ORE User Guide

Acadia Inc.

30 June 2021

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
20 March 2019	Quaternion	updates for release 4
19 June 2020	Quaternion	updates for release 5
30 June 2021	Acadia	updates for release 6

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] has been 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. Since Quaternion's acquisition by Acadia Inc. in February 2021, Acadia [4] is committed to continue the sponsorship. The Open Source Risk project works will continue with former Quaternion now operating as Acadia's Quantitative Services unit.

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 fifth release of ORE in June 2020 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 European and Asian Options	Y	Y	Y	Y
Equity Forwards	Y	Y	Y	Y
Equity Swaps	Y	Y	Y	N
Equity European and Asian Options	Y	Y	Y	Y
Equity Future Options	Y	Y	Y	Y
Commodity Forwards	Y	Y	N	N
Commodity European and Asian Options	Y	Y	N	N
CPI Swaps	Y	Y	N	Y
CPI Caps/Floors	Y	Y	N	N
Year-on-Year Inflation Swaps	Y	Y	N	Y
Year-on-Year Inflation Caps/Floors	Y	Y	N	N
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* [21]: 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, Dodgson-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 5 (June 2020) and 6 (June 2021).

INSTRUMENTS

- Add Equity, FX and Commodity Asian Options (thanks to Skandinaviska Enskilda Banken)
- Add Equity Future Option
- Support Quanto Equity Options
- Add Digital CMS Leg
- Add Duration Adjusted CMS Leg

- Allow settlement delay in FX Forwards
- Support separate caps/floors on inflation coupon and redemption
- CPI and YoY coupons can now pay the inflation ratio or the inflation rate = inflation ratio - 1
- Support caps / floors on ON index fixings, see LocalCapFloor tag in the User Guide
- Support local and global caps / floors on ON average coupons
- Reference commodity index in Commodity forwards
- Support strike of type yield-to-maturity in bond forwards (T-Locks etc.)
- Support averaging and in-advance fixing for SOFR/ON-Index linked legs
- Add LastRecentPeriodCalendar to handle SOFR30A, 90A, 180A
- Fix construction of notional amortization schedules
- Update CDS Option Pricing (O'Kane, 2008)
- Refine Cross Currency Swap pricing (thanks to Ioannis Rigopoulos)
- Allow for more than one premium, introduce new premium node (ORE ticket 1705)

MARKETS

- Add minor currency support (GBX, ZAC, ILA)
- Add currencies - AOA, ETB, GEL and XOF
- Add configurable currencies support to cover all remaining ISO currencies, see Examples/Input/currencies.xml
- Add configurable calendars
- Add various hard-coded IBOR Indices, DKK CITA and SEK STINA (thanks to Skandinaviska Enskilda Banken), AED EIBOR, GBP BoEBase, CNH HIBOR, CNY REPOFIX, HKD HONIA, JPY-Euroyen TIBOR, ZAR SABOR
- Add Inflation indices - AUCPI, FRCPI, BEHICP
- Add configurable, conventions based, IR/INF index support for greater flexibility in adding indexes to ORE
- Add LIBOR fallback support based on configurable adjustment spread, cessation date and RFR index
- Fix the FX triangulation to avoid spurious small FX sensitivities

TERM STRUCTURES

- Support 1M SOFR Futures in yield curve building
- Support CDS upfront quotes in default curve building

- Add support for delta-based Equity volatility surfaces
- Improve robustness of Cap/Floor volatility surfaces to missing data
- Support commodity volatility surfaces built from option prices
- Support flat zero extrapolation on yield curves
- Support linear zero interpolation for yield curves (thanks to Steven van Haren)
- Support delta interpolated FX volatility surfaces accepting BF/RR market data
- Support wild cards in FX volatility surface configurations
- Support Hermite interpolation in commodity price curves (thanks to Skandinaviska Enskilda Banken)
- Arbitrage checks for Equity and FX volatility surfaces
- Report to list market data points used in t0 market curve building
- Set up ScenarioSimMarket curves and surfaces as spreads over the t0 market
- Support fix-float mtm reset cross currency swaps in curve building
- Extend yield curve interpolation methods (quadratic, log-quadratic, Hermite, cubic spline)

ANALYTICS

- Introduce an optional close-out grid for exposure simulation and XVA, see Example 31, section 5.31
- Add inflation simulation for exposure and XVA using Jarrow-Yildirim, see Example 32, section 5.32
- Add credit simulation for exposure and XVA using Gaussian and Cox-Ingersoll-Ross models, see Examples 33 and 34, sections 5.33 and 5.34
- Add "flipped view" XVA feature, flipping "our" perspective to "theirs" (thanks to Roland Kapl), see Example 35, section 5.35
- Add option to switch from the LGM measure to the Bank Account measure for exposure simulations, see Example 36, section 5.36
- Refactoring of the exposure and XVA post processor, separating out various classes
- Support one-sided Initial Margin in CSAs
- Fix the treatment of independent amounts in Variation Margin calculation
- Export additional pricing engine and trade additional results for a range of instruments (Bonds, CDS, Swaps, Swaptions, Equity Forwards/Swaps/Options, FX Forwards and Options)
- Add an option to include past cashflows in the cashflow report
- Add accrual start/end dates and accrued amounts to the cash flow report (thanks to Roland Kapl)

- Fix LGM calibration with Best Fit
- Update trade notional/maturity in line with ISDA AANA/GRID recommendations

TESTS

- QuantExt: 203 test functions (vs 180 in the previous release)
- OREData: 160 test functions (vs 135 in the previous release)
- OREAnalytics: 65 test functions (vs 61 in the previous release)

with associated increases in the number of data driven test cases

EXAMPLES

- Added Examples 31 - 36, see the first five bullet points in the ANALYTICS section above
- Established regression tests for Example outputs

USER GUIDE

- Grown from 267 to 326 pages (thanks also to Roland Kapl and Skandinaviska Enskilda Banken)

LANGUAGE BINDUNGS

- Maintenance to ensure ORE SWIG wrappers build with current ORE, QuantLib-1.22 and QuantLib-SWIG-1.22

OTHER

- Fix the instrument schema to include ore_types.xsd again (thanks to Tomass Wilson)

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 outputs 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 the 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

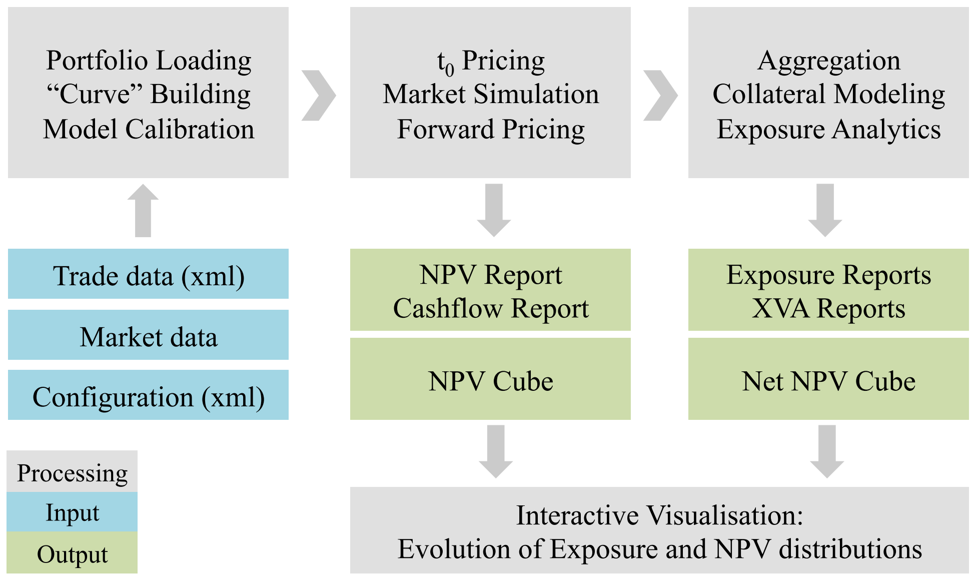


Figure 1: Sketch of the ORE process, inputs and outputs.

- 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 outputs 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 contains the QuantLib source version that ORE depends on. This is the latest QuantLib release that precedes the ORE release including a small number of patches.

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. `ORETest/`
8. `QuantExt/`
9. `QuantLib/`
10. `ThirdPartyLibs/`
11. `tools/`
12. `xsd/`
13. `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 contains the ORE source code, which is sufficient to build ORE from sources 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
- Build QuantLib, 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 RapidXml, 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 at `https://github.com/opensourcerisk/engine`.

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 [6]
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
   ```

Note that one can also run

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

in step 2, which also performs the steps in 3.

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 5.

Windows

1. Download the pre-compiled binaries for MSVC-14 (MSVC2015) from [7]
   - 32-bit: [7]\VERSION\boost_VERSION-msvc-14.0-32.exe\download
   - 64-bit: [7]\VERSION\boost_VERSION-msvc-14.0-64.exe\download
2. Start the installation file and choose an installation folder (the "boost root directory"). 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 [8] 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. Configure boost paths:

   Since the release of Visual Studio 2019, the user property files 'Microsoft.Cpp.Win32.user' and 'Microsoft.Cpp.x64.user' are no longer created on installation. There are now 2 ways to configure the boost paths for Visual Studio.

   (a) Set environment variables:

      - `%BOOST%` pointing to your directory, e.g, `C:\boost_1_72_0`
      - `%BOOST_LIB32%` pointing to your Win32 lib directory, e.g, `C:\boost_1_72_0\lib32msvc14.0`
      - `%BOOST_LIB64%` pointing to your x64 lib directory, e.g, `C:\boost_1_72_0\lib64msvc14.0`

      These are used by the ore user.props files.

   (b) Create you own property file and add as a property sheet following the instructions at https://docs.microsoft.com/en-us/cpp/build/create-reusable-property-configurations?view=msvc-160&viewFallbackFrom=vs-2019

   For older versions of Visual Studio you can use the Microsoft user property files. 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 'quantexttestsuite' 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 (the directory that contains the *.lib files) to the 'Library Directories' entry. If for example your boost installation is in `C:\boost_1_72_0` and the libraries reside in the `stage\lib` subfolder, add `C:\boost_1_72_0` to the 'Include Directories' entry and `C:\boost_1_72_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_72_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 the `oreEverything_*.sln` and build the entire solution (again, make sure to select the correct platform in the configuration manager first).

Unix

With the 5th release we have discontinued automake support so that ORE can only 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, QuantLib, as well as the doxygen API documentation for QuantExt, OREData and OREAnalytics, by invoking

   ```
   make -j4
   ```

   using four threads in this example.
4. Run all test suites by invoking

   ```
   ctest -j4
   ```
5. Run Examples (see section 5)

Note:

- If the boost libraries are not installed in a standard path 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`.
- Unset `LD_LIBRARY_PATH` respectively `DYLD_LIBRARY_PATH` before running the ORE executable or the test suites, in order not to override the rpath information embedded into the libaries built with CMake
- 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.
- Generate `CMakeLists.txt`:

  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. The python script (in `Tools/update_cmake_files.py`) can be used to update all CMakeLists.txt files automatically.

Building on Windows with CMake

One advantage of the CMake build system is that it covers both Unix and Windows builds. The same set of `CMakeLists.txt` files as above 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 CMAKE_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 use the ORE Python language bindings in section 4.4, or to run the examples in section 5 and plot exposure evolutions. Moreover, we use Jupyter [9] in section 6 to visualise simulation results. Both are part of the 'Anaconda Open Data Science Analytics Platform' [10]. 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

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

[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.

- 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.

4.4 Building ORE-SWIG

Since release 4, ORE comes with Python and Java language bindings following the QuantLib-SWIG example. The ORE bindings extend the QuantLib SWIG wrappers and allow calling ORE functionality in the QuantExt/OREData/OREAnalytics libraries alongside with functionality in QuantLib.

The ORE-SWIG source code is hosted in a separate git repository at https://github.com/opensourcerisk/ore-swig.

To build the wrappers on Windows, Linux, mac OS

1. build ORE and QuantLib first, as shown in the previous section
2. check out the ORE-SWIG repository into a project directory `oreswig`, change into that directory
3. pull in the QuantLib-SWIG project by running
   `git submodule init`
   `git submodule update`
4. Create a subdirectry `build`, change into that directory and configure cmake and then build using Ninja as follows

```
cmake -G Ninja \
      -D ORE=<ORE Root Directory> \
      -D BOOST_ROOT=<Top level boost include directory> \
      -D BOOST_LIBRARYDIR=<Location of the compiled boost libraries> \
      -D PYTHON_LIBRARY=<Full name including path to the 'libpython*' library>
\
      -D PYTHON_INCLUDE_DIR=<Directory that contains Python.h> \
      ..
ninja
```

   On Linux or mac OS one can also use `make` instead of `ninja`. In that case, omit the `-G Ninja` part in the configuration.

   To build on Windows using CMake and an existing Visual Studio installation you can e.g. run this from the top-level oreswig directory

