying name of the equity
Currency	See `Currency` in Table 11	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 36: Equity Dividend Yield Rate

Examples:

- EQUITY_DIVIDEND/RATE/SP5/USD/2016-06-16
- EQUITY_DIVIDEND/RATE/SP5/USD/2Y

10.21 Equity Option Implied Volatility

Property	Allowable values	Description
Instrument Type	*EQUITY_OPTION*	
Quote Type	*RATE_LNVOL*	
Name	String	Identifying name of the equity
Currency	See `Currency` in Table 11	Currency of the equity
Expiry	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote
Strike	`ATMF`, or else any `Real`	strike price

Table 37: Equity Option Implied Volatility

Volatilities are quoted as a function of strike price - either at-the-money forward (`ATMF`) or else a specified real number, corresponding to the absolute strike value.Only log-normal implied volatilities (`RATE_LNVOL`) are supported.

Example:

- EQUITY_OPTION/RATE_LNVOL/SP5/USD/6M/ATMF
- EQUITY_OPTION/RATE_LNVOL/SP5/USD/2018-06-30/ATMF

10.22 Zero Coupon Inflation Swap Rate

Property	Allowable values	Description
Instrument Type	*ZC_INFLATIONSWAP*	
Quote Type	*RATE*	
Index	String	Identifying name of the inflation index
Maturity	integer followed by D, W, M or Y	Maturity of the swap quote

Table 38: Zero Coupon Inflation Swap Rate

Examples:

- ZC_INFLATIONSWAP/RATE/EUHICPXT/1Y
- ZC_INFLATIONSWAP/RATE/EUHICPXT/2Y

Examples for inflation index names include EUHICP, EUHICPXT,FRHICP, UKRPI, USCPI, ZACPI.

10.23 Year on Year Inflation Swap Rate

Property	Allowable values	Description
Instrument Type	*YY_INFLATIONSWAP*	
Quote Type	*RATE*	
Index	String	Identifying name of the inflation index
Maturity	integer followed by D, W, M or Y	Maturity of the swap quote

Table 39: Year on Year Inflation Swap Rate

Examples:

- YY_INFLATIONSWAP/RATE/EUHICPXT/1Y
- YY_INFLATIONSWAP/RATE/EUHICPXT/2Y

Examples for inflation index names include EUHICP, EUHICPXT,FRHICP, UKRPI, USCPI, ZACPI.

10.24 Zero Coupon Inflation Cap Floor Price

Property	Allowable values	Description
Instrument Type	*ZC_INFLATIONCAPFLOOR*	
Quote Type	*PRICE*	
Index	String	Identifying name of the inflation index
Maturity	integer followed by D, W, M or Y	Maturity of the swap quote
Cap/Floor	C or F	Cap or Floor tag
Strike	Real number	Strike

Table 40: Zero Coupon Inflation Cap Floor Price

Examples:

- ZC_INFLATIONCAPFLOOR/PRICE/EUHICPXT/1Y/F/-0.02
- ZC_INFLATIONCAPFLOOR/PRICE/EUHICPXT/2Y/C/0.01

Examples for inflation index names include EUHICP, EUHICPXT,FRHICP, UKRPI, USCPI, ZACPI.

10.25 Inflation Seasonality Correction Factors

Property	Allowable values	Description
Instrument Type	*SEASONALITY*	
Quote Type	*RATE*	
Type	MULT	Type of the correction factor
Index	String	Identifying name of the inflation index
Month	JAN, ..., DEC	Month of the correction factor

Table 41: Inflation Seasonality Correction Factors

Examples:

- SEASONALITY/RATE/MULT/EUHICPXT/JAN
- SEASONALITY/RATE/MULT/EUHICPXT/FEB
- SEASONALITY/RATE/MULT/EUHICPXT/NOV

Examples for inflation index names include EUHICP, EUHICPXT,FRHICP, UKRPI, USCPI, ZACPI.

10.26 Bond Yield Spreads

Property	Allowable values	Description
Instrument Type	*BOND*	
Quote Type	*YIELD_SPREAD*	
Name	String	Identifying name of the bond

Table 42: Bond Yield Spreads

This quote provides the spread for a specified bond over the benchmark rate.

Examples:

- BOND/YIELD_SPREAD/SECURITY_1

10.27 Correlations

Property	Allowable values	Description
Instrument Type	*Correlation*	
Quote Type	*RATE or PRICE*	
Index1	String	Identifying name of the first index
Index2	String	Identifying name of the second index

Table 43: Correlation quotes

This quote either provides the correlation between two indices, in which case Quote Type is RATE, or a premium that can be used to bootstrap the correlations, in which case Quote Type is Price. Currently only CMS Spread correlations are supported, in this case the Price quote is the price of a CMS Spread Cap.

Examples:

- CORRELATION/RATE/INDEX1/INDEX2/1Y/ATM

10.28 Conditional Prepayment Rates

Property	Allowable values	Description
Instrument Type	*CPR*	
Quote Type	*RATE*	
Name	String	Identifying name of the bond

Table 44: Conditional Prepayment Rates

This quote provides the spread for a specified bond over the benchmark rate.

Examples:

- CPR/RATE/SECURITY_1

11 Fixing History

Historical fixings data in the `fixings.txt` file is given in three columns; Index Name, Fixing Date and Index value. Columns are separated by semicolons ";" or blanks. Fixings are used in cases where the current coupon of a trade has been fixed in the past, or other path dependent features.

- Fixing Date: The date of the fixing.

  Allowable values: See `Date` in Table 11.

- Index Name: The name of the Index.

  Allowable values are given in Table 45.

- Index Value: The index value for the given fixing date.

  Allowable values: Any real number (not expressed as a percentage or basis points).

An excerpt of a fixings file is shown in Listing 119. Note that alternative index name formats are used (Table 45).

Listing 119: Excerpt of a fixings file

```
20150202 EUR-EONIA -0.00024
20150202 EUR-EURIBOR-1M 0.00003
20150202 EUR-EURIBOR-1W -0.00022
20150202 EUR-EURIBOR-2W -0.00017
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-6M 0.00134
20150202 EUR-EURIBOR-6M 0.00271
20150202 GBP-LIBOR-12M 0.009565
20150202 GBP-LIBOR-1M 0.0050381
20150202 GBP-LIBOR-1W 0.0047938
20150202 GBP-LIBOR-3M 0.0056338
20150202 GBP-LIBOR-6M 0.006825
20150202 JPY-LIBOR-12M 0.0026471
20150202 JPY-LIBOR-1M 0.0007143
20150202 JPY-LIBOR-1W 0.0004357
20150202 JPY-LIBOR-3M 0.0010429
20150202 JPY-LIBOR-6M 0.0014357
20150202 USD-LIBOR-12M 0.006194
20150202 USD-LIBOR-1M 0.001695
20150202 USD-LIBOR-1W 0.00136
20150202 USD-CMS-10Y 0.01500
20150202 EUR-CMS-20Y 0.01700
20150202 FX-ECB-EUR-USD 1.0919
20150801 FRHICP 100.36
```

IR Index of form CCY-INDEX-TENOR:	
Index Component	**Allowable Values**
CCY-INDEX	*EUR-EONIA* *EUR-EURIBOR* *EUR-LIBOR* *USD-FedFunds* *USD-LIBOR* *GBP-SONIA* *GBP-LIBOR* *JPY-TONAR* *JPY-LIBOR* *CHF-LIBOR* *CHF-TOIS* *AUD-LIBOR* *AUD-BBSW* *CAD-CDOR* *CAD-BA* *CAD-LIBOR* *SEK-STIBOR* *SEK-LIBOR* *DKK-LIBOR* *DKK-CIBOR* *SGD-SIBOR* *HKD-HIBOR* *CZK-PRIBOR* *HUF-BUBOR* *IDR-IDRFIX* *INR-MIFOR* *JPY-TIBOR* *KRW-KORIBOR* *MXN-TIIE* *MYR-KLIBOR* *NOK-NIBOR* *NZD-BKBM* *PLN-WIBOR* *SEK-STIBOR* *SGD-SOR* *SKK-BRIBOR* *TWD-TAIBOR* *ZAR-JIBAR* *DEM-LIBOR*
TENOR	An integer followed by *D, W, M or Y*

Table 45: Allowable values for IR indices.