```
mkdir build \
cmake -G "Visual Studio 15 2017" \
      -A x64 \
      -D SWIG_DIR=C:\dev\swigwin\Lib \
      -D SWIG_EXECUTABLE=C:\dev\swigwin\swig.exe \
      -D ORE:PATHNAME=C:\dev\ORE\master \
      -D BOOST_ROOT=C:\dev\boost \
      -S OREAnalytics-SWIG/Python \
      -B build \
cmake -build build -v
```

By default this builds the OREAnalytics-Python bindings which include the wrapped parts of QuantLib, QuantExt, OREData and OREAnalytics.

5. Try a Python example: Update your PYTHONPATH environment variable to include directory `oreswig/build/OREAnalytics-SWIG/Python` which contains both the new python module and the associated native library loaded by the python module; change to the `oreswig/OREAnalytics/Python/Examples` directory and run e.g.

   ```
   python ore.py
   ```

   There is also an IPython example in the same directory. To try it, launch

   ```
   jupyter notebook
   ```

   wait for your browser to open, select `ore.ipy` from the list of files and then run all cells.

6. Try a Java example: Make sure that the line `add_subdirectory(OREAnalytics-SWIG/Java)` is uncommented in the top-level CMakeLists.txt file when building; change to directory `ORE-SWIG/OREAnalytics-SWIG/Java/Examples` and run

   ```
   java -Djava.library.path=../../../build/OREAnalytics-SWIG/Java \
         -jar ../../../build/OREAnalytics-SWIG/Java/ORERunner.jar \
         Input/ore.xml
   ```

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

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 under Dodgson-Kainth
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
30	USD-Prime Curve Building via Prime-LIBOR Basis Swap
31	Exposure Simulation using a Close-out Grid
32	Inflation Swap Exposure under Jarrow-Yildrim
33	CDS Exposure Simulation
34	Wrong Way Risk
35	Flip View
36	Choice of Measure

Table 2: ORE examples.

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 `App/bin/win32/` respectively `App/bin/x64/`. Otherwise the examples may be run using the pre-compiled executables which come with the ORE release.

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

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
`dividends.txt`	Dividends 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

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.4. In this example, the expected exposure at 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 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

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

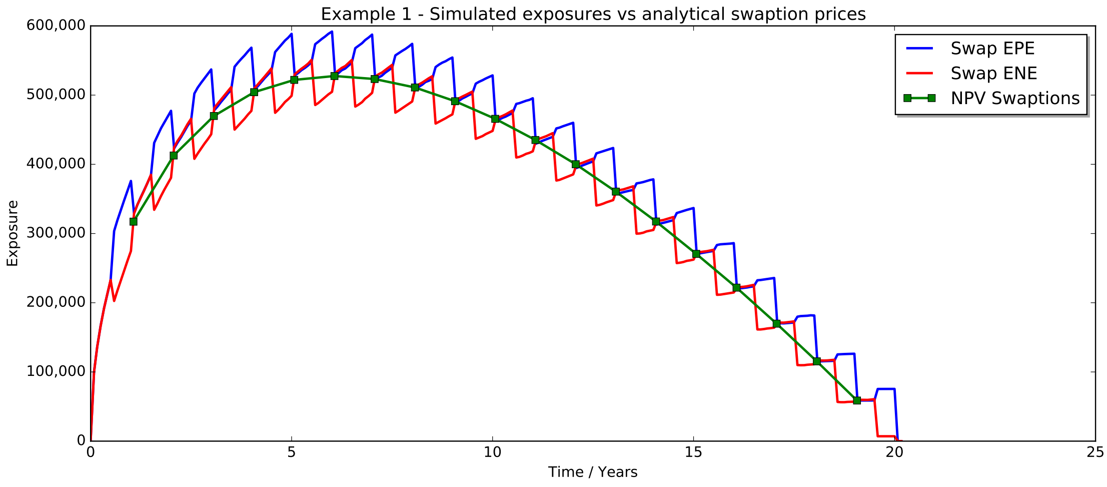


Figure 2: Vanilla ATM Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are European Swaption prices. The simulation was run with monthly time steps and 10,000 Monte Carlo samples to demonstrate the convergence of EPE and ENE profiles. A similar outcome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

figure 2. In Example_1 we used co-terminal ATM Swaptions for both model calibration and Swaption pricing. Moreover, as 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 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 (cash vs physical) yields significantly different valuations as of today due

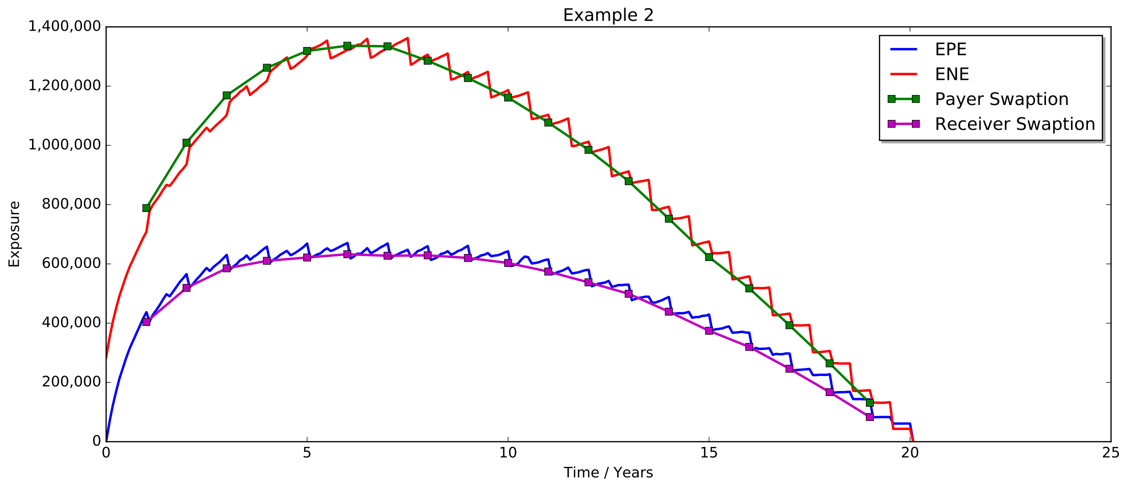


Figure 3: Vanilla ATM Swap expected exposure in a realistic market environment as of 05/02/2016 from both parties' perspectives. The Swap is the same as in figure 2 but receiving fixed 1%, roughly at the money. The symbols are the prices of European payer and receiver Swaptions. Simulation with 5000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

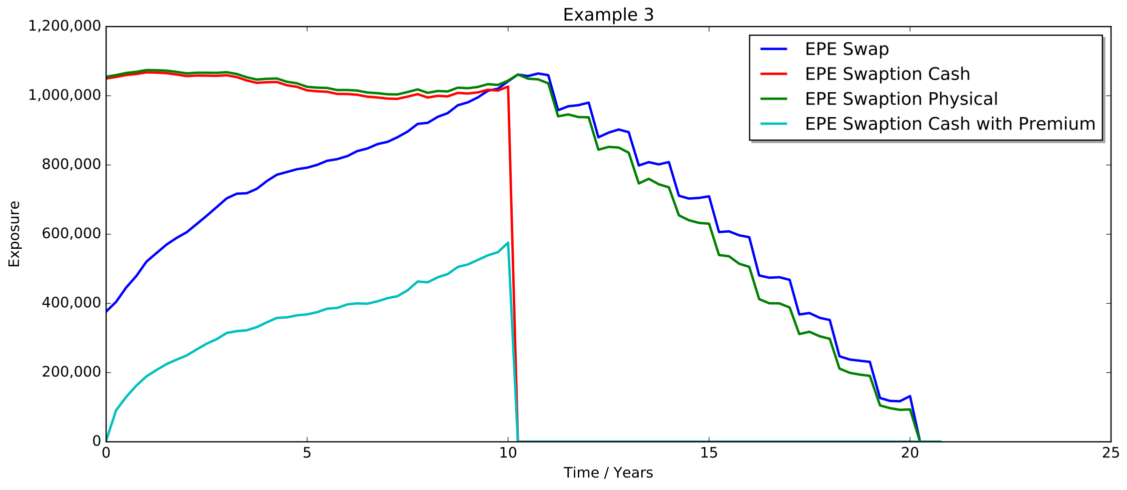


Figure 4: European Swaption exposure evolution, expiry in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

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

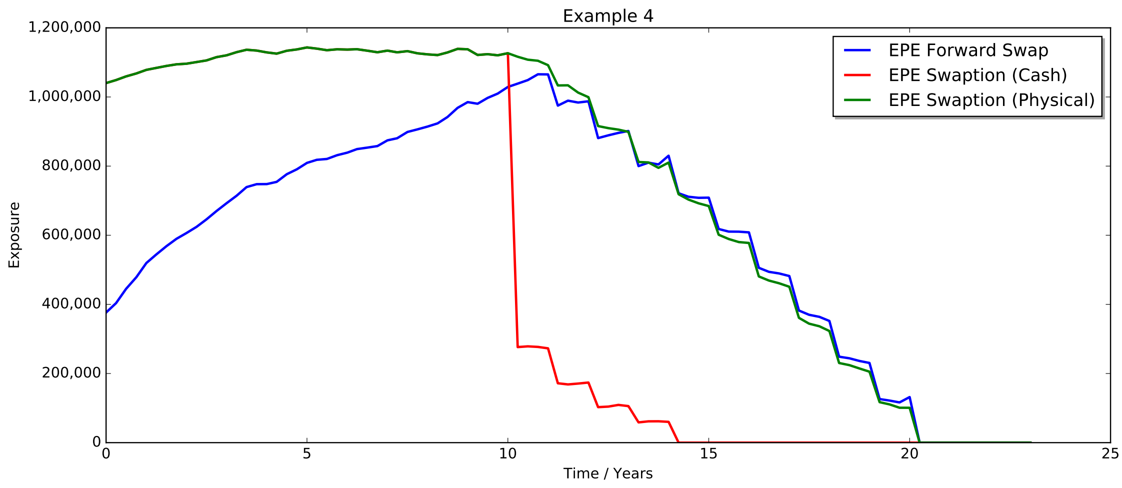


Figure 5: Bermudan Swaption exposure evolution, 5 annual exercise dates starting in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

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 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.

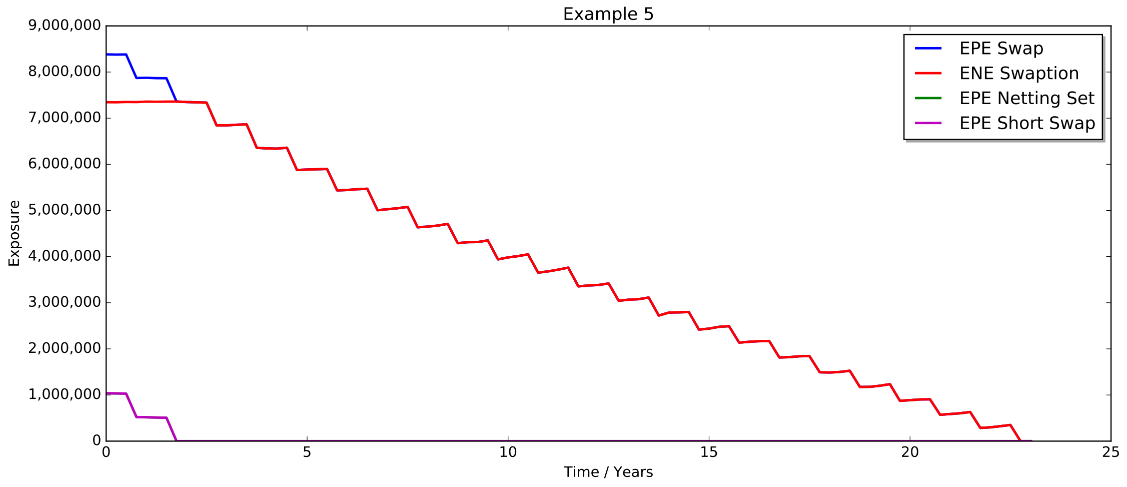


Figure 6: European callable Swap represented as a package consisiting of non-callable Swap and Swaption. The Swaption has physical delivery and offsets all future Swap cash flows if exercised. The exposure evolution of the package is shown here as 'EPE Netting Set' (green line). This is covered by the 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.

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.

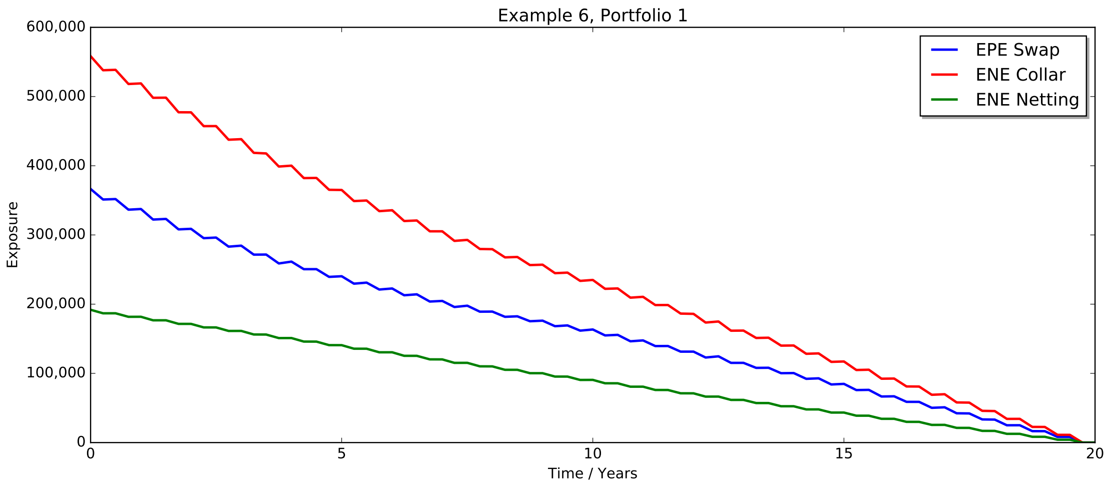


Figure 7: Swap+Collar, portfolio 1. The Collar has identical cap and floor rates at 4% so that it corresponds to a fixed leg which reduces the exposure of the Swap, which receives 3% fixed. Simulation with 1000 paths and quarterly time steps.

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.

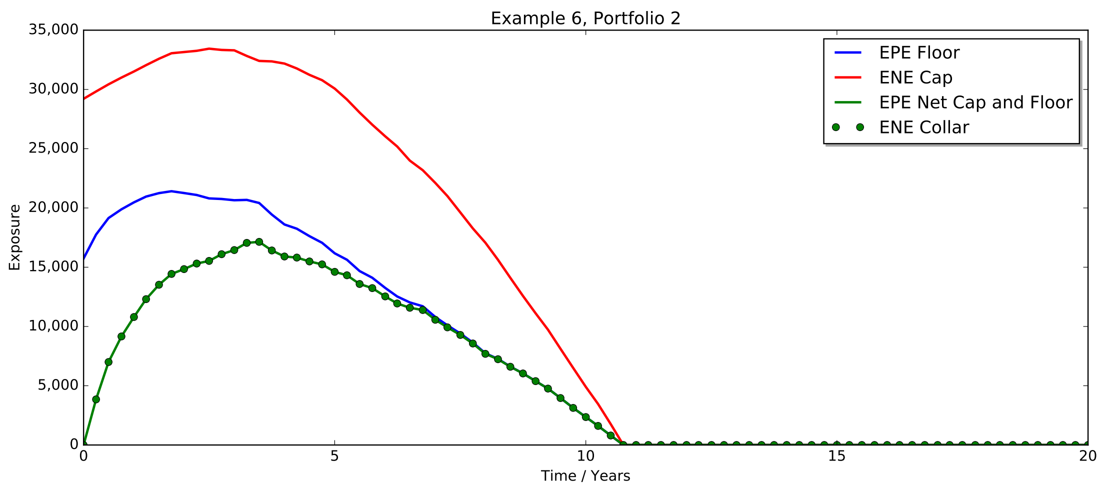


Figure 8: Short Cap and long Floor vs long Collar, portfolio 2. Simulation with 1000 paths and quarterly time steps.

Further three test portfolios are provided as part of this example. Run the example and inspect the respective output directories `Examples/Example_6/Output/portfolio_#`. Note that these directories have to be present/created before running the batch with `python run.py`.

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 shown in figure 9.

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

$$\frac{NPV(t)}{N(t)} = \text{Nominal} \times \mathbb{E}_t \left[\frac{(X(T)-K)^+}{N(T)} \right]$$
$$NPV(0) = \mathbb{E} \left[\frac{NPV(t)}{N(t)} \right] = \mathbb{E} \left[\frac{NPV^+(t)}{N(t)} \right] = EPE(t)$$

where $N(t)$ denotes the numeraire asset. 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

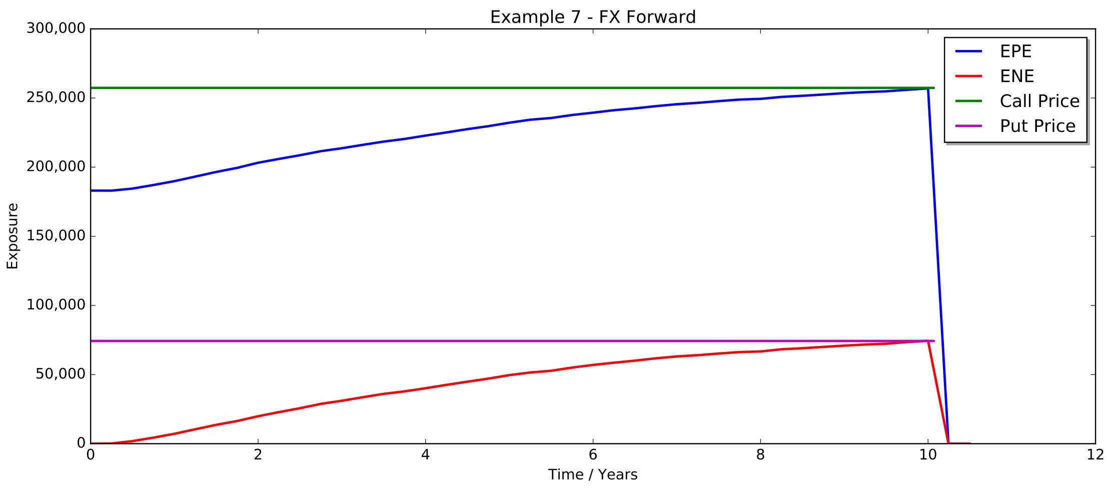


Figure 9: EUR/USD FX Forward expected exposure in a realistic market environment as of 26/02/2016 from both parties' perspectives. Value date is obviously in 10Y. The flat lines are FX Option prices which coincide with EPE and ENE, respectively, on the value date. Simulation with 5000 paths and quarterly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

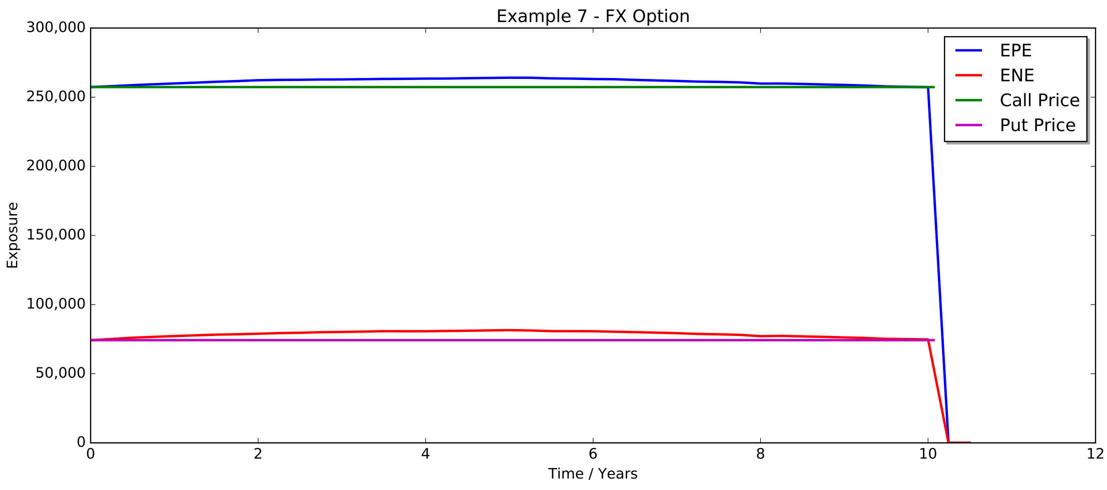


Figure 10: EUR/USD FX Call and Put Option exposure evolution, same underlying and market data as in section 5.7, compared to the call and put option price as of today (flat line). Simulation with 5000 paths and quarterly time steps.

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.

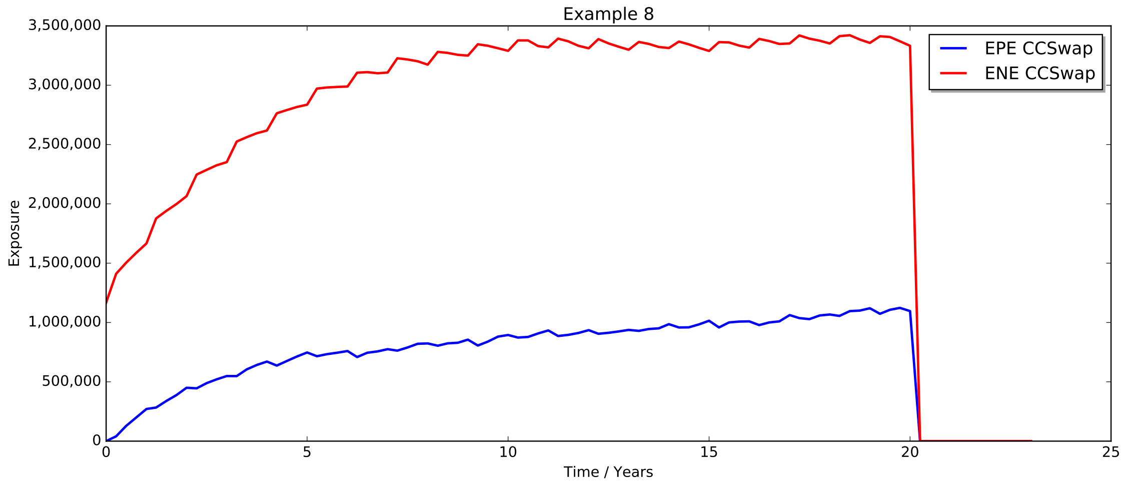


Figure 11: Cross Currency Swap exposure evolution without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

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.

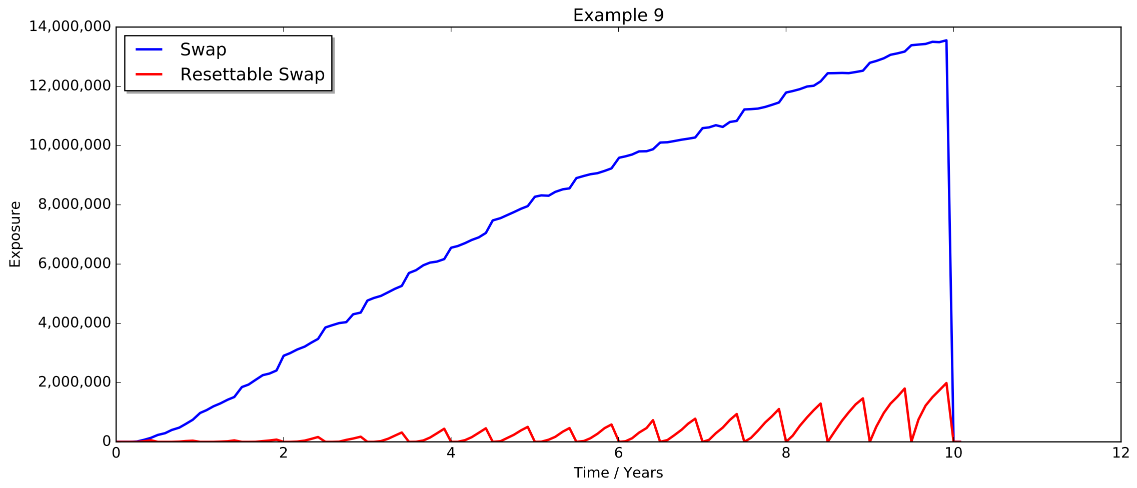


Figure 12: Cross Currency Basis Swap exposure evolution with and without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

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.12, 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 [23] for a recent detailed discussion of collateral modelling. The approach currently implemented in ORE corresponds to *Classical+* in [23], 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.