If the interest rate index is for an overnight rate (e.g. EONIA), then the third token (i.e. the tenor) is not needed.

IR Swap Index of form CCY-CMS-TENOR:	
Index Component	**Allowable Values**
CCY	Any supported currency code
CMS	Must be "CMS" (to denote a swap index)
TENOR	An integer followed by *D, W, M or Y*

Table 46: Allowable values for IR swap indices.

FX fixings of form FX-SOURCE-FOR-DOM:	
Index Component	**Allowable Values**
FX	Must be "FX" (to denote an FX fixing)
SOURCE	Any string
FOR	Any supported currency code
DOM	Any supported currency code

Table 47: Allowable values for FX fixings.

Zero Inflation Indices of form NAME:	
Index Component	**Allowable Values**
NAME	*EUHICP* *EUHICPXT* *FRHICP* *UKRPI* *USCPI* *ZACPI*

Table 48: Allowable values for zero inflation indices.

A Methodology Summary

A.1 Risk Factor Evolution Model

ORE applies the cross asset model described in detail in [20] to evolve the market through time. So far the evolution model in ORE supports IR and FX risk factors for any number of currencies, Equity and Inflation as well as Credit (for $t0$-pricing). Extensions to full simulation of Credit and Commodity is planned.
The Cross Asset Model is based on the Linear Gauss Markov model (LGM) for interest rates, lognormal FX and equity processes and the Dodgson-Kainth model for inflation. We identify a single *domestic* currency; its LGM process, which is labelled z_0; and a set of n foreign currencies with associated LGM processes that are labelled z_i, $i = 1, \ldots, n$. We denote the equity spot price processes with state variables s_j and the index of the denominating currency for the equity process as $\phi(j)$. The dividend yield corresponding to each equity process s_j is denoted by q_j.
Following [20], 13.27 - 13.29 we write the inflation processes in the domestic LGM measure with state variables $z_{I,k}$ and $y_{I,k}$ for $k = 1, \ldots, K$. If we consider n foreign exchange rates for converting foreign currency amounts into the single domestic currency by multiplication, x_i, $i = 1, \ldots, n$, then the cross asset model is given by the system of SDEs

$$
\begin{aligned}
dz_0 &= \alpha_0 \, dW_0^z \\
dz_i &= \gamma_i \, dt + \alpha_i \, dW_i^z, \qquad i > 0 \\
\frac{dx_i}{x_i} &= \mu_i \, dt + \sigma_i \, dW_i^x, \qquad i > 0 \\
\frac{ds_j}{s_j} &= \mu_j^S \, dt + \sigma_j^S \, dW_j^S \\
dz_{I,k} &= \alpha_{I,k}(t) dW_k^I \\
dy_{I,k} &= \alpha_{I,k}(t) H_{I,k}(t) dW_k^I \\
\\
\gamma_i &= -\alpha_i^2 \, H_i - \rho_{ii}^{zx} \, \sigma_i \, \alpha_i + \rho_{i0}^{zz} \, \alpha_i \, \alpha_0 \, H_0 \\
\mu_i &= r_0 - r_i + \rho_{0i}^{zx} \, \alpha_0 \, H_0 \, \sigma_i \\
\mu_j^S &= (r_{\phi(j)}(t) - q_j(t) + \rho_{0j}^{zs} \alpha_0 H_0 \sigma_j^S - \epsilon_{\phi(j)} \rho_{j\phi(j)}^{sx} \sigma_j^S \sigma_{\phi(j)}) \\
r_i &= f_i(0,t) + z_i(t) \, H_i'(t) + \zeta_i(t) \, H_i(t) \, H_i'(t), \quad \zeta_i(t) = \int_0^t \alpha_i^2(s) \, ds \\
dW_a^\alpha \, dW_b^\beta &= \rho_{ij}^{\alpha\beta} \, dt, \qquad \alpha, \beta \in \{z, x, I\}, \qquad a, b \text{ suitable indices}
\end{aligned}
$$

where we have dropped time dependencies for readability, $f_i(0,t)$ is the instantaneous forward curve in currency i, and ϵ_i is an indicator such that $\epsilon_i = 1 - \delta_{0i}$, where δ is the Kronecker delta.

Parameters $H_i(t)$ and $\alpha_i(t)$ (or alternatively $\zeta_i(t)$) are LGM model parameters which determine, together with the stochastic factor $z_i(t)$, the evolution of numeraire and zero bond prices in the LGM model:

$$N(t) = \frac{1}{P(0,t)} \exp\left\{ H_t\, z_t + \frac{1}{2} H_t^2\, \zeta_t \right\} \tag{1}$$

$$P(t,T,z_t) = \frac{P(0,T)}{P(0,t)} \exp\left\{ -(H_T - H_t)\, z_t - \frac{1}{2}\left(H_T^2 - H_t^2\right)\, \zeta_t \right\}. \tag{2}$$

Note that the LGM model is closely related to the Hull-White model in T-forward measure [20].

The parameters $H_{I,k}(t)$ and $\alpha_{I,k}(t)$ determine together with the factors $z_{I,k}(t), y_{I,k}(t)$ the evolution of the spot Index $I(t)$ and the forward index $\hat{I}(t,T) = P_I(t,T)/P_n(t,T)$ defined as the ratio of the inflation linked zero bond and the nominal zero bond,

$$\begin{aligned}
\hat{I}(t,T) &= \frac{\hat{I}(0,T)}{\hat{I}(0,t)} e^{(H_{I,k}(T) - H_{I,k}(t)) z_{I,k}(t) + \tilde{V}(t,T)} \\
I(t) &= I(0)\hat{I}(0,t) e^{H_{I,k}(t) z_{I,k}(t) - y_{I,k}(t) - V(0,t)}
\end{aligned}$$

with, in case of domestic currency inflation,

$$\begin{aligned}
V(t,T) &= \frac{1}{2}\int_t^T (H_{I,k}(T) - H_{I,k}(s))^2 \alpha_{I,k}^2(s) ds \\
&\quad - \rho_{0,k}^{zI} H_0(T) \int_t^T (H_{I,k}(t) - H_{I,k}(s)) \alpha_0(s) \alpha_{I,k}(s) ds \\
\tilde{V}(t,T) &= V(t,T) - V(0,T) - V(0,t) \\
&= -\frac{1}{2}(H_{I,k}^2(T) - H_{I,k}^2(t)) \zeta_{I,k}(t,0) \\
&\quad + (H_{I,k}(T) - H_{I,k}(t)) \zeta_{I,k}(t,1) \\
&\quad + (H_0(T) H_{I,k}(T) - H_0(t) H_{I,k}(t)) \zeta_{0I}(t,0) \\
&\quad - (H_0(T) - H_0(t)) \zeta_{0I}(t,1) \\
V(0,t) &= \frac{1}{2} H_{I,k}^2(t) \zeta_{I,k}(t,0) - H_{I,k}(t) \zeta_{I,k}(t,1) + \frac{1}{2}\zeta_{I,k}(t,2) \\
&\quad - H_0(t) H_{I,k}(t) \zeta_{0I}(t,0) + H_0(t) \zeta_{0I}(t,1) \\
\zeta_{I,k}(t,k) &= \int_0^t H_{I,k}^k(s) \alpha_{I,k}^2(s) ds \\
\zeta_{0I}(t,k) &= \rho_{0,k}^{zI} \int_0^t H_{I,k}^k(t) \alpha_0(s) \alpha_{I,k}(s) ds
\end{aligned}$$

and for foreign currency inflation in currency $i > 0$, with

$$\tilde{V}(t,T) = V(t,T) - V(0,T) + V(0,T)$$

and

$$
\begin{aligned}
V(t,T) &= \frac{1}{2}\int_t^T (H_{I,k}(T) - H_{I,k}(s))^2 \alpha_{I,k}(s) ds \\
&\quad -\rho^{zI}_{0,k} \int_t^T H_0(s)\alpha_0(s)(H_{I,k}(T) - H_{I,k}(s)\alpha_{I,k}(s)) ds \\
&\quad -\rho^{zI}_{i,k} \int_t^T (H_i(T) - H_i(s))\alpha_i(s)(H_{I,k}(T) - H_{I,k}(s))\alpha_{I,k}(s) ds \\
&\quad +\rho^{xI}_{i,k} \int_t^T \sigma_i(s)(H_{I,k}(T) - H_{I,k}(s))\alpha_{I,k}(s) ds
\end{aligned}
$$

A.2 Analytical Moments of the Risk Factor Evolution Model

We follow [20], chapter 16. The expectation of the interest rate process z_i conditional on $\mathcal{F}_{t_0}$ at $t_0 + \Delta t$ is

$$
\begin{aligned}
\mathbb{E}_{t_0}[z_i(t_0 + \Delta t)] &= z_i(t_0) + \mathbb{E}_{t_0}[\Delta z_i], \qquad \text{with} \quad \Delta z_i = z_i(t_0 + \Delta t) - z_i(t_0) \\
&= z_i(t_0) - \int_{t_0}^{t_0+\Delta t} H_i^z\,(\alpha_i^z)^2\,du + \rho_{0i}^{zz} \int_{t_0}^{t_0+\Delta t} H_0^z\,\alpha_0^z\,\alpha_i^z\,du \\
&\quad -\epsilon_i \rho_{ii}^{zx} \int_{t_0}^{t_0+\Delta t} \sigma_i^x\,\alpha_i^z\,du
\end{aligned}
$$

where ϵ_i is zero for $i = 0$ (domestic currency) and one otherwise.

The expectation of the FX process x_i conditional on $\mathcal{F}_{t_0}$ at $t_0 + \Delta t$ is

$$
\begin{aligned}
\mathbb{E}_{t_0}[\ln x_i(t_0 + \Delta t)] &= \ln x_i(t_0) + \mathbb{E}_{t_0}[\Delta \ln x_i], \qquad \text{with} \quad \Delta \ln x_i = \ln x_i(t_0 + \Delta t) - \ln x_i(t_0) \\
&= \ln x_i(t_0) + (H_0^z(t) - H_0^z(s))\,z_0(s) - (H_i^z(t) - H_i^z(s))\,z_i(s) \\
&\quad + \ln\left(\frac{P_0^n(0,s)}{P_0^n(0,t)}\,\frac{P_i^n(0,t)}{P_i^n(0,s)}\right) \\
&\quad - \frac{1}{2}\int_s^t (\sigma_i^x)^2\,du \\
&\quad + \frac{1}{2}\left((H_0^z(t))^2\zeta_0^z(t) - (H_0^z(s))^2\zeta_0^z(s) - \int_s^t (H_0^z)^2(\alpha_0^z)^2\,du\right) \\
&\quad - \frac{1}{2}\left((H_i^z(t))^2\zeta_i^z(t) - (H_i^z(s))^2\zeta_i^z(s) - \int_s^t (H_i^z)^2(\alpha_i^z)^2\,du\right) \\
&\quad + \rho_{0i}^{zx} \int_s^t H_0^z\,\alpha_0^z\,\sigma_i^x\,du \\
&\quad - \int_s^t (H_i^z(t) - H_i^z)\,\gamma_i\,du, \qquad \text{with} \quad s = t_0, \quad t = t_0 + \Delta t
\end{aligned}
$$

with

$$
\gamma_i = -H_i^z\,(\alpha_i^z)^2 + H_0^z\,\alpha_0^z\,\alpha_i^z\,\rho_{0i}^{zz} - \sigma_i^x\,\alpha_i^z\,\rho_{ii}^{zx}
$$

The expectation of the Inflation processes $z_{I,k}, y_{I,k}$ conditional on $\mathcal{F}_{t_0}$ at any time $t > t_0$ is equal to $z_{I,k}(t_0)$ resp. $y_{I,k}(t_0)$ since both processes are drift free.

The expectation of the equity processes s_j conditional on $\mathcal{F}_{t_0}$ at $t_0 + \Delta t$ is

$$\begin{aligned}
\mathbb{E}_{t_0}[\ln s_j(t_0+\Delta t)] &= \ln s_j(t_0) + \mathbb{E}_{t_0}[\Delta \ln s_j], \qquad \text{with} \quad \Delta \ln s_j = \ln s_j(t_0+\Delta t) - \ln s_j(t_0) \\
&= \ln s_j(t_0) + \ln\left[\frac{P_{\phi(j)}(0,s)}{P_{\phi(j)}(0,t)}\right] - \int_s^t q_j(u)du - \frac{1}{2}\int_s^t \sigma_j^S(u)\sigma_j^S(u)du \\
&\quad +\rho_{0j}^{zs}\int_s^t \alpha_0(u)H_0(u)\sigma_j^S(u)du - \epsilon_{\phi(j)}\rho_{j\phi(j)}^{sx}\int_s^t \sigma_j^S(u)\sigma_{\phi(j)}(u)du \\
&\quad +\frac{1}{2}\left(H_{\phi(j)}^2(t)\zeta_{\phi(j)}(t) - H_{\phi(j)}^2(s)\zeta_{\phi(j)}(s) - \int_s^t H_{\phi(j)}^2(u)\alpha_{\phi(j)}^2(u)du\right) \\
&\quad +(H_{\phi(j)}(t) - H_{\phi(j)}(s))z_{\phi(j)}(s) + \epsilon_{\phi(j)}\int_s^t \gamma_{\phi(j)}(u)(H_{\phi(j)}(t) - H_{\phi(j)}(u))du
\end{aligned}$$

The IR-IR covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$\begin{aligned}
\mathrm{Cov}[\Delta z_a, \Delta \ln x_b] &= \rho_{0a}^{zz}\int_s^t (H_0^z(t) - H_0^z)\,\alpha_0^z\,\alpha_a^z\,du \\
&\quad -\rho_{ab}^{zz}\int_s^t \alpha_a^z\,(H_b^z(t) - H_b^z)\,\alpha_b^z\,du \\
&\quad +\rho_{ab}^{zx}\int_s^t \alpha_a^z\,\sigma_b^x\,du.
\end{aligned}$$

The IR-FX covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$\begin{aligned}
\mathrm{Cov}[\Delta z_a, \Delta \ln x_b] &= \rho_{0a}^{zz}\int_s^t (H_0^z(t) - H_0^z)\,\alpha_0^z\,\alpha_a^z\,du \\
&\quad -\rho_{ab}^{zz}\int_s^t \alpha_a^z\,(H_b^z(t) - H_b^z)\,\alpha_b^z\,du \\
&\quad +\rho_{ab}^{zx}\int_s^t \alpha_a^z\,\sigma_b^x\,du.
\end{aligned}$$

The FX-FX covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
\mathrm{Cov}[\Delta \ln x_a, \Delta \ln x_b] \;=\; & \int_s^t (H_0^z(t) - H_0^z)^2 \, (\alpha_0^z)^2 \, du \\
& -\rho_{0a}^{zz} \int_s^t (H_a^z(t) - H_a^z) \, \alpha_a^z \, (H_0^z(t) - H_0^z) \, \alpha_0^z \, du \\
& -\rho_{0b}^{zz} \int_s^t (H_0^z(t) - H_0^z) \, \alpha_0^z \, (H_b^z(t) - H_b^z) \, \alpha_b^z \, du \\
& +\rho_{0b}^{zx} \int_s^t (H_0^z(t) - H_0^z) \, \alpha_0^z \sigma_b^x \, du \\
& +\rho_{0a}^{zx} \int_s^t (H_0^z(t) - H_0^z) \, \alpha_0^z \, \sigma_a^x \, du \\
& -\rho_{ab}^{zx} \int_s^t (H_a^z(t) - H_a^z) \, \alpha_a^z \sigma_b^x, du \\
& -\rho_{ba}^{zx} \int_s^t (H_b^z(t) - H_b^z) \, \alpha_b^z \, \sigma_a^x \, du \\
& +\rho_{ab}^{zz} \int_s^t (H_a^z(t) - H_a^z) \, \alpha_a^z \, (H_b^z(t) - H_b^z) \, \alpha_b^z \, du \\
& +\rho_{ab}^{xx} \int_s^t \sigma_a^x \, \sigma_b^x \, du
\end{aligned}
$$

The IR-INF covariance over the interval $[s, t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
\mathrm{Cov}[\Delta z_a, \Delta z_{I,b}] &= \rho_{ab}^{zI} \int_s^t \alpha_a(s) \alpha_{I,b}(s) ds \\
\mathrm{Cov}[\Delta z_a, \Delta y_{I,b}] &= \rho_{ab}^{zI} \int_s^t \alpha_a(s) H_{I,b}(s) \alpha_{I,b}(s) ds
\end{aligned}
$$