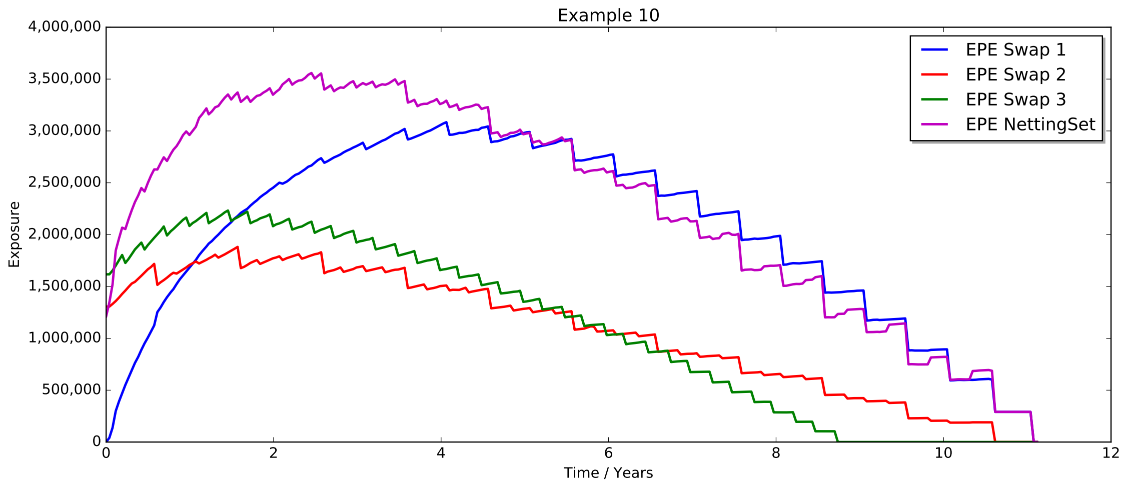


Figure 13: Three Swaps netting set, no collateral. Simulation with 5000 paths and bi-weekly time steps.

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.

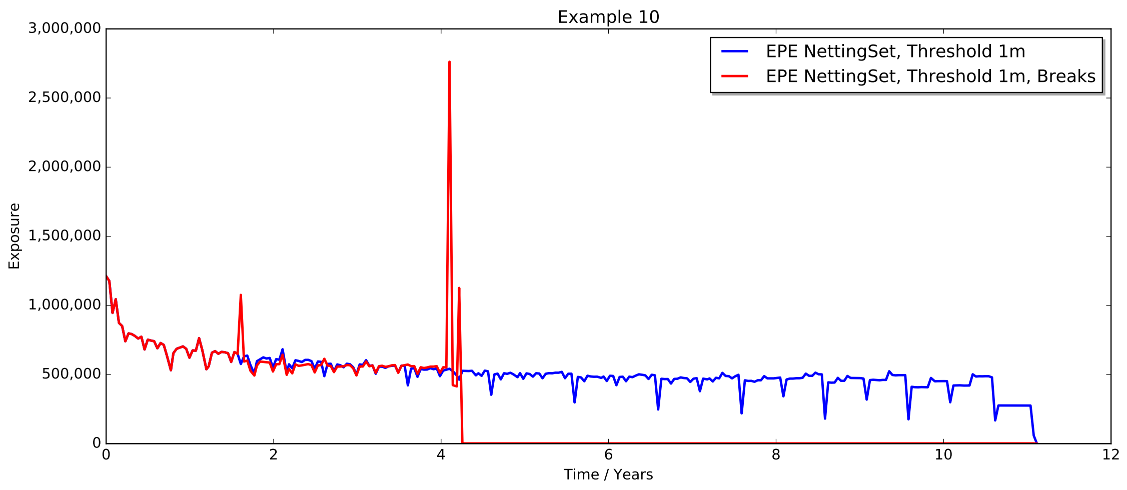


Figure 14: Three Swaps netting set, THR=1m EUR, MTA=100k EUR, MPOR=2w. The red evolution assumes that the each trade is terminated at the next break date. The blue evolution ignores break dates. Simulation with 5000 paths and bi-weekly time steps.

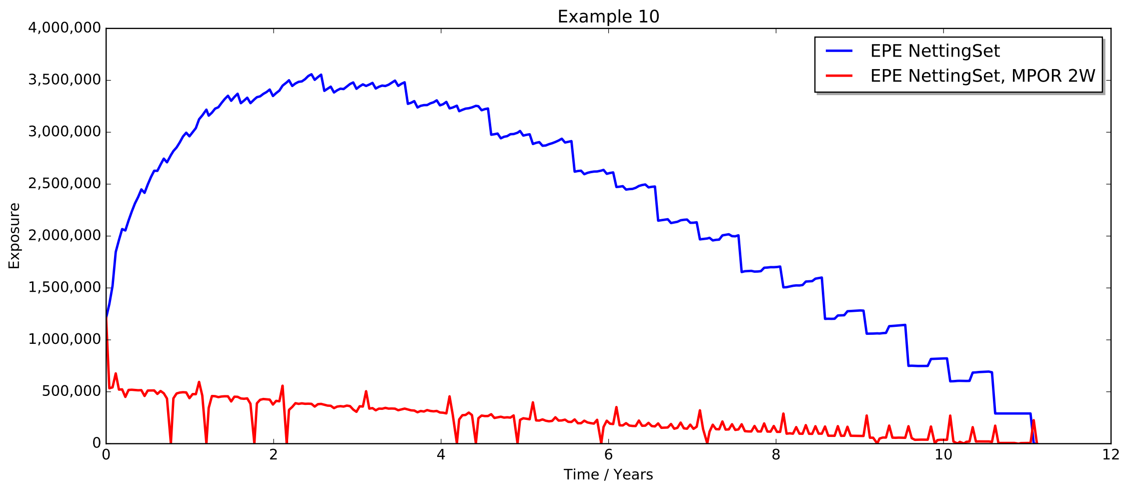


Figure 15: Three Swaps, THR=MTA=0, MPOR=2w. Simulation with 5000 paths and bi-weekly time steps.

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.

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.4. Allocated exposures are defined in appendix A.13. 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

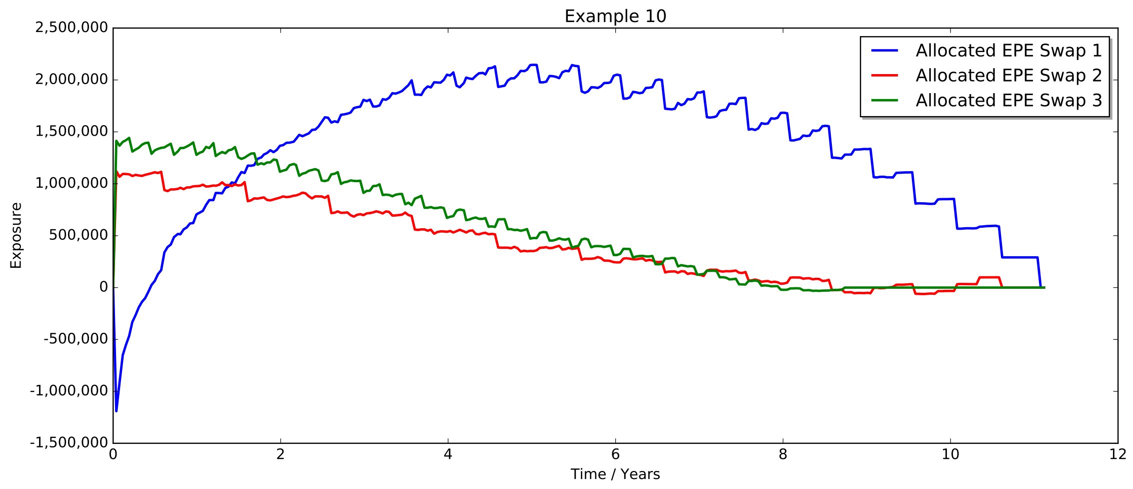


Figure 16: Exposure allocation without collateral. Simulation with 5000 paths and bi-weekly time steps.

marginal (Euler) allocation method as 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`

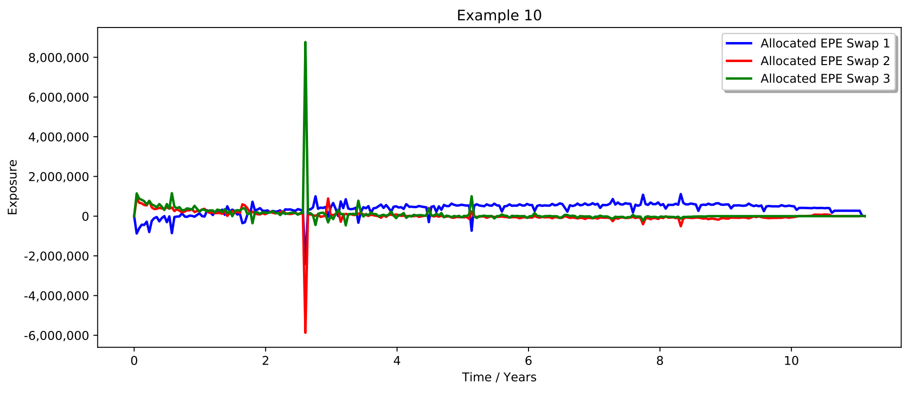


Figure 17: Exposure allocation with collateral and threshold 1m EUR. Simulation with 5000 paths and bi-weekly time steps.

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`.

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.4. 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.

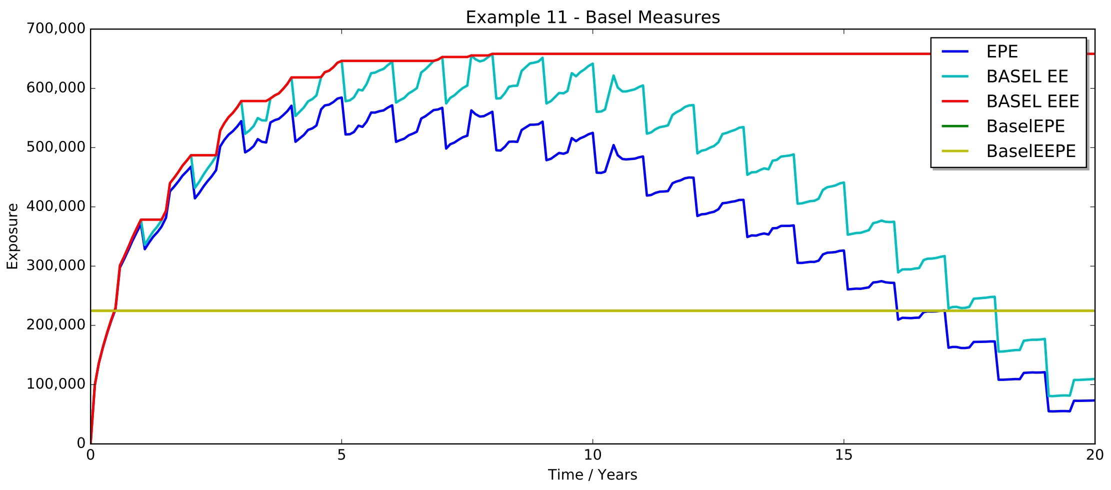


Figure 18: Evolution of the expected exposure of Vanilla Swap, comparison to the 'Basel' exposure measures EEE_B, EPE_B and EEPE_B. Note that the latter two are indistinguishable in this case, because the expected exposure is increasing for the first year.

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:

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

It can be shown that this leaves all expectations computed in the model (such as EPE and ENE) invariant. As explained in [21], 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.9) for a number of elementary products:

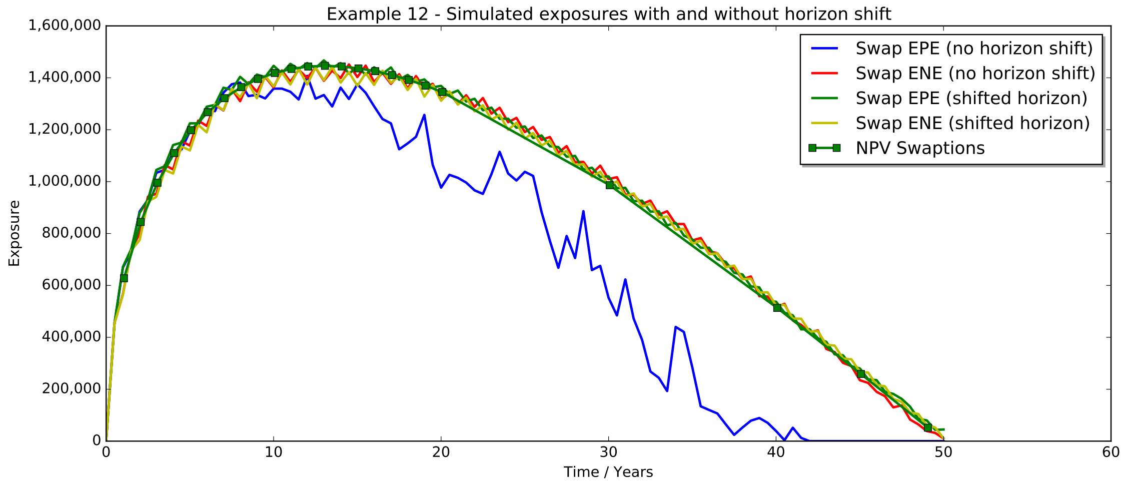


Figure 19: Long term Swap exposure simulation with and without horizon shift.

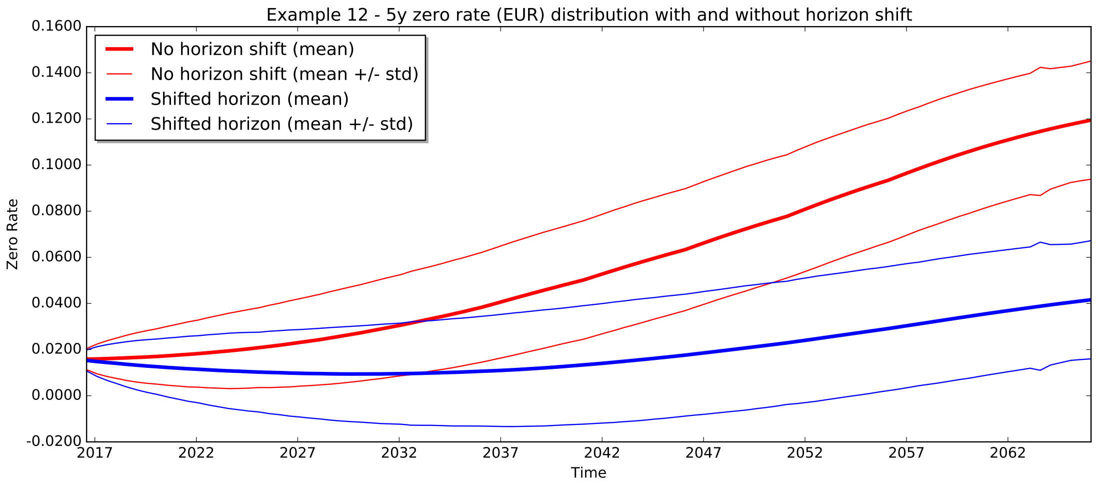


Figure 20: Evolution of rate distributions with and without horizon shift (change of measure). Thick lines indicate mean values, thin lines are contours of the rate distribution at $\pm$ one standard devation.

- A single currency Swap in EUR (case A),
- 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 [22, 24].

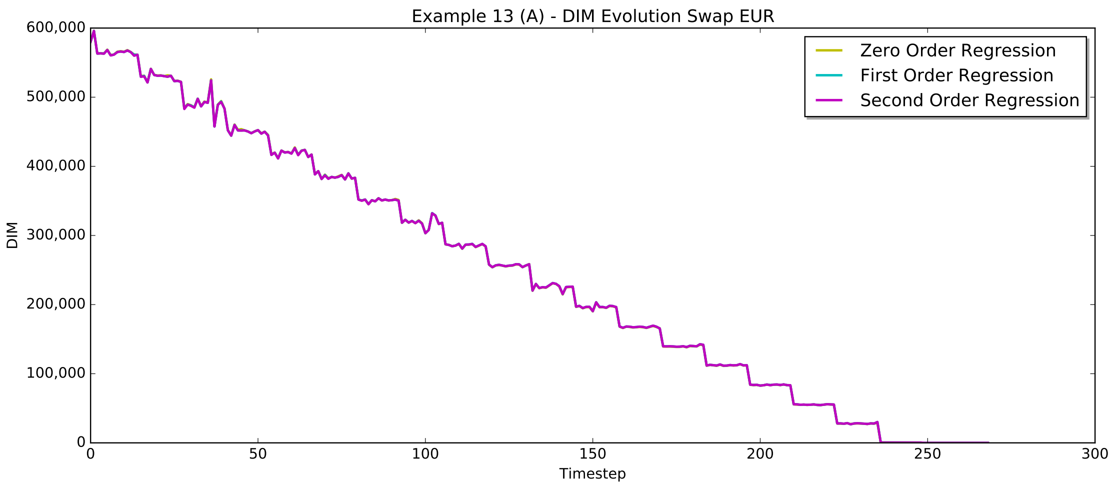


Figure 21: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swap of Example 13 A. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order (first and second order curves are not distinguishable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

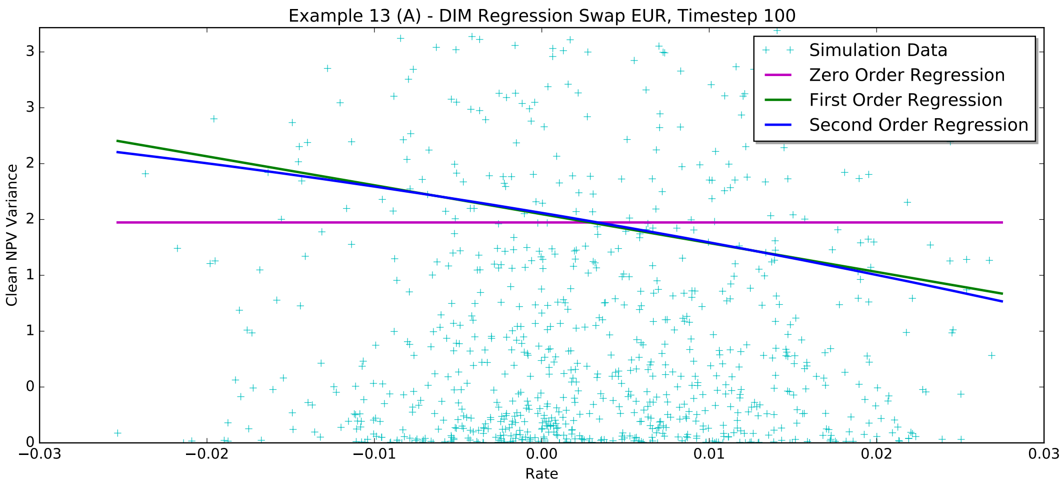


Figure 22: Regression snapshot at time step 100 for the EUR Swap of Example 13 A.

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

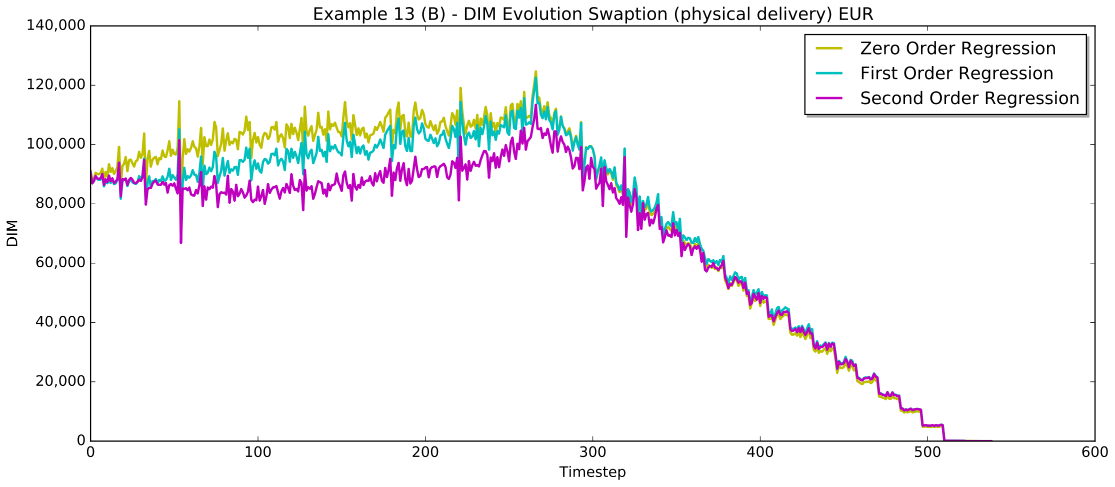


Figure 23: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swaption of Example 13 B with expiry in 10Y around time step 100. DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order. The simulation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

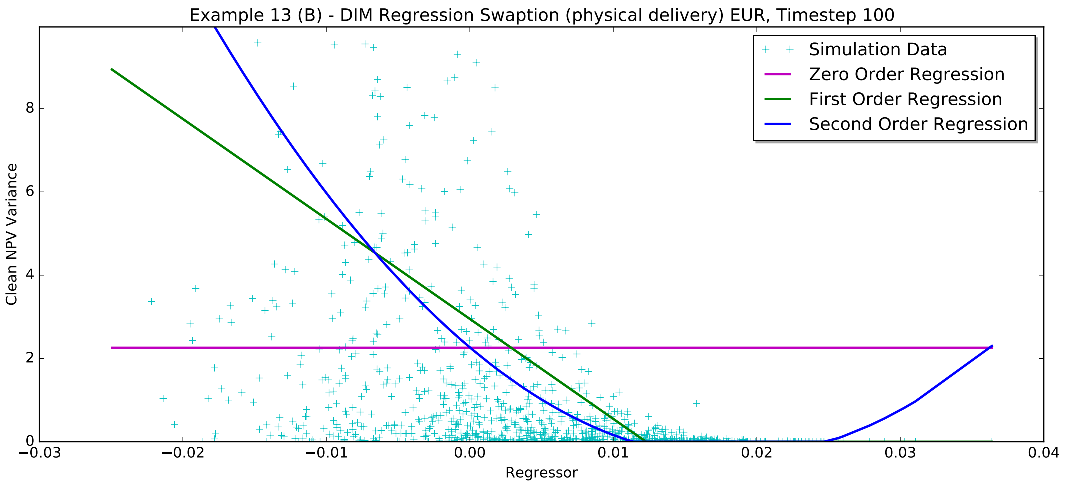


Figure 24: Regression snapshot at time step 100 (before expiry) for the EUR Swaption of Example 13 B.

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.

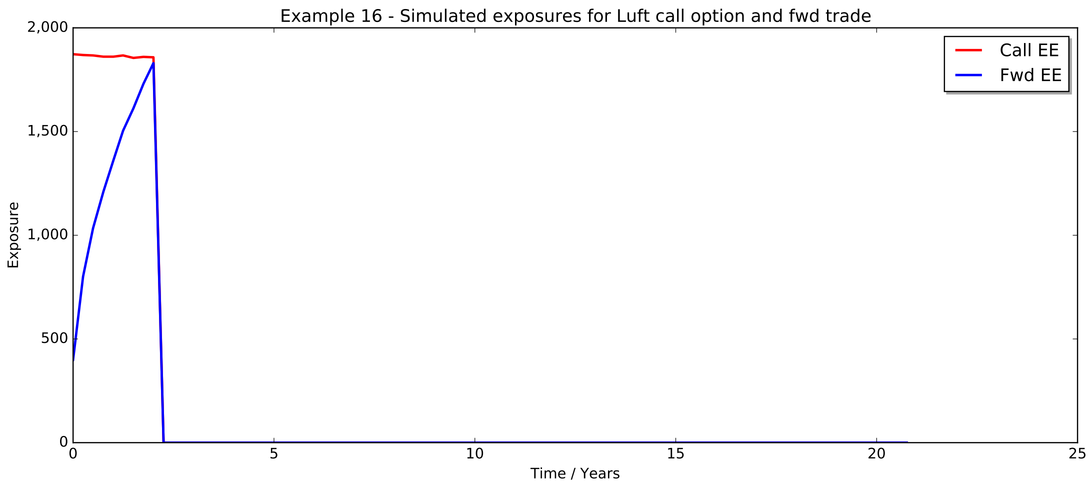


Figure 25: Equity (“Luft”) call option and OTC forward exposure evolution, maturity in approximately 2.5 years. Simulation with 10000 paths and quarterly time steps.

5.17 Inflation Swap Exposure under Dodgson-Kainth

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.

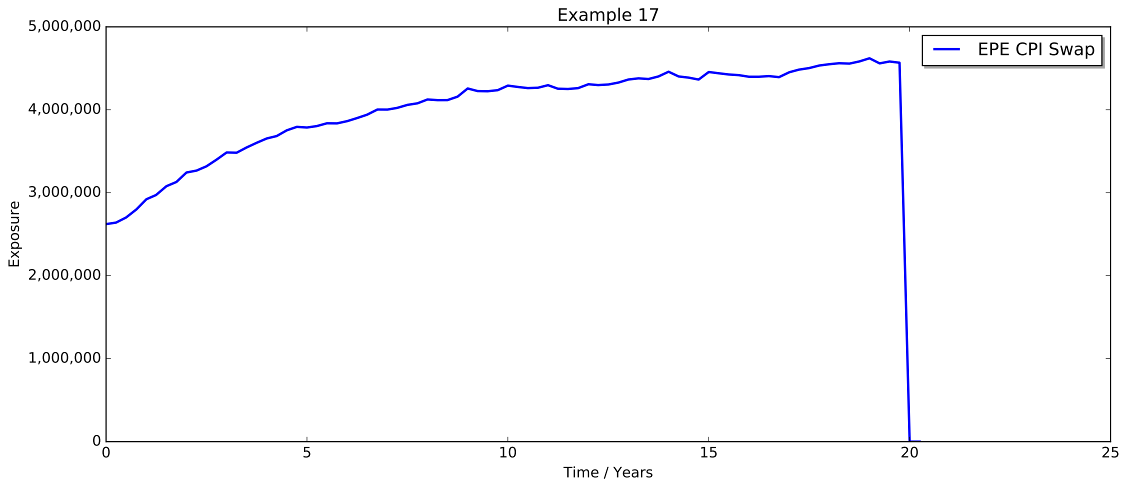


Figure 26: CPI Swap 1 exposure evolution. Simulation with 1000 paths and quarterly time steps.

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.

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

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

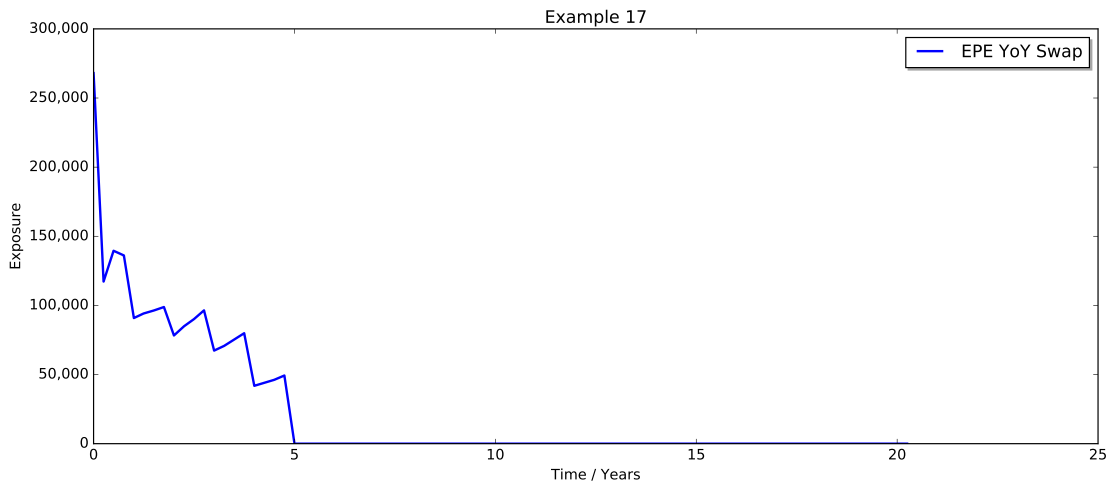


Figure 27: Year-on-Year Inflation Swap exposure evolution. Simulation with 1000 paths and quarterly time steps.

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

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 8.3.7.

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.14 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.14);
- 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`.

5.30 USD-Prime Curve Building via Prime-LIBOR Basis Swap

The example in folder `Examples/Example_30` demonstrates the implementation of the USD-Prime index in the ORE. The USD-Prime yield curve is built from USD-Prime vs USD 3M Libor basis swap quotes. The portfolio consists of two fair basis swaps (NPVs equal to 0):

- US Dollar Prime Rate vs 3 Month LIBOR
- US Dollar 3 Month LIBOR vs Fed Funds + 0.027

In particular, it is confirmed that the bootstrapped curves USD-FedFunds and USD-Prime follow the 3% rule observed on the market: `U.S. Prime Rate = (The Fed Funds Target Rate + 3%)`. (See http://www.fedprimerate.com/.)

Running ORE in directory `Examples/Example_30` with `python run.py` yields the USD-Prime curve in `Examples/Example_30/Output/curves.csv`.

5.31 Exposure Simulation using a Close-Out Grid

In the previous examples we have used a "lagged" approach, described at the end of appendix A.12, to take the Margin Period of Risk into account in exposure modelling. This has the disadvantage in ORE that we need to use equally-spaced time grids with time steps that match the MPoR, e.g. 2W, out to final portfolio maturity.

In this example we demonstrate an alternative approach supported by ORE since release 6. In this approach we use two nested grids: The (almost) arbitrary main simulation grid is used to compute "default values" which feed into the collateral balance $C(t)$ filtered by MTA and Threshold etc; an auxiliary "close-out" grid, offset from the main grid by the MPoR, is used to compute the delayed close-out values $V(t)$ associated with time default time t. The difference between $V(t)$ and $C(t)$ causes a residual exposure $[V(t) - C(t)]^+$ even if minimum transfer amounts and thresholds are zero.

The close-out date value can be computed in two ways in ORE

- as of default date, by just evolving the market from default date to close-out date ("sticky date"), or
- as of close-out date, by evolving both valuation date and market over the close-out period ("actual date"), i.e., the portfolio ages and cash flows might occur in the close-out period causing spikes in the evolution of exposures.

We are reusing one case from Example 10 here, perfect CSA with zero threshold and minimum transfer amount, so that the remaining exposure is solely due to the MPoR effect. The portfolio consists of a single at-the-money Swap in GBP. The relevant configuration changes that trigger this modelling are in the Parameters section of `simulation.xml` as shown in Listing 1

Listing 1: Close-out grid specification