The FX-INF covariance over the interval $[s, t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
\mathrm{Cov}[\Delta x_a, \Delta z_{I,b}] \;=\; & \rho_{0b}^{zI} \int_s^t \alpha_0(s)(H_0(t) - H_0(s)) \alpha_{I,b}(s) ds \\
& -\rho_{ab}^{zI} \int_s^t \alpha_a(s)(H_a(t) - H_a(s)) \alpha_{I,b}(s) ds \\
& +\rho_{ab}^{xI} \int_s^t \sigma_a(s) \alpha_{I,b}(s) ds \\
\mathrm{Cov}[\Delta x_a, \Delta y_{I,b}] \;=\; & \rho_{0b}^{zI} \int_s^t \alpha_0(s)(H_0(t) - H_0(s)) H_{I,b}(s) \alpha_{I,b}(s) ds \\
& -\rho_{ab}^{zI} \int_s^t \alpha_a(s)(H_a(t) - H_a(s)) H_{I,b}(s) \alpha_{I,b}(s) ds \\
& +\rho_{ab}^{xI} \int_s^t \sigma_a(s) H_{I,b}(s) \alpha_{I,b}(s) ds
\end{aligned}
$$

The INF-INF covariance over the interval $[s, t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
\mathrm{Cov}[\Delta z_{I,a}, \Delta z_{I,b}] &= \rho^{II}_{ab} \int_s^t \alpha_{I,a}(s)\alpha_{I,b}(s)ds \\
\mathrm{Cov}[\Delta z_{I,a}, \Delta y_{I,b}] &= \rho^{II}_{ab} \int_s^t \alpha_{I,a}(s)H_{I,b}(s)\alpha_{I,b}(s)ds \\
\mathrm{Cov}[\Delta y_{I,a}, \Delta y_{I,b}] &= \rho^{II}_{ab} \int_s^t H_{I,a}(s)\alpha_{I,a}(s)H_{I,b}(s)\alpha_{I,b}(s)ds
\end{aligned}
$$

The equity/equity covariance over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
Cov\left[\Delta ln[s_i], \Delta ln[s_j]\right] &= \rho^{zz}_{\phi(i)\phi(j)} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_{\phi(j)}(t) \\
&\quad - H_{\phi(j)}(u))\alpha_{\phi(i)}(u)\alpha_{\phi(j)}(u)du \\
&\quad + \rho^{zs}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\sigma^S_j(u)du \\
&\quad + \rho^{zs}_{\phi(j)i} \int_s^t (H_{\phi(j)}(t) - H_{\phi(j)}(u))\alpha_{\phi(j)}(u)\sigma^S_i(u)du \\
&\quad + \rho^{ss}_{ij} \int_s^t \sigma^S_i(u)\sigma^S_j(u)du
\end{aligned}
$$

The equity/FX covariance over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
Cov\left[\Delta ln[s_i], \Delta ln[x_j]\right] &= \rho^{zz}_{\phi(i)0} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_0(t) - H_0(u))\alpha_{\phi(i)}(u)\alpha_0(u)du \\
&\quad - \rho^{zz}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_j(t) - H_j(u))\alpha_{\phi(i)}(u)\alpha_j(u)du \\
&\quad + \rho^{zx}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\sigma_j(u)du \\
&\quad + \rho^{sz}_{i0} \int_s^t (H_0(t) - H_0(u))\alpha_0(u)\sigma^S_i(u)du \\
&\quad - \rho^{sz}_{ij} \int_s^t (H_j(t) - H_j(u))\alpha_j(u)\sigma^S_i(u)du \\
&\quad + \rho^{sx}_{ij} \int_s^t \sigma^S_i(u)\sigma_j(u)du
\end{aligned}
$$

The equity/IR covariance over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
Cov\left[\Delta ln[s_i], \Delta z_j\right] &= \rho^{zz}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\alpha_j(u)du \\
&\quad + \rho^{sz}_{ij} \int_s^t \sigma^S_i(u)\alpha_j(u)du
\end{aligned}
$$

The equity/inflation covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on

$\mathcal{F}_{t_0}$) are as follows:

$$\begin{aligned} Cov\left[\Delta ln[s_i], \Delta z_{I,j}\right] &= \rho^{zI}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\alpha_{I,j}(u)du \\ &\quad + \rho^{sI}_{ij} \int_s^t \sigma_i^S(u)\alpha_{I,j}(u)du \\ Cov\left[\Delta ln[s_i], \Delta y_{I,j}\right] &= \rho^{zI}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)H_{I,j}(u)\alpha_{I,j}(u)du \\ &\quad + \rho^{sI}_{ij} \int_s^t \sigma_i^S(u)H_{I,j}(u)\alpha_{I,j}(u)du \end{aligned}$$

A.3 Exposures

In ORE we use the following exposure definitions

$$EE(t) = EPE(t) = \mathbb{E}^N\left[\frac{(NPV(t) - C(t))^+}{N(t)}\right] \tag{3}$$

$$ENE(t) = \mathbb{E}^N\left[\frac{(-NPV(t) + C(t))^+}{N(t)}\right] \tag{4}$$

where $NPV(t)$ stands for the netting set NPV and $C(t)$ is the collateral balance[9] at time t. Note that these exposures are expectations of values discounted with numeraire N (in ORE the Linear Gauss Markov model's numeraire) to today, and expectations are taken in the measure associated with numeraire N. These are the exposures which enter into unilateral CVA and DVA calculation, respectively, see next section. Note that we sometimes label the expected exposure (3) EPE, not to be confused with the Basel III Expected Positive Exposure below.

Basel III defines a number of exposures each of which is a 'derivative' of Basel's Expected Exposure:

Expected Exposure

$$EE_B(t) = \mathbb{E}[\max(NPV(t) - C(t), 0)] \tag{5}$$

Expected Positive Exposure

$$EPE_B(T) = \frac{1}{T}\sum_{t<T} EE_B(t) \cdot \Delta t \tag{6}$$

Effective Expected Exposure, recursively defined as running maximum

$$EEE_B(t) = \max(EEE_B(t - \Delta t), EE_B(t)) \tag{7}$$

Effective Expected Positive Exposure

$$EEPE_B(T) = \frac{1}{T}\sum_{t<T} EEE_B(t) \cdot \Delta t \tag{8}$$

[9] $C(t) > 0$ means that we have *received* collateral from the counterparty

The last definition, Effective EPE, is used in Basel documents since Basel II for Exposure At Default and capital calculation. Following [11, 12] the time averages in the EPE and EEPE calculations are taken over *the first year* of the exposure evolution (or until maturity if all positions of the netting set mature before one year).

To compute $EE_B(t)$ consistently in a risk-neutral setting, we compound (3) with the deterministic discount factor $P(t)$ up to horizon t:

$$EE_B(t) = \frac{1}{P(t)}\, EE(t)$$

Finally, we define another common exposure measure, the *Potential Future Exposure* (PFE), as a (typically high) quantile α of the NPV distribution through time, similar to Value at Risk but at the upper end of the NPV distribution:

$$PFE_\alpha(t) = \left(\inf\left\{x | F_t(x) \geq \alpha\right\}\right)^+ \tag{9}$$

where F_t is the cumulative NPV distribution function at time t. Note that we also take the positive part to ensure that PFE is a positive measure even if the quantile yields a negative value which is possible in extreme cases.

A.4 CVA and DVA

Using the expected exposures in A.3 unilateral discretised CVA and DVA are given by [20]

$$CVA = \sum_i PD(t_{i-1}, t_i) \times LGD \times EPE(t_i) \tag{10}$$

$$DVA = \sum_i PD_{Bank}(t_{i-1}, t_i) \times LGD_{Bank} \times ENE(t_i) \tag{11}$$

where

$$\begin{aligned}
EPE(t) &\text{ expected exposure (3)} \\
ENE(t) &\text{ expected negative exposure (4)} \\
PD(t_i, t_j) &\text{ counterparty probability of default in } [t_i; t_j] \\
PD_{Bank}(t_i, t_j) &\text{ our probability of default in } [t_i; t_j] \\
LGD &\text{ counterparty loss given default} \\
LGD_{Bank} &\text{ our loss given default}
\end{aligned}$$

Note that the choice t_i in the arguments of $EPE(t_i)$ and $ENE(t_i)$ means we are choosing the *advanced* rather than the *postponed* discretization of the CVA/DVA integral [15]. This choice can be easily changed in the ORE source code or made configurable.
Moreover, formulas (10, 11) assume independence of credit and other market risk factors, so that PD and LGD factors are outside the expectations. With the extension of ORE to credit asset classes and in particular for wrong-way-risk analysis, CVA/DVA formulas will be generalised.