```
<Parameters>
  <Discretization> ... </Discretization>
  <Grid> ... </Grid>
  <Calendar> ... </Calendar>
  <Sequence> ... </Sequence>
  <Scenario> ... </Scenario>
  <Seed> ... </Seed>
  <Samples> ... </Samples>
  <CloseOutLag> 2W </CloseOutLag>
  <MporMode> StickyDate </MporMode><!-- Alternative: ActualDate -->
</Parameters>
```

and moreover in the XVA analytics section of `ore_mpor.xml` as shown in Listing 2.

Listing 2: Close-out grid specification

```
<Analytic type="xva">
  ...
  <Parameter name="calculationType"> NoLag </Parameter>
  ...
</Parameters>
```

Run as usual calling `python run.py`.

5.32 Inflation Swap Exposure under Jarrow-Yildrim

The example here is similar to that in Section 5.17 in that we are generating exposures for inflation swaps. The example in Section 5.17 uses the Dodgson-Kainth model whereas this example uses the Jarrow-Yildrim model. The valuation date is 5 Oct 2020 and the portfolio contains four spot starting inflation swaps:

- trade_01: 20Y standard UKRPI ZCIIS struck at the fair market rate of 3.1925% giving an NPV of 0.0.
- trade_02: 20Y standard EUHICPXT ZCIIS struck at the fair market rate of 1.16875% giving an NPV of 0.0.

- trade_03: 20Y year on year EUHICPXT swap.
- trade_04: 20Y year on year UKRPI swap.

The example generates cash flows, NPVs, exposure evolutions and XVAs.

5.33 CDS Exposure Simulation

The example in folder `Examples/Example_33` is the credit variant of the example in 5.1. Running ORE in directory `Examples/Example_33` with