A.5 FVA

Any exposure (uncollateralised or residual after taking collateral into account) gives rise to funding cost or benefits depending on the sign of the residual position. This can be expressed as a Funding Value Adjustment (FVA). A simple definition of FVA can be given in a very similar fashion as the sum of unilateral CVA and DVA which we defined by (10,11), namely as an expectation of exposures times funding spreads:

$$
\begin{aligned}
FVA = & \underbrace{\sum_{i=1}^{n} f_l(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left\{ S_C(t_{i-1})\, S_B(t_{i-1})\, [-NPV(t_i) + C(t_i)]^+ \, D(t_i) \right\}}_{\text{Funding Benefit Adjustment (FBA)}} \\
& - \underbrace{\sum_{i=1}^{n} f_b(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left\{ S_C(t_{i-1})\, S_B(t_{i-1})\, [NPV(t_i) - C(t_i)]^+ \, D(t_i) \right\}}_{\text{Funding Cost Adjustment (FCA)}} \qquad (12)
\end{aligned}
$$

where

$$
\begin{aligned}
D(t_i) & \text{ stochastic discount factor, } 1/N(t_i) \text{ in LGM} \\
NPV(t_i) & \text{ portfolio value at time } t_i \\
C(t_i) & \text{Collateral account balance at time } t_i \\
S_C(t_j) & \text{ survival probability of the counterparty} \\
S_B(t_j) & \text{ survival probability of the bank} \\
f_b(t_j) & \text{ borrowing spread for the bank relative to OIS flat} \\
f_l(t_j) & \text{ lending spread for the bank relative to OIS flat}
\end{aligned}
$$

For details see e.g. Chapter 14 in Gregory [18] and the discussion in [20].

The reasoning leading to the expression above is as follows. Consider, for example, a single partially collateralised derivative (no collateral at all or CSA with a significant threshold) between us (the Bank) and counterparty 1 (trade 1).
We assume that we enter into an offsetting trade with (hypothetical) counterparty 2 which is perfectly collateralised (trade 2). We label the NPV of trade 1 and 2 $NPV_{1,2}$ respectively (from our perspective, excluding CVA). Then $NPV_2 = -NPV_1$. The respective collateral amounts due to trade 1 and 2 are C_1 and C_2 from our perspective. Because of the perfect collateralisation of trade 2 we assume $C_2 = NPV_2$. The imperfect collateralisation of trade 1 means $C_1 \neq NPV_1$. The net collateral balance from our perspective is then $C = C_1 + C_2$ which can be written $C = C_1 + C_2 = C_1 + NPV_2 = -NPV_1 + C_1$.

- If $C > 0$ we receive net collateral and pay the overnight rate on this notional amount. On the other hand we can invest the received collateral and earn our lending rate, so that we have a benefit proportional to the lending spread f_l (lending rate minus overnight rate). It is a benefit assuming $f_l > 0$. $C > 0$ means $-NPV_1 + C_1 > 0$ so that we can cover this case with "lending notional" $[-NPV_1 + C_1]^+$.

- If $C < 0$ we post collateral amount $-C$ and receive the overnight rate on this amount. Amount $-C$ needs to be funded in the market, and we pay our borrowing rate on it. This leads to a funding cost proportional to the borrowing spread f_b

(borrowing rate minus overnight). $C < 0$ means $NPV_1 - C_1 > 0$, so that we can cover this case with "borrowing notional" $[NPV_1 - C_1]^+$. If the borrowing spread is positive, this term proportional to $f_b \times [NPV_1 - C_1]^+$ is indeed a cost and therefore needs to be subtracted from the benefit above.

Formula (12) evaluates these funding cost components on the basis of the original trade's or portfolio's NPV. Perfectly collateralised portfolios hence do not contribute to FVA because under the hedging fiction, they are hedged with a perfectly collateralised opposite portfolio, so any collateral payments on portfolio 1 are canceled out by those of the opposite sign on portfolio 2.

A.6 COLVA

When the CSA defines a collateral compounding rate that deviates from the overnight rate, this gives rise to another value adjustment labeled COLVA [20]. In the simplest case the deviation is just given by a constant spread Δ:

$$COLVA = \mathbb{E}^N \left[\sum_i -C(t_i) \cdot \Delta \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{13}$$

where $C(t)$ is the collateral balance[10] at time t and $D(t)$ is the stochastic discount factor $1/N(t)$ in LGM. Both $C(t)$ and $N(t)$ are computed in ORE's Monte Carlo framework, and the expectation yields the desired adjustment.
Replacing the constant spread by a time-dependent deterministic function in ORE is straight forward.

A.7 Collateral Floor Value

A less trivial extension of the simple COLVA calculation above, also covered in ORE, is the case where the deviation between overnight rate and collateral rate is stochastic itself. A popular example is a CSA under which the collateral rate is the overnight rate *floored at zero*. To work out the value of this CSA feature one can take the difference of discounted margin cash flows with and without the floor feature. It is shown in [20] that the following formula is a good approximation to the collateral floor value

$$\Pi_{Floor} = \mathbb{E}^N \left[\sum_i -C(t_i) \cdot (-r(t_i))^+ \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{14}$$

where r is the stochastic overnight rate and $(-r)^+ = r^+ - r$ is the difference between floored and 'un-floored' compounding rate.
Taking both collateral spread and floor into account, the value adjustment is

$$\Pi_{Floor,\Delta} = \mathbb{E}^N \left[\sum_i -C(t_i) \cdot ((r(t_i) - \Delta)^+ - r(t_i)) \cdot \delta_i \cdot D(t_{i+1}) \right] \tag{15}$$

[10] see A.3, $C(t) > 0$ means that we have *received* collateral from the counterparty

A.8 Dynamic Initial Margin and MVA

The introduction of Initial Margin posting in non-cleared OTC derivatives business reduces residual credit exposures and the associated value adjustments, **CVA** and **DVA**. On the other hand, it gives rise to additional funding cost. The value of the latter is referred to as Margin Value Adjustment (**MVA**).
To quantify these two effects one needs to model Initial Margin under future market scenarios, i.e. Dynamic Initial Margin (**DIM**). Potential approaches comprise

- Monte Carlo VaR embedded into the Monte Carlo simulation
- Regression-based methods
- Delta VaR under scenarios
- ISDA's Standard Initial Margin (SIMM) under scenarios

We skip the first option as too computationally expensive for ORE. In the current ORE release we focus on a relatively simple regression approach as in [21, 23]. Consider the netting set values $NPV(t)$ and $NPV(t+\Delta)$ that are spaced one margin period of risk Δ apart. Moreover, let $F(t, t+\Delta)$ denote cumulative netting set cash flows between time t and $t+\Delta$, converted into the NPV currency. Let $X(t)$ then denote the netting set value change during the margin period of risk excluding cash flows in that period:

$$X(t) = NPV(t+\Delta) + F(t, t+\Delta) - NPV(t)$$

ignoring discounting/compounding over the margin period of risk. We actually want to determine the distribution of $X(t)$ conditional on the 'state of the world' at time t, and pick a high (99%) quantile to determine the Initial Margin amount for each time t. Instead of working out the distribution, we content ourselves with estimating the conditional variance $\mathbb{V}(t)$ or standard deviation $S(t)$ of $X(t)$, assuming a normal distribution and scaling $S(t)$ to the desired 99% quantile by multiplying with the usual factor $\alpha = 2.33$ to get an estimate of the Dynamic Initial Margin DIM:

$$\mathbb{V}(t) = \mathbb{E}_t[X^2] - \mathbb{E}_t^2[X], \qquad S(t) = \sqrt{\mathbb{V}(t)}, \qquad DIM(t) = \alpha\, S(t)$$

We further assume that $\mathbb{E}_t[X]$ is small enough to set it to the expected value of $X(t)$ across all Monte Carlo samples X at time t (rather than estimating a scenario dependent mean). The remaining task is then to estimate the conditional expectation $\mathbb{E}_t[X^2]$. We do this in the spirit of the Longstaff Schwartz method using regression of $X^2(t)$ across all Monte Carlo samples at a given time. As a regressor (in the one-dimensional case) we could use $NPV(t)$ itself. However, we rather choose to use an adequate market point (interest rate, FX spot rate) as regression variable x, because this is generalised more easily to the multi-dimensional case. As regression basis functions we use polynomials, i.e. regression functions of the form $c_0 + c_1\, x + c_2\, x^2 + ... + c_n\, x^n$ where the order n of the polynomial can be selected by the user. Choosing the lowest order $n = 0$, we obtain the simplest possible estimate, the variance of X across all samples at time t, so that we apply a single $DIM(t)$ irrespective of the 'state of the world' at time t in that case. The extension to multi-dimensional regression is also implemented in ORE. The user can choose several regressors simultaneously (e.g. a EUR rate, a USD rate, USD/EUR spot FX rate, etc.) in order order to cover complex multi-currency portfolios.

Given the DIM estimate along all paths, we can next work out the Margin Value Adjustment [20] in discrete form

$$MVA = \sum_{i=1}^{n} (f_b - s_I)\, \delta_i\, S_C(t_i)\, S_B(t_i) \times \mathbb{E}^N \left[DIM(t_i)\, D(t_i) \right]. \tag{16}$$

with borrowing spread f_b as in the FVA section A.5 and spread s_I received on initial margin, both spreads relative to the cash collateral rate.

A.9 KVA (CCR)

The KVA is calculated for the Counterparty Credit Risk Capital charge (CCR) following the IRB method concisely described in [19], Appendix 8A. It is following the Basel rules by computing risk capital as the product of alpha weighted exposure at default, worst case probability of default at 99.9 and a maturity adjustment factor also described in the Basel annex 4. The risk capital charges are discounted with a capital discount factor and summed up to give the total CCR KVA after being multiplied with the risk weight and a capital charge (following the RWA method).

Basel II internal rating based (IRB) estimate of worst case probability of default: large homogeneous pool (LHP) approximation of Vasicek (1997), KVA regulatory probability of default is the worst case probability of default floored at 0.03 (the latter is valid for corporates and banks, no such floor applies to sovereign counterparties):

$$PD_{99.9\%} = \max\left(floor, N\left(\frac{N^{-1}(PD) + \sqrt{\rho} N^{-1}(0.999)}{\sqrt{1-\rho}} \right) - PD \right)$$

N is the cumulative standard normal distribution,

$$\rho = 0.12 \frac{1 - e^{-50PD}}{1 - e^{-50}} + 0.24 \left(1 - \frac{1 - e^{-50PD}}{1 - e^{-50}} \right)$$

Maturity adjustment factor for RWA method capped at 5, floored at 1:

$$MA(PD, M) = \min\left(5, \max\left(1, \frac{1 + (M - 2.5)B(PD)}{1 - 1.5B(PD)} \right) \right)$$

where $B(PD) = (0.11852 - 0.05478 \ln(PD))^2$ and M is the effective maturity of the portfolio (capped at 5):

$$M = \min\left(5, 1 + \frac{\sum\limits_{t_k > 1yr} EE_B(t_k) \Delta t_k B(0, t_k)}{\sum\limits_{t_k \leq 1yr} EEE_B(t_k) \Delta t_k B(0, t_k)} \right)$$

where $B(0, t_k)$ is the risk-free discount factor from the simulation date t_k to today, Δt_k is the difference between time points, $EE_B(t_k)$ is the expected (Basel) exposure at time t_k and $EEE_B(t_k)$ is the associated effective expected exposure.

Expected risk capital at t_i:

$$RC(t_i) = EAD(t_i) \times LGD \times PD_{99.9\%} \times MA(PD, M)$$

where

- $EAD(t_i) = \alpha \times EEPE(t_i)$
- $EEPE(t_i)$ is estimated as the time average of the running maximum of $EPE(t)$ over the time interval $t_i \leq t \leq t_i + 1$
- α is the multiplier resulting from the IRB calculations (Basel II defines a supervisory alpha of 1.4, but gives banks the option to estimate their own α,subject to a floor of 1.2).
- the maturity adjustment MA is derived from the EPE profile for times $t \geq t_i$

KVA_{CCR} is the sum of the expected risk capital amount discounted at *capital discount rate* r_{cd} and compounded at rate given by the product of *capital hurdle* h and *regulatory adjustment* a:

$$KVA_{CCR} = \sum_i RC(t_i) \times \frac{1}{(1+r_{cd})^{\delta(t_{i-1},t_i)}} \times \delta(t_{i-1}, t_i) \times h \times a$$

assuming Actual/Actual day count to compute the year factions *delta*.
In ORE we compute KVA CCR from both perspectives - "our" KVA driven by EPE and the counterparty default risk, and similarly "their" KVA driven by ENE and our default risk.

A.10 KVA (BA-CVA)

This section briefly summarizes the calculation of a capital value adjustment associated with the CVA capital charge (in the basic approach, BA-CVA) as introduced in Basel III [12, 13, 14]. ORE implements the *stand-alone* capital charge $SCVA$ for a netting set and computes a KVA for it[11]. In the basic approach, the stand-alone capital charge for a netting set is given by

$$SCVA = RW_c \cdot M \cdot EEPE \cdot DF$$

with

- supervisory risk weight RW_c for the counterparty;
- effective netting set maturity M as in section A.9 (for a bank using IMM to calculate EAD), but without applying a cap of 5;
- supervisory discount DF for the netting set which is equal to one for banks using IMM to calculate $EEPE$ and $DF = (1 - \exp(-0.05\, M))\,/(0.05\, M)$ for banks not using IMM to calculate $EEPE$.

[11] In the reduced version of BA-CVA, where hedges are not recognized, the total BA-CVA capital charge across all counterparties c is given by

$$K = \sqrt{\left(\rho \sum_c SCVA_c\right)^2 + (1-\rho^2) \sum_c SCVA_c^2}$$

with supervisory correlation $\rho = 0.5$ to reflect that the credit spread risk factors across counterparties are not perfectly correlated. Each counterparty $SCVA_c$ is given by a sum over all netting sets with this counterparty.

The associated capital value adjustment is then computed for each netting set's stand-alone CVA charge as above

$$KVA_{BA-CVA} = \sum_i SCVA(t_i) \times \frac{1}{(1+r_{cd})^{\delta(t_{i-1},t_i)}} \times \delta(t_{i-1}, t_i) \times h \times a$$

with

$$SCVA(t_i) = RW_c \cdot M(t_i) \cdot EEPE(t_i) \cdot DF$$

where we derive both M and EEPE from the EPE profile for times $t \geq t_i$.
In ORE we compute KVA BA-CVA from both perspectives - "our" KVA driven by EPE and the counterparty risk weight, and similarly "their" KVA driven by ENE and our risk weight.
Note: Banks that use the BA-CVA for calculating CVA capital requirements are allowed to cap the maturity adjustment factor $MA(PD, M)$ in section A.9 at 1 for netting sets that contribute to CVA capital, if using the IRB approach for CCR capital.

A.11 Collateral Model

The collateral model implemented in ORE is based on the evolution of collateral account balances along each Monte Carlo path taking into account thresholds, minimum transfer amounts and independent amounts defined in the CSA, as well as margin periods of risk. ORE computes the collateral requirement (aka *Credit Support Amount*) through time along each Monte Carlo path

$$CSA(t_m) = \begin{cases} \max(0, V_{set}(t_m) - I_A - T_{hold}), & V_{set}(t_m) - I_A \geq 0 \\ \min(0, V_{set}(t_m) - I_A + T_{hold}), & V_{set}(t_m) - I_A < 0 \end{cases} \tag{17}$$

where

- $V_{set}(t_m)$ is the value of the netting set as of time t_m,
- T_{hold} is the threshold exposure below which no collateral is required (possibly asymmetric),
- I_A is the sum of all collateral independent amounts attached to the underlying portfolio of trades (positive amounts imply that the bank has received a net inflow of independent amounts from the counterparty), assumed here to be cash.