```
python run.py
```

yields the exposure evolution in

```
Examples/Example_33/Output/*.pdf
```

and shown in figure 28. Both CDS simulation and CDS Option pricing are run with

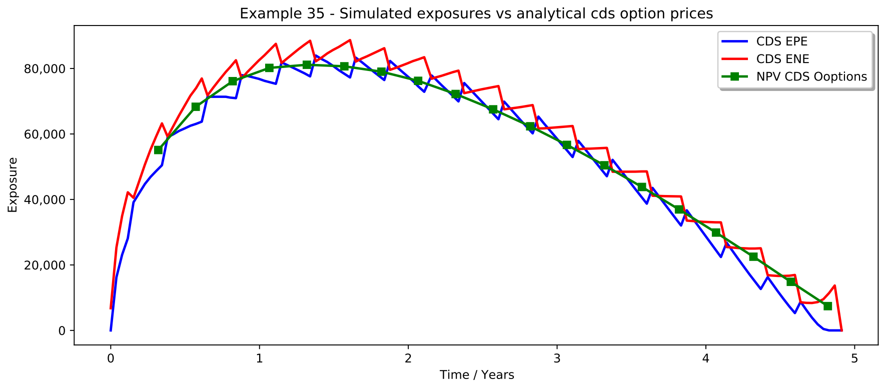


Figure 28: Credit Default Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are CDS Option prices. The simulation was run with bi-weekly time steps and 10,000 Monte Carlo samples to demonstrate the convergence of EPE and ENE profiles. A similar outcome can be obtained more quickly with 5,000 samples on a monthly time grid which is the default setting of Example_33.

calls to the ORE executable, essentially

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

which are wrapped into the script `Examples/Example_33/run.py` provided with the ORE release.

This example demonstrates credit simulation using the LGM model and the calculation of Wrong Way Risk due to credit correlation between the underlying entity of the CDS and the counterparty of the CDS trade via dynamic credit. Positive correlation between the two names weakens the protection of the CDS whilst negative correlation strengthens the protection.

The following table lists the XVA result from the example at different levels of correlation.

Correlation	NettingSetId	CVA	DVA	FBA	FCA
-100%	CPTY_B	-2,638	2,906	486	-1,057
-90%	CPTY_B	-2,204	2,906	488	-1,053
-50%	CPTY_B	-485	2,906	493	-1,040
-40%	CPTY_B	-60	2,906	495	-1,037
-30%	CPTY_B	363	2,906	496	-1,033
-20%	CPTY_B	784	2,906	498	-1,030
-10%	CPTY_B	1,204	2,906	500	-1,027
0%	CPTY_B	1,621	2,906	501	-1,023
10%	CPTY_B	2,036	2,906	503	-1,020
20%	CPTY_B	2,450	2,906	504	-1,017
30%	CPTY_B	2,861	2,906	506	-1,013
40%	CPTY_B	3,271	2,906	507	-1,010
50%	CPTY_B	3,679	2,906	509	-1,017
90%	CPTY_B	5,290	2,906	515	-994
100%	CPTY_B	5,689	2,906	517	-991

Table 5: CDS XVA results with LGM model

5.34 Wrong Way Risk

The example in folder `Examples/Example_34` is an extension of the example in 5.1 with dynamic credit and IR-CR correlation. As we are paying float, negative correlation implies that we pay more when the counterparty's credit worsens, leading to a surge of CVA.

The following table lists the XVA result from the example at different levels of correlation.

Correlation	NettingSetId	CVA	DVA	FBA	FCA
-30%	CPTY_A	105,146	68,061	31,519	-4,127
-20%	CPTY_A	88,442	68,061	30,976	-4,219
-10%	CPTY_A	71,059	68,061	30,439	-4,314
0%	CPTY_A	52,983	68,061	29,909	-4,411
10%	CPTY_A	34,199	68,061	29,386	-4,511
20%	CPTY_A	14,691	68,061	28,869	-4,614
30%	CPTY_A	-5,554	68,061	28,360	-4,719

Table 6: IR Swap XVA results with LGM model

5.35 Flip View

The example in folder `Examples/Example_35` demonstrates how ORE can be used to quickly switch perspectives in XVA calculations with minimal changes in the `ore.xml` file only. In particular it does not involve manipulating the portfolio input.

5.36 Choice of Measure

The example in folder `Examples/Example_36` illustrates the effect of measure changes on simulated expected and peak exposures. For that purpose we reuse Example 1 (un-collateralized vanilla swap exposure) and run the simulation three times with different risk-neutral measures,

- in the LGM measure as in Example 1 (note `<Measure>LGM</Measure>` in `simulation_lgm.xml`, this is the default also if the Measure tag is omitted)
- in the more common Bank Account measure (note `<Measure>BA</Measure>` in `simulation_ba.xml`)

- in the T-Forward measure with horizon T=20 at the Swap maturity (note `<Measure>LGM</Measure>` and `<ShiftHorizon>20.0</ShiftHorizon>` in `simulation_fwd.xml`)

The results are summarized in the exposure evolution graphs in figure 29. As expected, the expected exposures evolutions match across measures, as these are expected discounted NPVs and hence measure independent. However, peak exposures are dependent on the measure choice as confirmed graphically here. Many more measures are accessible with ORE, by way of varying the T-Forward horizon which was chosen arbitrarily here to match the Swap's maturity.

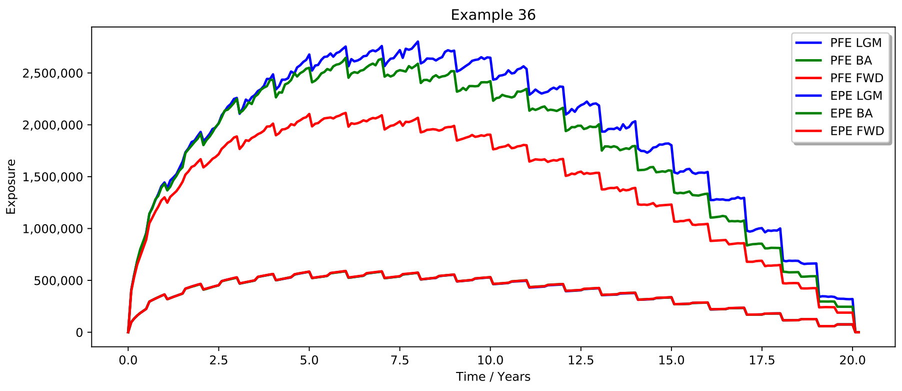


Figure 29: Evolution of expected exposures (EPE) and peak exposures (PFE at the 95% quantile) in three measures, LGM, Bank Account, T-Forward with T=20, with 10k Monte Carlo samples.

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 ipython 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).

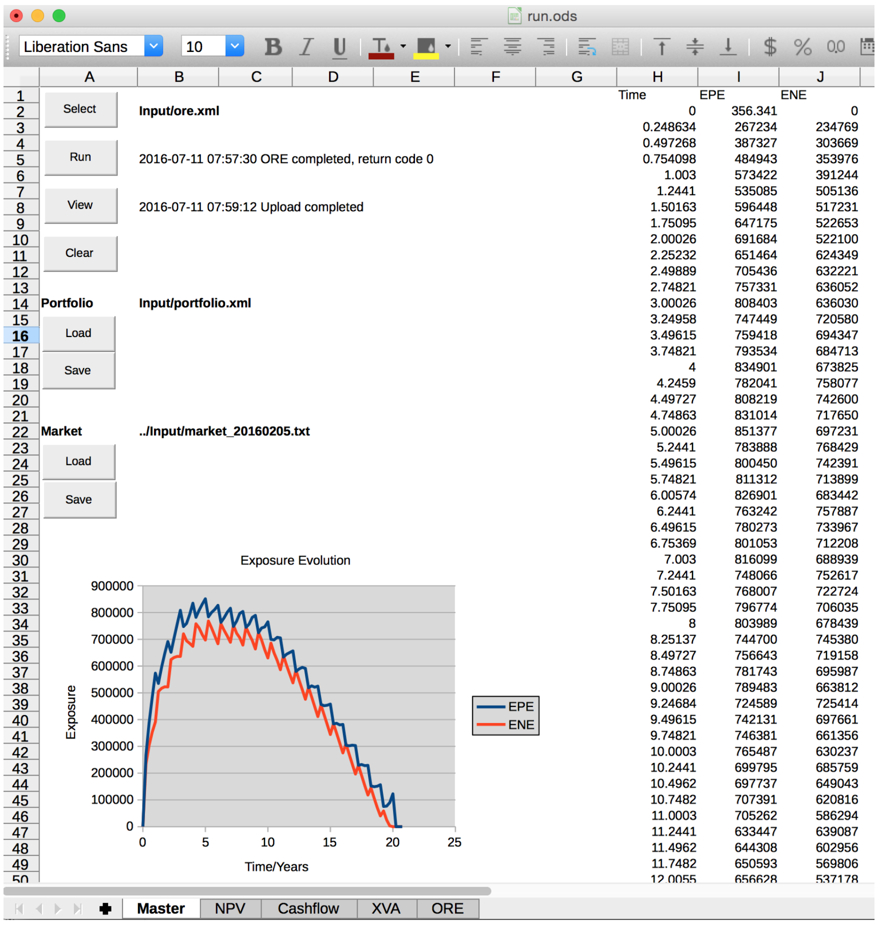
6.2 Calc

ORE comes with a simple LibreOffice Calc [11] 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 30. 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.



Input/ore.xml

2016-07-11 07:57:30 ORE completed, return code 0

2016-07-11 07:59:12 Upload completed

Portfolio **Input/portfolio.xml**

Market **../Input/market_20160205.txt**

Time	EPE	ENE
0	356.341	0
0.248634	267234	234769
0.497268	387327	303669
0.754098	484943	353976
1.003	573422	391244
1.2441	535085	505136
1.50163	596448	517231
1.75095	647175	522653
2.00026	691684	522100
2.25232	651464	624349
2.49889	705436	632221
2.74821	757331	636052
3.00026	808403	636030
3.24958	747449	720580
3.49615	759418	694347
3.74821	793534	684713
4	834901	673825
4.2459	782041	758077
4.49727	808219	742600
4.74863	831014	717650
5.00026	851377	697231
5.2441	783888	768429
5.49615	800450	742391
5.74821	811312	713899
6.00574	826901	683442
6.2441	763242	757887
6.49615	780273	733967
6.75369	801053	712208
7.003	816099	688939
7.2441	748066	752617
7.50163	768007	722724
7.75095	796774	706035
8	803989	678439
8.25137	744700	745380
8.49727	756643	719158
8.74863	781743	695987
9.00026	789483	663812
9.24684	724589	725414
9.49615	742131	697661
9.74821	746381	661356
10.0003	765487	630237
10.2441	699795	685759
10.4962	697737	649043
10.7482	707391	620816
11.0003	705262	586294
11.2441	633447	639087
11.4962	644308	602956
11.7482	650593	569806
12.0055	656628	537178

Figure 30: 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 Parameterisation

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 3.

Listing 3: ORE setup example

```
<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="dividendDataFile">../../Input/dividends_20160205.txt</Parameter> <!-- Optional --
  <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>
  <Parameter name="calendarAdjustment">../../Input/calendaradjustment.xml</Parameter>
  <Parameter name="currencyConfiguration">../../Input/currencies.xml</Parameter>
  <Parameter name="referenceDataFile">../../Input/referencedata.xml</Parameter>
  <Parameter name="iborFallbackConfig">../../Input/iborFallbackConfig.xml</Parameter>
  <!-- None, Unregister, Defer or Disable -->
  <Parameter name="observationModel">Disable</Parameter>
  <Parameter name="lazyMarketBuilding">false</Parameter>
  <Parameter name="continueOnError">false</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, fixings and dividends, 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 would not be loaded but implied, relevant when pricing/bootstrapping off hypothetical market data as e.g. in scenario analysis and stress testing. The curveConfigFile `curveconfig.xml`, the conventionsFile `conventions.xml`, the referenceDataFile `referencedata.xml`, the iborFallbackConfig, the marketDataFile and the fixingDataFile are explained in the sections below.

Parameter `calendarAdjustment` includes the `calendarAdjustment.xml` which lists out additional holidays and business days to be added to specified calendars.

The optional parameter `currencyConfiguration` points to a configuration file that contains additional currencies to be added to ORE's setup, see `Examples/Input/currencies.xml` for a full list of ISO currencies and a few unofficial currency codes that can thus be made available in ORE. Note that the external configuration does not override any currencies that are hard-coded in the QuantLib/QuantExt libraries, only currencies not present already are added from the external configuration file.

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.

If the parameter `lazyMarketBuilding` is set to true, the build of the curves in the TodaysMarket is delayed until they are actually requested. This can speed up the processing when some curves configured in TodaysMarket are not used. If not given, the parameter defaults to `true`.

If the parameter `continueOnError` is set to true, the application will not exit on an error, but try to continue the processing. If not given, the parameter defaults to `false`.

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

7.1.2 Markets

The `Markets` section (see listing 4) 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 4: 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
- additionalResults
- todaysMarketCalibration
- 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 5 shows the parametrisation of the first four basic analytics in the list above.

Listing 5: ORE analytics: npv, cashflow, curves, additional results, todays market calibratino

```
<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="additionalResults">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">additional\_results.csv</Parameter>
  </Analytic>
  <Analytic type="todaysMarketCalibration">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">todaysmarketcalibration.csv</Parameter>
  </Analytic>
  <Analytic type="...">
    <!-- ... -->
  </Analytic>
</Analytics>
```

The cashflow analytic writes a report containing all future cashflows of the portfolio. Table 7 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 7: 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 additionalResults analytic writes a report containing any additional results generated for the portfolio. The results are pricingengine specific but Table 8 shows the output for a vanilla swaption.

#TradeId	ResultId	ResultType	ResultValue
example_swaption	annuity	double	2123720984
example_swaption	atmForward	double	0.01664135
example_swaption	spreadCorrection	double	0
example_swaption	stdDev	double	0.00546015
example_swaption	strike	double	0.024
example_swaption	swapLength	double	4
example_swaption	vega	double	309237709.5
...			

Table 8: AddiitonalResults Report

The todaysMarketCalibration analytic writes a report containing information on the build of the t0 market.

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 6: 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="storeSurvivalProbabilities">Y</Parameter>
    <Parameter name="cubeFile">cube_A.dat</Parameter>
    <Parameter name="nettingSetCubeFile">nettingSetCube_A.dat</Parameter>
    <Parameter name="cptyCubeFile">cptyCube_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. 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. And finally, the key ‘store survival probabilities’ (Y or N) controls whether survival probabilities on simulation dates are stored in the cube for post processing in the context of Dynamic Credit XVA calculation. 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 7: 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="dynamicCredit">N</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, choices are:
  - *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
  - *NoLag* - used to disable any delayed settlement of the margin; this option is applied in combination with a "close-out" grid, see section 7.4.
- `allocationMethod`: XVA allocation method, choices are *None, Marginal, RelativeXVA, RelativeFairValueGross, RelativeFairValueNet*
- `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
- `dynamicCredit`: Flag to enable using pathwise survival probabilies when calculating CVA, DVA, FVA and MVA increments from exposures. If set to N the survival probabilities are extracted from T0 curves.

- `kva`: Flag to enable setting the kva ccr parameters.
- `kvaCapitalDiscountRate, kvaAlpha, kvaRegAdjustment, kvaCapitalHurdle, kvaOurPdFloor, kvaTheirPdFloor kvaOurCvaRiskWeight, kvaTheirCvaRiskWeight`: the kva CCR parameters (see A.10 and A.11.
- `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 8 shows a typical configuration for sensitvity calculation.

Listing 8: ORE analytic: sensitivity

```
<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 9 shows an example.

Listing 9: 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 10 shows a configuration example.

Listing 10: 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.15
- `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 volatility 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 11: 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 7.8 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 12: 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 13: 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 14: 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 15: 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 16: *FX volatility block with ID 'default'*

```
<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 17: *Swaption volatility block with ID 'default'*

```
<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 18: *Cap/Floor volatility block with ID 'default'*

```
<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 19: 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 20: 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 21: 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 22: 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 23: 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 24: 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 Volatility Surfaces

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

Listing 25: Inflation Cap/Floor Volatility Surfaces block with ID 'default'

```
<YYInflationCapFloorVolatilities id="default">
  <YYInflationCapFloorVolatitlity name="EUHICPXT">
      InflationCapFloorVolatility/EUHICPXT/EUHICPXT_YY_CF
  </InflationCapFloorVolatility>
</YYInflationCapFloorVolatilities>

<ZeroInflationCapFloorVolatilities id="default">
  <ZeroInflationCapFloorVolatility name="UKRPI">
      InflationCapFloorVolatility/UKRPI/UKRPI_ZC_CF
  </ZeroInflationCapFloorVolatility>
  <ZeroInflationCapFloorVolatility name="EUHICPXT">
      InflationCapFloorVolatility/EUHICPXT/EUHICPXT_ZC_CF
  </ZeroInflationCapFloorVolatility>
  <ZeroInflationCapFloorVolatility name="USCPI">
      InflationCapFloorVolatility/USCPI/USCPI_ZC_CF
  </ZeroInflationCapFloorVolatility>
</ZeroInflationCapFloorVolatilities>
```

This instructs ORE to build a set of year-on-year and zero inflation cap floor volatility 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.

7.2.15 CDS Volatility Structures

CDS volatility structures are configured as follows

Listing 26: CDS volatility structure block with ID 'default'

```
<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 27: Base Correlations block with ID 'default'

```
<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 28: Correlations block with ID 'default'

```
<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 29: Market configurations

```
<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.

```
<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="FxOptionAmerican">
  <Model>GarmanKohlhagen</Model>
  <ModelParameters/>
  <Engine>FdBlackScholesVanillaEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Douglas</Parameter>
    <Parameter name="TimeGridPerYear">100</Parameter>
    <Parameter name="XGrid">100</Parameter>
    <Parameter name="DampingSteps">0</Parameter>
    <!-- optional, prevents too small time grids for increased
         Greek precision when expiry is near, set to 1 if omitted -->
    <Parameter name="TimeGridMinimumSize">1</Parameter>
  </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>
    <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/>
  <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>
  <GlobalParameters>
    <Parameter name="ContinueOnCalibrationError">true</Parameter>
    <!-- typcially not present in a user configuration, but used internally -->
    <Parameter name="Calibrate">true</Parameter>
    <Parameter name="GenerateAdditionalResults"</Parameter>
  </GlobalParameters>
```

Listing 30: 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.

In addition to product specific settings there is also a block with global parameters with the following meaning:

- ContinueOnCalibrationError: If set to true an exceedence of a prescribed model calibration tolerance (for e.g. the LGM model) will not cause the trade building to fail, instead a warning is logged and the trade is processed anyway. Optional, defaults to false.
- Calibrate: If false, model calibration is disabled. This flag is usually not present in a user configuration, but only used internally for certain workflows within ORE which do not require a model calibration. Optional, defaults to true.
- GenerateAdditionalResults: If false, the generation of additional results within pricing engines will be suppressed (for those pricing engines which support this). This flag is usually not present in a user configuration, but only used internally to improve the performance for processes which only rely on the NPV as a result from pricing engines, e.g. when repricing trades under sensitivity or stress scenarios. Option, defaults to false.

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 31: 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 32: Simulation configuration

```
<Parameters>
  <Discretization>Exact</Discretization>
  <Grid>80,3M</Grid>
  <Calendar>EUR,USD,GBP,CHF</Calendar>
  <DayCounter>ACT/ACT</DayCounter>
  <Sequence>SobolBrownianBridge</Sequence>
  <Seed>42</Seed>
  <Samples>1000</Samples>
  <Ordering>Steps</Ordering>
  <DirectionIntegers>JoeKuoD7</DirectionIntegers>
  <!-- The following two nodes are optional -->
  <CloseOutLag>2W</CloseOutLag>
  <MporMode>StickyDate</MporMode>
</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.
- `DayCounter:` Day count convention used to translate dates to times. Optional, defaults to ActualActual ISDA.
- `Sequence:` Choose random sequence generator (*MersenneTwister, MersenneTwisterAntithetic, Sobol, SobolBrownianBridge*).
- `Seed:` Random number generator seed
- `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*)
- `CloseOutLag`: If this tag is present, this specifies the close-out period length (e.g. 2W) used; otherwise no close-out grid is built. The close-out grid is an auxiliary time grid that is offset from the main default date grid by the close-out period, typically set to the applicable margin period of risk. If present, it is used to

[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`.

evolve the portfolio value and determine close-out values associated with the preceding default date valuation.

- `MporMode`: This tag is expected if the previous one is present, permissible values are then `StickyDate` and `ActualDate`. `StickyDate` means that only market data is evolved from the default date to close-out date for close-out date valuation, the valuation as of date remains unchanged and trades do not "age" over the period. As a consequence, exposure evolutions will not show spikes caused by cash flows within the close-out period. `ActualDate` means that trades will also age over the close-out period so that one can experience exposure evolution spikes due to cash flows.

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 33: 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>
  <Measure>LGM</Measure><!-- Choices: LGM, BA -->
  <!-- ... -->
</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.

The Measure tag allows switching between the LGM and the Bank Account (BA) measure for the risk-neutral market simulations using the Cross Asset Model. Note that within LGM one can shift the horizon (see ParameterTransformation below) to effectively switch to a T-Forward measure.

Each interest rate model is specified by a block as follows

Listing 34: 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 *HullWhite* 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 [21] 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 35: 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 36: Simulation model equity configuration

```
<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)

For the inflation model component, there is a choice between a Dodgson Kainth model and a Jarrow Yildrim model. The Dodgson Kainth model is specified in a `LGM` or `DodgsonKainth` node as outlined in Listing 37. The inflation model parameterisation inherits from the LGM parameterisation for interest rate components, in particular the `CalibrationType`, `Volatility` and `Reversion` elements. The `CalibrationCapFloors` element specify the model's calibration to a selection of either CPI caps or CPI floors with specified strike.

Listing 37: Simulation model DK 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 calibration instruments may be specified in an alternative way via a `CalibrationBaskets` node. In general, a `CalibrationBaskets` node can contain multiple `CalibrationBasket` nodes each containing a list of calibration instruments of the same type. For Dodgson Kainth, only a single calibration basket is allowed and the instruments must be of type `CpiCapFloor`. So, for example, the `CalibrationCapFloors` node in Listing 37 could be replaced with the `CalibrationBaskets` node in 38.

Listing 38: Calibration basket for DK inflation model component

```
<CalibrationBaskets>
  <CalibrationBasket>
    <CpiCapFloor>
      <Type>Floor</Type>
      <Maturity>2Y</Maturity>
      <Strike>0.03</Strike>
    </CpiCapFloor>
    <CpiCapFloor>
      <Type>Floor</Type>
      <Maturity>4Y</Maturity>
      <Strike>0.03</Strike>
    </CpiCapFloor>
    <CpiCapFloor>
      <Type>Floor</Type>
      <Maturity>6Y</Maturity>
      <Strike>0.03</Strike>
    </CpiCapFloor>
    <CpiCapFloor>
      <Type>Floor</Type>
      <Maturity>8Y</Maturity>
      <Strike>0.03</Strike>
    </CpiCapFloor>
    <CpiCapFloor>
      <Type>Floor</Type>
      <Maturity>10Y</Maturity>
      <Strike>0.03</Strike>
    </CpiCapFloor>
  </CalibrationBasket>
</CalibrationBaskets>
```

The Jarrow Yildrim model is specified in a `JarrowYildirim` node as outlined in Listing 39. The `RealRate` node describes the JY real rate process and has `Volatility` and `Reversion` nodes that follow those outlined in the interest rate LGM section above. The `Index` node describes the JY index process and has a `Volatility` component that follows the `Sigma` component of the FX model above. The `CalibrationBaskets` node is as outlined above for Dodgson Kainth but up to two baskets may be used and extra inflation instruments are supported in the calibration. More information is provided below.

The `CalibrationType` determines the calibration approach, if any, that is used to calibrate the various parameters of the model i.e. the real rate reversion, the real rate volatility and the index volatility. If the `CalibrationType` is `None`, no calibration is attempted and all parameter values must be explicitly specified. If the `CalibrationType` is `BestFit`, the parameters that have `Calibrate` set to `Y` will be calibrated to the instruments specified in the `CalibrationBaskets` node. If the `CalibrationType` is `Bootstrap`, there are a number of options:

1. The index volatility parameter may be calibrated, indicated by setting `Calibrate` to `Y` for that parameter, with both of the real rate parameters not calibrated and set explicitly in the `RealRate` node. There should be exactly one `CalibrationBasket` in the `CalibrationBaskets` node and its `parameter` attribute may be set to `Index` or omitted.

2. One of the real rate parameters may be calibrated, indicated by setting `Calibrate` to `Y` for that parameter, with the index volatility not calibrated and set explicitly in the `Volatility` node. There should be exactly one `CalibrationBasket` in the `CalibrationBaskets` node and its `parameter` attribute may be set to `RealRate` or omitted.

3. One of the real rate parameters and the index volatility parameter may be calibrated together. There should be exactly two `CalibrationBasket` nodes in the `CalibrationBaskets` node. The `parameter` attribute should be set to `RealRate` on the `CalibrationBasket` node that should be used for the real rate parameter calibration. Similarly, the `parameter` attribute should be set to `Index` on the `CalibrationBasket` node that should be used for the index volatility parameter calibration. The parameters are calibrated iteratively in turn until the root mean squared error over all calibration instruments in the two baskets is below the tolerance specified by the `RmseTolerance` in the `CalibrationConfiguration` node or until the maximum number of iterations as specified by the `MaxIterations` in the `CalibrationConfiguration` node has been reached. The `CalibrationConfiguration` node is optional. If it is omitted, the `RmseTolerance` defaults to 0.0001 and the `MaxIterations` defaults to 50.

Note that it is an error to attempt to calibrate both of the real rate parameters together when `CalibrationType` is `Bootstrap`. If a parameter is being calibrated with `CalibrationType` set to `Bootstrap`, the `ParamType` should be `Piecewise`. The `TimeGrid` will be overridden for that parameter by the relevant calibration instrument times and the parameter's initial values are set to the first element of the `InitialValue` list. So, leaving the `TimeGrid` node empty and giving a single value in the `InitialValue` node is the clearest XML setup in this case.