As the collateral account already has a value of $C(t_m)$ at time t_m, the collateral shortfall is simply the difference between $C(t_m)$ and $CSA(t_m)$. However, we also need to account for the possibility that margin calls issued in the past have not yet been settled (for instance, because of disputes). If $M(t_m)$ denotes the net value of all outstanding margin calls at t_m, and $\Delta(t)$ is the difference $\Delta(t) = CSA(t_m) - C(t_m) - M(t_m)$ between the *Credit Support Amount* and the current and outstanding collateral, then the actual margin *Delivery Amount* $D(t_m)$ is calculated as follows:

$$D(t_m) = \begin{cases} \Delta(t), & |\Delta(t)| \geq MTA \\ 0, & |\Delta(t)| < MTA \end{cases} \tag{18}$$

where MTA is the minimum transfer amount.

Finally, the *Delivery Amount* is settled with a delay specified by the *Margin Period of Risk* (MPoR) which leads to residual exposure and XVA even for daily margining, zero thresholds and minimum transfer amounts, see for example [16]. A more detailed framework for collateralised exposure modelling is introduced in the 2016 article [22], indicating a potential route for extending ORE.

A.12 Exposure Allocation

XVAs and exposures are typically computed at netting set level. For accounting purposes it is typically required to *allocate* XVAs from netting set to individual trade level such that the allocated XVAs add up to the netting set XVA. This distribution is not trivial, since due to netting and imperfect correlation single trade (stand-alone) XVAs hardly ever add up to the netting set XVA: XVA is sub-additive similar to VaR. ORE provides an allocation method (labeled *marginal allocation* in the following) which slightly generalises the one proposed in [17]. Allocation is done pathwise which first leads to allocated expected exposures and then to allocated CVA/DVA by inserting these exposures into equations (10,11). The allocation algorithm in ORE is as follows:

- Consider the netting set's discounted NPV after taking collateral into account, on a given path at time t:

$$E(t) = D(0,t)\,(NPV(t) - C(t))$$

- On each path, compute contributions A_i of the latter to trade i as

$$A_i(t) = \begin{cases} E(t) \times NPV_i(t)/NPV(t), & |NPV(t)| > \epsilon \\ E(t)/n, & |NPV(t)| \leq \epsilon \end{cases}$$

  with number of trades n in the netting set and trade i's value $NPV_i(t)$.

- The EPE fraction allocated to trade i at time t by averaging over paths:

$$EPE_i(t) = \mathbb{E}\left[A_i^+(t)\right]$$

By construction, $\sum_i A_i(t) = E(t)$ and hence $\sum_i EPE_i(t) = EPE(t)$.
We introduced the *cutoff* parameter $\epsilon > 0$ above in order to handle the case where the netting set value $NPV(t)$ (almost) vanishes due to netting, while the netting set 'exposure' $E(t)$ does not. This is possible in a model with nonzero MTA and MPoR. Since a single scenario with vanishing $NPV(t)$ suffices to invalidate the expected exposure at this time t, the cutoff is essential. Despite introducing this cutoff, it is obvious that the marginal allocation method can lead to spikes in the allocated exposures. And generally, the marginal allocation leads to both positive and negative EPE allocations.

As a an example for a simple alternative to the marginal allocation of EPE we provide allocation based on today's single-trade CVAs

$$w_i = CVA_i / \sum_i CVA_i.$$

This yields allocated exposures proportional to the netting set exposure, avoids spikes and negative EPE, but does not distinguish the 'direction' of each trade's contribution to EPE and CVA.

A.13 Sensitivity Analysis

ORE's sensitivity analysis framework uses "bump and revalue" to compute Interest Rate, FX, Inflation, Equity and Credit sensitivities to

- Discount curves (in the zero rate domain)
- Index curves (in the zero rate domain)
- Yield curves including e.g. equity forecast yield curves (in the zero rate domain)
- FX Spots
- FX volatilities
- Swaption volatilities, ATM matrix or cube
- Cap/Floor volatility matrices (in the caplet/floorlet domain)
- Default probability curves (in the "zero rate" domain, expressing survival probabilities $S(t)$ in term of zero rates $z(t)$ via $S(t) = \exp(-z(t) \times t)$ with Actual/365 day counter)
- Equity spot prices
- Equity volatilities, ATM or including strike dimension
- Zero inflation curves
- Year-on-Year inflation curves
- CDS volatilities
- Base correlation curves

Apart from first order sensitivities (deltas), ORE computes second order sensitivities (gammas and cross gammas) as well. Deltas are computed using up-shifts and base values as

$$\delta = \frac{f(x+\Delta) - f(x)}{\Delta},$$

where the shift Δ can be absolute or expressed as a relative move Δ_r from the current level, $\Delta = x\,\Delta_r$. Gammas are computed using up- and down-shifts

$$\gamma = \frac{f(x+\Delta) + f(x-\Delta) - 2\,f(x)}{\Delta^2},$$

cross gammas using up-shifts and base values as

$$\gamma_{cross} = \frac{f(x+\Delta_x, y+\Delta_y) - f(x+\Delta_x, y) - f(x, y+\Delta_y) + f(x,y)}{\Delta_x\,\Delta_y}.$$

From the above it is clear that this involves the application of 1-d shifts (e.g. to discount zero curves) and 2-d shifts (e.g. to Swaption volatility matrices). The structure of the shift curves/matrices does not have to match the structure of the underlying data to be shifted, in particular the shift "curves/matrices" can be less granular than the market to be shifted. Figure 29 illustrates for the one-dimensional case how shifts are applied.

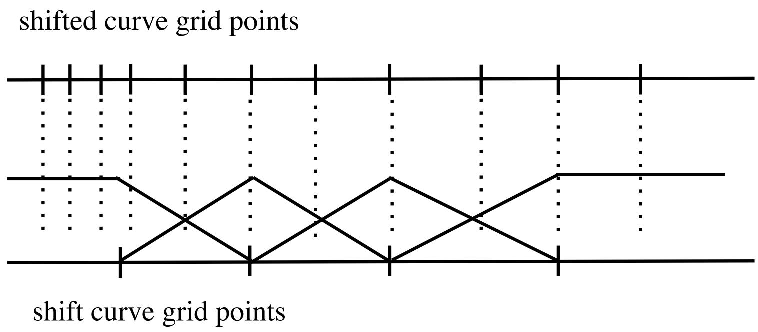

Figure 29: 1-d shift curve (bottom) applied to a more granular underlying curve (top).

Shifts at the left and right end of the shift curve are extrapolated flat, i.e. applied to all data of the original curve to the left and to the right of the shift curve ends. In between, all shifts are distributed linearly as indicated to the left and right up to the adjacent shift grid points. As a result, a parallel shift of the all points on the shift curve yields a parallel shift of all points on the underlying curve.
The two-dimensional case is covered in an analogous way, applying flat extrapolation at the boundaries and "pyramidal-shaped" linear interpolation for the bulk of the points.
The details of the computation of sensitivities to implied volatilities in strike direction can be summarised as follows, see also table 49 for an overview of the admissible configurations and the results that are obtained using them.

For *Swaption Volatilities*, the initial market setup can be an ATM surface only or a full cube. The simulation market can be set up to simulate ATM only or to simulate the full cube, but the latter choice is only possible if a full cube is set up in the initial market. The sensitivity set up must match the simulation setup with regards to the strikes (i.e. it is ATM only if and only if the simulation setup is ATM only, or it must contain exactly the same strike spreads relative to ATM as the simulation setup). Finally, if the initial market setup is a full cube, and the simulation / sensitivity setup is to simulate ATM only, then sensitivities are computed by shifting the ATM volatility w.r.t. the given shift size and type and shifting the non-ATM volatilities by the same absolute amount as the ATM volatility.

For *Cap/Floor Volatilities*, the initial market setup always contains a set of fixed strikes, i.e. there is no distinction between ATM only and a full surface. The same holds for the simulation market setup. The sensitivity setup may contain a different strike grid in this case than the simulation market. Sensitivity are computed per expiry and per strike in every case.

For *Equity Volatilities*, the initial market setup can be an ATM curve or a full surface. The simulation market can be set up to simulate ATM only or to simulate the full surface, where a full surface is allowed even if the initial market setup in an ATM curve only. If we have a full surface in the initial market and simulate the ATM curve only in the simulation market, sensitivities are computed as in the case of Swaption Volatilities, i.e. the ATM volatility is shifted w.r.t. the specified shift size and type and the non-ATM volatilities are shifted by the same absolute amount as the ATM volatility. If the simulation market is set up to simulate the full surface, then all volatilities are shifted individually using the specified shift size and type. In every case the sensitivities are aggregated on the ATM bucket in the sensitivity report.