Listing 39: Simulation model JY inflation component configuration

```
<JarrowYildirim index="EUHICPXT">
  <Currency>EUR</Currency>
  <CalibrationType>Bootstrap</CalibrationType>
  <RealRate>
    <Volatility>
      <Calibrate>Y</Calibrate>
      <VolatilityType>Hagan</VolatilityType>
      <ParamType>Piecewise</ParamType>
      <TimeGrid/>
      <InitialValue>0.0001</InitialValue>
    </Volatility>
    <Reversion>
      <Calibrate>N</Calibrate>
      <ReversionType>HullWhite</ReversionType>
      <ParamType>Constant</ParamType>
      <TimeGrid/>
      <InitialValue>0.5</InitialValue>
    </Reversion>
    <ParameterTransformation>
      <ShiftHorizon>0.0</ShiftHorizon>
      <Scaling>1.0</Scaling>
    </ParameterTransformation>
  </RealRate>
  <Index>
    <Volatility>
      <Calibrate>Y</Calibrate>
      <ParamType>Piecewise</ParamType>
      <TimeGrid/>
      <InitialValue>0.0001</InitialValue>
    </Volatility>
  </Index>
  <CalibrationBaskets>
    <CalibrationBasket parameter="Index">
      <CpiCapFloor>
        <Type>Floor</Type>
        <Maturity>2Y</Maturity>
        <Strike>0.0</Strike>
      </CpiCapFloor>
      ...
    </CalibrationBasket>
    <CalibrationBasket parameter="RealRate">
      <YoYSwap>
        <Tenor>2Y</Tenor>
      </YoYSwap>
      ...
    </CalibrationBasket>
  </CalibrationBaskets>
  <CalibrationConfiguration>
    <RmseTolerance>0.00000001</RmseTolerance>
    <MaxIterations>40</MaxIterations>
  </CalibrationConfiguration>
</JarrowYildirim>
```

The `CpiCapFloor` and `YoYSwap` calibration instruments can be seen in Listing 39. A `YoYCapFloor` is also allowed and it has the structure shown in Listing 40. The `Type` may be Cap or Floor. The Tenor should be a maturity period e.g. 5Y. The Strike should be an absolute strike level for the year on year cap or floor e.g. 0.01 for 1%.

Listing 40: Layout for YoYCapFloor *calibration instrument.*

```
<YoYCapFloor>
  <Type>...</Type>
  <Tenor>...</Tenor>
  <Strike>...</Strike>
</YoYCapFloor>
```

Finally, the instantaneous correlation structure is specified as follows.

Listing 41: 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>`.

For Yield Curves, there are several choices for the interpolation and extrapolation:

- Interpolation: This can be LogLinear or LinearZero. If not given, the value defaults to LogLinear.
- Extrpolation: This can be FlatFwd or FlatZero. If not given, the value defaults to FlatFwd.

```
<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>FlatFwd</Extrapolation>
      <DayCounter>ACT/ACT</DayCounter> <!-- Sets a new default for all yieldCurves -->
    </Configuration>
  </YieldCurves>
  <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>
      <TimeExtrapolation>Flat</TimeExtrapolation>
      <StrikeExtrapolation>Flat</StrikeExtrapolation>

  </EquityVolatilities>
  ...
  <BenchmarkCurves>
    <BenchmarkCurve>
      <Currency>EUR</Currency>
      <Name>BENCHMARK_EUR</Name>
  </BenchmarkCurve>
  ...
  </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 42: 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 43, 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

Additional flags:

- ComputeGamma: If set to false, second order sensitivity computation is suppressed
- UseSpreadedTermStructures: If set to true, spreaded termstructures over t0 will be used for sensitivity calculation (where supported), to improve the alignment of the scenario sim market and t0 curves

```
<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>
  ...
  <ComputeGamma>true</ComputeGamma>
  <UseSpreadedTermStructures>false</UseSpreadedTermStructures>
</SensitivityAnalysis>
```

Listing 43: 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 44.

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 44: Stress configuration

7.7 Calendar Adjustment: `calendaradjustment.xml`

This file `calendaradjustment.xml` list out all additional holidays and business days that are added to a specified calendar in ORE. These dates would originally be missing from the calendar and has to be added.The general structure is shown in listing 45. In this example, two additional dates had been added to the calendar "Japan", one addtional holiday and one additional business day. If the user is not certain wether the date is already included or not, adding it to the `calendaradjustment.xml` to be safe won't raise any errors. A sample `calendaradjustment.xml` file can be found in the global example input directory. However, it is only used in Example_1.

```
<CalendarAdjustments>
  <Calendar name="Japan">
    <AdditionalHolidays>
      <Date>2020-01-01</Date>
    </AdditionalHolidays>
    <AdditionalBusinessDays>
      <Date>2020-01-02</Date>
    </AdditionalBusinessDays>
</CalendarAdjustments>
```

Listing 45: Calendar Adjustment

If the parameter `BaseCalendar` is provided then a new calendar will be created using the specified calendar as a base, and adding any `AdditionalHolidays` or `AdditionalBusinessDays`. In the example below a new calendar `CUSTOM\_Japan` is being created, it will include any additional holidays or business days specified in the original `Japan` calendar plus one additional date.

If a new calendar is added in this way and the schema is being used to validate XML input, the corresponding calendar name must be prefixed with 'CUSTOM_'.

```
<CalendarAdjustments>
  <Calendar name="CUSTOM\_Japan">
    <BaseCalendar>Japan</BaseCalendar>
    <AdditionalHolidays>
      <Date>2020-04-06</Date>
    </AdditionalHolidays>
</CalendarAdjustments>
```

Listing 46: Calendar Adjustment creating a new calendar

7.8 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.

7.8.1 Yield Curves

The top level XML elements for each `YieldCurve` node are shown in Listing 47.

Listing 47: 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>
  <BootstrapConfig>
    ...
  </BootstrapConfig>
</YieldCurve>
```

The meaning of each of the top level elements in Listing 47 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 9. 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 10. 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}. It is preferable to use the `Accuracy` node in the `BootstrapConfig` node below for specifying this value. However, if this node is explicitly supplied, it takes precedence for backwards compatibility purposes.

- 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*.
- `BootstrapConfig` [Optional]: this node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's default values described in section 7.8.19 are used.

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

Table 9: 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)
Quadratic	Quadratic interpolation
LogQuadratic	Quadratic interpolation on the natural log of the interpolation variable
Hermite	Hermite cubic spline interpolation
CubicSpline	Non-monotonic cubic spline interpolation with second derivative at left and right
ExponentialSplines	Exponential Spline curve fitting, for Fitted Bond Curves only
NelsonSiegel	Nelson-Siegel curve fitting, for Fitted Bond Curves only
Svensson	Svensson curve fitting, for Fitted Bond Curves only

Table 10: 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. Note that for all segment types below, with the exception of `DiscountRatio` and `AverageOIS`, the `Quote` elements within the `Quotes` node may have an `optional` attribute indicating whether or not the quote is optional. Example:

```
<Quotes>
  <Quote optional="true"></Quote>
</Quotes>
```

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 48. 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 7.11. The `Conventions` node associates conventions with the quotes.

Listing 48: 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*, *Swap* or *BMA Basis Swap* and the node has the structure shown in Listing 49. 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 49: 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 50. This segment is used to hold quotes for Average OIS swap instruments. The `Quotes` node has the structure shown in Listing 51. 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 50: Average OIS yield curve segment

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

Listing 51: Average OIS segment's quotes section

```
<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 52. 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 49 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 52: Tenor basis yield curve segment

```
<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*, *Cross Currency Basis Swap* or *Cross Currency Fix Float Swap*. When the `Type` node has the value *FX Forward*, the node has the structure shown in Listing 53. 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 47. 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 53: 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 54. 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 47. 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 47. 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 54: 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 55. This segment is used to build yield curves which are expressed as a spread over some reference yield curve.

Listing 55: Zero spread yield curve segment

```
    <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>
```

Fitted Bond Segment

When the node name is `FittedBond`, the `Type` node has the only allowable value *FittedBond*, and the node has the structure shown in Listing 56. This segment is used to build yield curves which are fitted to liquid bond prices. The segment has the following elements:

- Quotes: a list of bond price quotes, for each security in the list, reference data must be available
- IborIndexCurves: for each Ibor index that is required by one of the bonds to which the curve is fitted, a mapping to an estimation curve for that index must be provided
- ExtrapolateFlat: if true, the parametric curve is extrapolated flat in the instantaneous forward rate before the first and after the last maturity of the bonds in the calibration basket. This avoids unrealistic rates at the short end or for long maturities in the resulting curve.

The `BootstrapConfig` has the following interpretation for a fitted bond curve:

- Accuracy [Optional, defaults to 1E-12]: the desired accuracy expressed as a weighted rmse in the implied quote, where 0.01 = 1 bp. Once this accuracy is reached in a calibration trial, the fit is accepted, no further calibration trials re run. In general, this parameter should be set to a higher than the default value for fitted bond curves.
- GlobalAccuracy [Optional]: the acceptable accuracy. If the Accuracy is not reached in any calibration trial, but the GloablAccuracy is met, the best fit among the calibration trials is selected as a result of the calibration. If not given, the best calibration trial is compared to the Accuracy parameter instead.
- DontThrow [Optional, defaults to false]: If true, the best calibration is always accepted as a result, i.e. no error is thrown even if the GlobalAccuracy is breached.
- MaxAttempts [Optional, defaults to 5]: The maximum number of calibration trials. Each calibration trial is run with a random calibratio seed. Random calibration seeds are currently only supported for the NelsonSiegel interpolation method.

Listing 56: Fitted bond yield curve segment

```
<YieldCurve>
  ...
  <Segments>
    <FittedBond>
      <Type>FittedBond</Type>
      <Quotes>
        <Quote>BOND/PRICE/SECURITY_1</Quote>
        <Quote>BOND/PRICE/SECURITY_2</Quote>
        <Quote>BOND/PRICE/SECURITY_3</Quote>
        <Quote>BOND/PRICE/SECURITY_4</Quote>
        <Quote>BOND/PRICE/SECURITY_5</Quote>
      </Quotes>
      <!-- mapping of Ibor curves used in the bonds from which the curve is built -->
      <IborIndexCurves>
        <IborIndexCurve iborIndex="EUR-EURIBOR-6M">EUR-EURIBOR-6M</IborIndexCurve>
      </IborIndexCurves>
      <!-- flat extrapolation before first and after last bond maturity -->
      <ExtrapolateFlat>true</ExtrapolateFlat>
    </FittedBond>
  </Segments>
  <!-- NelsonSiegel, Svensson, ExponentialSplines -->
  <InterpolationMethod>NelsonSiegel</InterpolationMethod>
  <YieldCurveDayCounter>A365</YieldCurveDayCounter>
  <Extrapolation>true</Extrapolation>
  <BootstrapConfig>
    <!-- desired accuracy (in implied quote) -->
    <Accuracy>0.1</Accuracy>
    <!-- tolerable accuracy -->
    <GlobalAccuracy>0.5</GlobalAccuracy>
    <!-- do not throw even if tolerable accuracy is breached -->
    <DontThrow>false</DontThrow>
    <!-- max calibration trials to reach desired accuracy -->
    <MaxAttempts>20</MaxAttempts>
  </BootstrapConfig>
</YieldCurve>
```

Yield plus Default Segment

When the node name is `YieldPlusDefault`, the `Type` node has the only allowable value *Yield Plus Default*, and the node has the structure shown in Listing 57. This segment is used to build all-in discounting yield curves from a benchmark curve and (a weighted sum of) default curves. The construction is in some sense inverse to the benchmark default curve construction, see 7.8.3.

- ReferenceCurve: the benchmark yield curve serving as the basis of the resulting yield curve
- DefaultCurves: a list of default curves whose weighted sum is added to the benchmark yield curve
- Weights: a list of weights for the default curves, the number of weights must match the number of default curves

Notice that it is explicitly allowed to use default curves in different currencies than the benchmark yield curve. In the construction, the hazard rate is reinterpreted as an instantaneous forward rate, and the sum of the curves is being built in the instantaneous forward rate.

The definition takes into account the recovery rates associated to each default curve. The resulting discount factor is computed as

$$P(0,t) = \prod_i S_i(t)^{(1-R)w_i} \tag{1}$$

where S_i and R_i are the survival probabilities and recovery rates of the source default curves, and w_i are the weights.

Listing 57: Yield plus default curve segment

```
  <YieldCurve>
    <CurveId>BenchmarkPlusDefault</CurveId>
    <CurveDescription>USD Libor 3M + 0.5 x CDX.NA.HY + 0.5 x EUR.10BP</CurveDescription>
    <Currency>USD</Currency>
    <DiscountCurve/>
    <Segments>
      <YieldPlusDefault>
        <Type>Yield Plus Default</Type>
        <ReferenceCurve>USD3M</ReferenceCurve>
        <DefaultCurves>
          <DefaultCurve>Default/USD/CDX.NA.HY</DefaultCurve>
          <DefaultCurve>Default/EUR/EUR.10BP</DefaultCurve>
        </DefaultCurves>
        <Weights>
          <Weight>0.5</Weight>
          <Weight>0.5</Weight>
        </Weights>
      </YieldPlusDefault>
    </Segments>
  </YieldCurve>
</YieldCurves>
```

Weighted Average Segment

When the node name is `WeightedAverage`, the `Type` node has the only allowable value *Weighted Average*, and the node has the structure shown in Listing 58. This segment is used to build a curve with instantaneous forward rates that are the weighted sum of instantaneous forward rates of reference curves. This way a projection curve for non-standard Ibor curves can be build, e.g. to project a Euribor2M index using the curves for 1M and 3M.

- ReferenceCurve1: the first source curve
- ReferenceCurve2: the second source curve
- Weight1: the weight of the first curve
- Weights: the weight of the second curve

If $P_1(0,t)$ and $P_2(0,t)$ denote the discount factors of the two reference curves, the discount factor $P(0,t)$ of the resulting curve is defined as

$$P(0,t) = P_1(0,t)^{w_1} P_2(0,t)^{w_2} \tag{2}$$

Listing 58: Weighted Average yield curve segment

```
<YieldCurve>
  <CurveId>EUR2M</CurveId>
  <CurveDescription>Euribor2M forwarding curve, interpolated from 1M and 3M</CurveDescription>
  <Currency>EUR</Currency>
  <DiscountCurve>EUR1D</DiscountCurve>
  <Segments>
    <WeightedAverage>
      <Type>Weighted Average</Type>
      <ReferenceCurve1>EUR1M</ReferenceCurve1>
      <ReferenceCurve2>EUR3M</ReferenceCurve2>
      <Weight1>0.5</Weight1>
      <Weight2>0.5</Weight2>
    </WeightedAverage>
  </Segments>
</YieldCurve>
```

Ibor Fallback Segment

When the node name is `IborFallback`, the `Type` node has the only allowable value *Ibor Fallback*, and the node has the structure shown in Listing 59. This segment is used to build a projection curve for an Ibor index based on a risk free rate and a spread.

Listing 59: Ibor fallback segment

```
<YieldCurve>
  <CurveId>USD-LIBOR-3M</CurveId>
  <CurveDescription>USD-Libor-3M built from USD-SOFR plus spread</CurveDescription>
  <Currency>USD</Currency>
  <DiscountCurve/>
  <Segments>
    <IborFallback>
      <Type>Ibor Fallback</Type>
      <IborIndex>USD-LIBOR-3M</IborIndex>
      <RfrCurve>Yield/USD/USD-SOFR</RfrCurve>
      <!-- optional, if not given the rfr index and sprad are read from the ibor
           fallback configuration -->
      <RfrIndex>USD-SOFR</RfrIndex>
      <Spread>0.0026161</Spread>
    </IborFallback>
  </Segments>
</YieldCurve>
```

7.8.2 Default Curves from CDS

Default curves can be bootstrapped from credit default swap (CDS) market instruments. The CDS market quotes may be given as a par spread or as an upfront price. These market quotes are documented in Sections 10.12 and 10.13 respectively. The bootstrap also requires a market recovery rate quote and this is documented in Section 10.14.