For *FX Volatilities*, the treatment is similar to Equity Volatilities, except for the case of a full surface definition in the initial market and an ATM only curve in the simulation market. In this case, the pricing in the simulation market is using the ATM curve only, i.e. the initial market's smile structure is lost.

For *CDS Volatilities* only an ATM curve can be defined.

In all cases the smile dynamics is "sticky strike", i.e. the implied vol used for pricing a deal does not change if the underlying spot price changes.

Type	Init Mkt. Config.	Sim. Mkt Config.	Sensitivity Config.	Pricing	Sensitivities w.r.t.
Swaption	ATM	Simulate ATM only	Shift ATM only	ATM Curve	ATM Shifts
Swaption	Cube	Simulate Cube	Shift Smile Strikes	Full Cube	Smile Strike Shifts[a]
Swaption	Cube	Simulate ATM only	Shift ATM only	Full Cube	ATM Shifts[b]
Cap/Floor	Surface	Simulate Surface	Shift Smile Strikes	Full Surface	Smile Strike Shifts
Equity	ATM	Simulate ATM only	Shift ATM only	ATM Curve	ATM Shifts
Equity	ATM	Simulate Surface	Shift ATM only	ATM Curve	Smile Strike Shifts[c]
Equity	Surface	Simulate ATM only	Shift ATM only	Full Surface	ATM Shifts[b]
Equity	Surface	Simulate Surface	Shift ATM only	Full Surface	Smile Strike Shifts[c]
FX	ATM	Simulate ATM only	Shift ATM only	ATM Curve	ATM Shifts
FX	ATM	Simulate Surface	Shift ATM only	ATM Curve	Smile Strike Shifts[c]
FX	Surface	Simulate ATM only	Shift ATM only	ATM Curve	ATM Shifts
FX	Surface	Simulate Surface	Shift ATM only	Full Surface	Smile Strike Shifts[c]
CDS	ATM	Simulate ATM only	Shift ATM only	ATM Curve	ATM Shifts

Table 49: Admissible configurations for Sensitivity computation in ORE

[a] smile strike spreads must match simulation market configuration
[b] smile is shifted in parallel
[c] result sensitivities are aggregated on ATM

A.14 Value at Risk

For the computation of the parametric, or variance-covariance VaR, we rely on a second order sensitivity-based P&L approximation

$$\pi_S = \sum_{i=1}^{n} D^i_{T_i} V \cdot Y_i + \frac{1}{2} \sum_{i,j=1}^{n} D^{i,j}_{T_i,T_j} V \cdot Y_i \cdot Y_j \tag{19}$$

with

- portfolio value V
- random variables Y_i representing risk factor returns; these are assumed to be multivariate normally distributed with zero mean and covariance matrix matrix $C = \{\rho_{i,k}\sigma_i\sigma_k\}_{i,k}$, where σ_i denotes the standard deviation of Y_i; covariance matrix C may be estimated using the Pearson estimator on historical return data $\{r_i(j)\}_{i,j}$. Since the raw estimate might not be positive semidefinite, we apply a salvaging algorithm to ensure this property, which basically replaces negative Eigenvalues by zero and renormalises the resulting matrix, see [24];
- first or second order derivative operators D, depending on the market factor specific

shift type $T_i \in \{A, R, L\}$ (absolute shifts, relative shifts, absolute log-shifts), i.e.

$$\begin{aligned} D^i_A V(x) &= \frac{\partial V(x)}{\partial x_i} \\ D^i_R V(x) = D^i_L f(x) &= x_i \frac{\partial V(x)}{\partial x_i} \end{aligned}$$

and using the short hand notation

$$D^{i,j}_{T_i,T_j} V(x) = D^i_{T_i} D^j_{T_j} V(x)$$

In ORE, these first and second order sensitivities are computed as finite difference approximations ("bump and revalue").

To approximate the p-quantile of π_S in (19) ORE offers the techniques outlined below.

Delta Gamma Normal Approximation

The distribution of (19) is non-normal due to the second order terms. The delta gamma normal approximation in ORE computes mean m and variance v of the portfolio value change π_S (discarding moments higher than two) following [25] and provides a simple VaR estimate

$$VaR = m + N^{-1}(q)\sqrt{v}$$

for the desired quantile q (N is the cumulative standard normal distribution). Omitting the second order terms in (19) yields the delta normal approximation.

Monte Carlo Simulation

By simulating a large number of realisations of the return vector $Y = \{Y_i\}_i$ and computing the corresponding realisations of π_S in (19) we can estimate the desired quantile as the quantile of the empirical distribution generated by the Monte Carlo samples. Apart from the Monte Carlo Error no approximation is involved in this method, so that albeit slow it is well suited to produce values against which any other approximate approaches can be tested. Numerically, the simulation is implemented using a Cholesky Decomposition of the covariance matrix C in conjunction with a pseudo random number generator (Mersenne Twister) and an implementation of the inverse cumulative normal distribution to transform $U[0, 1]$ variates to $N(0, 1)$ variates.

References

[1] `http://www.opensourcerisk.org`

[2] `http://www.quantlib.org`

[3] `http://www.quaternion.com`

[4] `http://quantlib.org/install/vc10.shtml`

[5] `https://git-scm.com/downloads`

[6] `https://sourceforge.net/projects/boost/files/boost-binaries`

[7] http://www.boost.org

[8] http://jupyter.org

[9] https://docs.continuum.io/anaconda

[10] http://www.libreoffice.org

[11] Basel Committee on Banking Supervision, *International Convergence of Capital Measurement and Capital Standards, A Revised Framework*, http://www.bis.org/publ/bcbs128.pdf, June 2006

[12] Basel Committee on Banking Supervision, *Basel III: A global regulatory framework for more resilient banks and banking systems*, http://www.bis.org/publ/bcbs189.pdf, June 2011

[13] Basel Committee on Banking Supervision, *Review of the Credit Valuation Adjustment Risk Framework*, https://www.bis.org/bcbs/publ/d325.pdf, 2015

[14] Basel Committee on Banking Supervision, *Basel III: Finalising post-crisis reforms*, https://www.bis.org/bcbs/publ/d424.pdf, 2017

[15] Damiano Brigo and Fabio Mercurio, *Interest Rate Models: Theory and Practice, 2nd Edition*, Springer, 2006.

[16] Michael Pykhtin, *Collateralized Credit Exposure*, in Counterparty Credit Risk, (E. Canabarro, ed.), Risk Books, 2010

[17] Michael Pykhtin and Dan Rosen, *Pricing Counterparty Risk at the Trade Level and CVA Allocations*, Finance and Economics Discussion Series, Divisions of Research & Statistics and Monetary Affairs, Federal Reserve Board, Washington, D.C., 2010

[18] Jon Gregory, *Counterparty Credit Risk and Credit Value Adjustment, 2nd Ed.*, Wiley Finance, 2013.

[19] Jon Gregory, *The xVA Challenge, 3rd Ed.*, Wiley Finance, 2015.

[20] Roland Lichters, Roland Stamm, Donal Gallagher, *Modern Derivatives Pricing and Credit Exposure Analysis, Theory and Practice of CSA and XVA Pricing, Exposure Simulation and Backtesting*, Palgrave Macmillan, 2015.

[21] Fabrizio Anfuso, Daniel Aziz, Paul Giltinan, Klearchos Loukopoulos, *A Sound Modelling and Backtesting Framework for Forecasting Initial Margin Requirements*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2716279, 2016

[22] Leif B. G. Andersen, Michael Pykhtin, Alexander Sokol, *Rethinking Margin Period of Risk*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2719964, 2016

[23] Peter Caspers, Paul Giltinan, Paul; Lichters, Roland; Nowaczyk , Nikolai. *Forecasting Initial Margin Requirements A Model Evaluation*, Journal of Risk Management in Financial Institutions, Vol. 10 (2017), No. 4, https://ssrn.com/abstract=2911167

[24] R. Rebonato and P. Jaeckel, The most general methodology to create a valid correlation matrix for risk management and option pricing purposes, The Journal of Risk, 2(2), Winter 1999/2000, http://www.quarchome.org/correlationmatrix.pdf

[25] Carol Alexander, Market Risk Analysis, Volume IV, Value at Risk Models, Wiley 2009