Listing 60 outlines the configuration required to build a default curve from CDS quotes. The meaning of each of the nodes is as follows:

- `CurveId`: Unique identifier for the bootstrapped default curve.
- `CurveDescription` [Optional]: A description of the default curve. It is for information only and may be left blank.
- `Currency`: The default curve's currency.
- `Type`: For a default curve built from CDS, the `Type` should be set to `SpreadCDS` if the `Quotes` reference CDS spread quotes or `Price` if the `Quotes` reference upfront price quotes.
- `DiscountCurve`: A reference to a valid discount curve specification that will be used to discount cashflows during the bootstrap process. It should be of the form `Yield/Currency/curve_name` where `curve_name` is the name of a yield curve defined in the yield curve configurations.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 17.
- `RecoveryRate`: A valid recovery rate quote name as documented in Section 10.14.
- `StartDate` [Optional]: The `StartDate` is optional and is used for index CDS to specify the start date of the index CDS. This is then used to determine the maturity associated with the index CDS spread quotes which are quoted with a tenor. For single name CDS, this should be omitted.
- `Quotes`: The `Quotes` element should be populated with a list of valid `Quote` elements. If the `Type` is `SpreadCDS`, the quotes should be CDS spread quote strings as documented in Section 10.12 and if `Type` is `Price`, the quotes should be CDS upfront price quote strings as documented in Section 10.13. The attribute `optional` in the `Quote` element should be set to `true` if the associated quote is optional and set to `false` if the associated quote is mandatory. If a quote is mandatory and not found in the market, the default curve building will fail. The attribute `optional` may be omitted from the quote element. In this case, it defaults to `false` and the quote is mandatory. Note also that instead of a list of explicit quotes, a single quote may be provided with the wildcard character `*`. In this case, the market is searched for quotes matching the pattern. For example, `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/XR14/*` would return all quotes in the market that start with `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/XR14`.
- `Conventions`: The name of a valid set of CDS conventions, as documented in Section 7.11.21, to use in the bootstrap.

- `Extrapolation` [Optional]: A boolean value indicating if the bootstrapped default curve allows for extrapolation past the last pillar date. Allowable boolean values are given in the Table 28. If omitted, it defaults to `true`.

- `ImplyDefaultFromMarket` [Optional]: A boolean value indicating if a reference entity's default should be implied from the market data. Allowable boolean values are given in the Table 28. If omitted, it defaults to `false`. When a default credit event has been determined for an entity, certain market data providers continue to supply a recovery rate from the credit event determination date up to the credit event auction settlement date. In this period, no CDS spreads or upfront prices are provided. When this flag is `true`, we assume an entity is in default and awaiting a credit event auction if we find a recovery rate in the market but no CDS spreads or upfront prices. In this case, we build a survival probability curve with a value of close to but greater than 0.0 for one day after the valuation date. This will give an approximation to the correct price for CDS and index CDS in these cases. When this flag is `false`, we make no such assumption and the default curve building will fail.

- `BootstrapConfig` [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's default values described in section 7.8.19 are used.

- AllowNegativeRates [Optional]: If set to false (default) negative instantaneous hazard rates implied by the CDS quotes lead to an exception or - if the DontThrow flag in the BootstrapConfig is set to true - to a zero instantaneous hazard rate in the relevant segment of the curve. In the latter case the market CDS instrument associated to the critical curve segment will not match the market quote exactly. If set to true, negative instantaneous hazard rates will be allowed during the bootstrap (in a range that is technically defined by the MaxFactor and MaxAttempts parameters for the survival probability in the bootstrap config).

```xml
<DefaultCurve>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>USD</Currency>
  <Type>...</Type>
  <DiscountCurve>...</DiscountCurve>
  <DayCounter>...</DayCounter>
  <RecoveryRate>...</RecoveryRate>
  <StartDate>...</StartDate>
  <Quotes>
    <Quote optional="true">...</Quote>
    ...
  </Quotes>
  <Conventions>...</Conventions>
  <Extrapolation>...</Extrapolation>
  <ImplyDefaultFromMarket>...</ImplyDefaultFromMarket>
  <BootstrapConfig>
    ...
  </BootstrapConfig>
  <AllowNegativeRates>...</AllowNegativeRates>
</DefaultCurve>
```

Listing 60: Default curve configuration based on CDS quotes

7.8.3 Benchmark Default Curve

Default curves can be set up as a difference curve of two yield curves as shown in listing 61. A typical use case is to back out a default curve from an all-in discounting curve fitted to a series of liquid bond prices (the “source curve”) and a benchmark curve representing a benchmark funding level. The default curve can then be used in models consuming a benchmark curve and a default curve.

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

$$S(t) = (P_S(0,t)/P_B(0,t))^{1/(1-R)} \tag{3}$$

on the given pillar times. Her, R is the specified recovery rate. If the recovery rate is zero, which is the usual case, the formula simplifies to

$$S(0,t) = P_S(0,t)/P_B(0,t) \tag{4}$$

The interpolation is backward flat in the hazard rate. The meaning of each node is as follows:

- CurveId: The curve id.
- CurveDescription: The curve description.
- Currency: The currency of the curve.
- Type: Must be set to Benchmark.
- DayCounter: The day counter used to convert dates to times.
- RecoveryRate [optional]: The recovery rate for the resulting default curve. Defaults to zero. The recovery rate can be a market quote as usual or also a fixed numeric value for this curve type.
- BenchmarkCurve: The benchmark yield curve, typically this is the standard Ibor curve in the currence (e.g. EUR-EURIBOR-6M, USD-Libor-3M, ...)
- SourceCurve: The all-in discounting curve.
- Pillars: The pillars on which to match the source curve
- SpotLag: The pillar dates are derived using the spot lag and the tenors as specified in the Pillars node using the specified calendar.
- Calendar: The calendar used to derive the pillar dates.
- Extrapolation [Optional]: If set to true, the curve is extrapoalted beyond the last pillar. Defaults to true.
- AllowNegativeRates [Optional]: If set to true, the check for non-negative instantaneous hazard rate in the result curve is disabled, i.e. the relation $P_S(0,t) \leq P_B(0,t)$ is not enforced. This flag should be enabled with care, i.e. a model consuming the resulting default curve must be able to handle negative

hazard rates appropriately. On the other hand in some situations it is natural that the source curve rates are below the benchmark rates. Defaults to false.

```
    <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>
      <RecoveryRate>RECOVERY_RATE/RATE//SNR/USD</RecoveryRate>
      <BenchmarkCurve>Yield/EUR/EUR6M</BenchmarkCurve>
      <SourceCurve>Yield/EUR/BOND_YIELD_EUR</SourceCurve>
      <Pillars>1Y,2Y,3Y,4Y,5Y,7Y,10Y</Pillars>
      <SpotLag>0</SpotLag>
      <Calendar>TARGET</Calendar>
      <Extrapolation>true</Extrapolation>
      <AllowNegativeRates>false</AllowNegativeRates>
    </DefaultCurve>
  </DefaultCurves>
```

Listing 61: Benchmark default curve

7.8.4 Multi-Section Default Curve

Default curves can be build by stitching together instantaneous hazard rates from multiple source curves for multiple date ranges as shown in listing 62.

The hazard rate of the resulting curve is taken from the ith input curve ($i = 0, 1, 2, \ldots$) for dates before the ith switch date and (if $i > 0$) on or after the $i-1$th switch date. The day counter of all input curves should be equal to the day counter of the result curve. The interpolation is hardcoded as backward flat in the hazard rate.

If not given, the recovery rate R is assumed to be zero. The result default curve's survival probabiltiies are computed as

$$S(t) = \left[\left(\frac{P_{S,n}(t)}{P_{S,n}(t_n)} \right)^{(1-R_n)} \Pi_{i=0}^{n-1} \left(\frac{P_{S,i}(t_{i+1})}{P_{S,i}(t_i)} \right)^{(1-R_i)} \right]^{\frac{1}{1-R}} \tag{5}$$

where $P_{S,i}$ is the survival probability of the ith source curve, R_i is the associated recovery rate for the ith source curve, n is chosen such that $P_{S,n}$ is the relevant source curve for time t according to the given switch dates and curve i is relevant for times in $[t_i, t_{i+1}]$.

The meaning of each node is as follows:

- CurveId: The curve id.
- CurveDescription: The curve description.
- Currency: The currency of the curve.
- Type: Must be set to MutliSection.
- SourceCurves: The list of input default curves.

- SwitchDates: The list of dates where we switch from one input curve to the next. The number of switch dates must be one less than the number of source curves.
- DayCounter: The day counter used to convert dates to times.
- RecoveryRate [optional]: The recovery rate for the resulting default curve. Defaults to zero. The recovery rate can be a market quote as usual or also a fixed numeric value for this curve type.
- Extrapolation [Optional]: If set to true, the curve is extrapoalted beyond the last pillar. Defaults to true.

```
<DefaultCurve>
   <CurveId>MyMultiSectionDefaultCurve</CurveId>
   <CurveDescription>Default curve with multiple sections</CurveDescription>
   <Currency>USD</Currency>
   <Type>MultiSection</Type>
   <SourceCurves>
     <SourceCurve>Default/USD/Generic_AA_Curve</SourceCurve>
     <SourceCurve>Default/USD/Generic_B_Curve</SourceCurve>
     <SourceCurve>Default/USD/Generic_C_Curve</SourceCurve>
   </SourceCurves>
   <SwitchDates>
     <SwitchDate>2020-10-01</SwitchDate>
     <SwitchDate>2021-12-01</SwitchDate>
   <SwitchDates>
   <Extrapolation>true</Extrapolation>
   <DayCounter>A365</DayCounter>
   <RecoveryRate>RECOVERY_RATE/RATE/NAME/SR/USD</RecoveryRate>
</DefaultCurve>
```

Listing 62: Multi-Section default curve

7.8.5 Swaption Volatility Structures

Listing 63 shows an example of a Swaption volatility structure configuration.

```
<SwaptionVolatilities>
  <SwaptionVolatility>
    <CurveId>EUR_SW_N</CurveId>
    <CurveDescription>EUR normal swaption volatilities</CurveDescription>
    <Dimension>ATM</Dimension>
    <VolatilityType>Normal</VolatilityType>
    <Extrapolation>Flat</Extrapolation>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <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 section specification -->
    <SmileOptionTenors>6M,1Y,10Y</SmileOptionTenors>
    <SmileSwapTenors>2Y,5Y</SmileSwapTenors>
    <SmileSpreads>-0.02,-0.01,0.01,0.02</SmileSpreads>
  </SwaptionVolatility>
  ...
</SwaptionVolatilities>
```

Listing 63: Swaption volatility configuration

The meaning of each of the elements in Listing 63 is given below.

- CurveId: Unique identifier of the swaption volatility structure
- CurveDescription: A description of the volatility structure, may be left blank.
- Dimension: Distinguishes at-the-money matrices and full volatility cubes.
  Allowable values: `ATM, Smile`
- VolatilityType: Specifies the type of market volatility inputs.
  Allowable values: `Normal, Lognormal, ShiftedLognormal`
  In the case of `ShiftedLognormal`, a matrix of shifts (by option and swap tenor) has to be provided in the market data input.
- Extrapolation: Specifies the extrapolation behaviour in all dimensions.
  Allowable values: `Linear, Flat, None`
- DayCounter: The term structure's day counter used in date to time conversions
- Calendar: The term structure's calendar used in option tenor to date conversions
- BusinessDayConvention: The term structure's business day convention used in option tenor to date conversion
- ATM Matrix specification, required for both Dimension choices:
  - OptionTenors: Option expiry in period form
  - SwapTenors: Underlying Swap term in period form

  - ShortSwapIndexBase: Swap index (ORE naming convention, e.g. EUR-CMS-1Y) used to compute ATM strikes for tenors up to and including the tenor given in the index (1Y in this example)
  - SwapIndexBase: Swap index used to compute ATM strikes for tenors longer than the one defined by the short index

- Smile section specification, this part is required when Dimension is set to `Smile`, otherwise it can be omitted:
  - SmileOptionTenors: Option expiries, in period form, where smile section data is to be taken into account
  - SmileSwapTenors: Underlying Swap term, in period form, where smile section data is to be taken into account
  - SmileSpreads: Strikes in smile direction expressed as strike spreads, relative to the ATM strike at the expiry/term point of the ATM matrix

7.8.6 Cap Floor Volatility Structures

The cap volatility structure parameterisation allows the user to pick out term cap volatilities in the market data and define how they should be stripped to create an optionlet volatility structure. The parameterisation allows for three separate types of input term cap volatility structures:

1. A strip of at-the-money (ATM) cap volatilities.
2. A cap maturity tenor by absolute cap strike grid of cap volatilities.
3. A combined structure containing both the ATM cap volatilities and the maturity by strike grid of cap volatilities.

The input cap volatilities may be normal, lognormal or shifted lognormal. The structure of the market quotes is provided in Table 48.

The structure of the XML, i.e. the nodes that are necessary, used and ignored, and the way that the optionlet volatilities are stripped hinges on the value of the `InterpolateOn` node. This node may be set to `TermVolatilities` or `OptionletVolatilities`.

When set to `TermVolatilities`, a column of sequential caps or floors, are created for each strike level out to the maximum cap maturity configured. In other words, if the index tenor is 6M, the first cap created would have a maturity of 1Y, the second cap 18M, the third cap 2Y and so on until we have a cap with maturity equal to the maximum maturity tenor in the configuration. The volatility for each of these caps or floors is then interpolated from the term cap volatility surface using the configured interpolation. Finally, the optionlet volatility at each cap or floor maturity, starting from the first, is derived in turn such that the column of cap or floor volatilities are matched.

When set to `OptionletVolatilities`, the optionlet volatility structure pillar dates are set to the fixing dates on the last caplet on each of the configured caps or floors i.e. caps or floors with the maturities in the configured `Tenors` or `AtmTenors`. The optionlet volatilities on these pillar dates are then solved for such that the configured cap or floor volatilities are matched. In the following sections, we describe four XML configurations separately for clarity:

1. ATM curve with interpolation on term volatilities.
2. ATM curve with interpolation on optionlet volatilities.
3. Surface, possibly including an ATM column, with interpolation on term volatilities.
4. Surface, possibly including an ATM column, with interpolation on optionlet volatilities.

Listing 64 shows the layout for parameterising an ATM cap volatility curve with interpolation on term volatilities. Nodes that have no effect for this parameterisation but that are allowed by the schema are not referenced. The meaning of each of the nodes is as follows:

- `CurveId`: Unique identifier for the cap floor volatility structure.

- `CurveDescription` [Optional]: A description of the volatility structure. It is for information only and may be left blank.
- `VolatilityType`: Indicates the cap floor volatility type. It may be `Normal`, `Lognormal` or `ShiftedLognormal`. Note that this then determines which market data points are looked up in the market when creating the ATM cap floor curve and how they are interpreted when stripping the optionlets. In particular, the market will be searched for market data points of the form `CAPFLOOR/RATE_NVOL/Currency/Tenor/IndexTenor/1/1/0`, `CAPFLOOR/RATE_LNVOL/Currency/Tenor/IndexTenor/1/1/0` or `CAPFLOOR/RATE_SLNVOL/Currency/Tenor/IndexTenor/1/1/0` respectively.
- `Extrapolation`: Indicates the extrapolation in the time direction before the first optionlet volatility and after the last optionlet volatility. The extrapolation occurs on the stripped optionlet volatilities. The allowable values are `None`, `Flat` and `Linear`. If set to `None`, extrapolation is turned off and an exception is thrown if the optionlet surface is queried outside the allowable times. If set to `Flat`, the first optionlet volatility is used before the first time and the last optionlet volatility is used after the last time. If set to `Linear`, the interpolation method configured in `InterpolationMethod` is used to extrapolate.
- `InterpolationMethod` [Optional]: Indicates the interpolation in the time direction. As `InterpolateOn` is set to `TermVolatilities` here, the interpolation is used in the stripping process to interpolate the term cap floor volatility curve as explained above. It is also used to interpolate the optionlet volatilities when an optionlet volatility is queried from the stripped optionlet structure. The allowable values are `Bilinear` and `BicubicSpline`. If not set, `BicubicSpline` is assumed. Obviously, as we are describing an ATM curve here, there is no interpolation in the strike direction so when `Bilinear` is set the time interpolation is linear and when `BicubicSpline` is set the time interpolation is cubic spline.
- `IncludeAtm`: A boolean value indicating if an ATM curve should be used. Allowable boolean values are given in the Table 28. As we are describing an ATM curve here, this node should be set to `true` as shown in 64.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 17.
- `Calendar`: The calendar used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 16.
- `BusinessDayConvention`: The business day convention used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 14 under `Roll Convention`.
- `Tenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the ATM curve. If omitted, the tenors for the ATM curve must be provided in the `AtmTenors` node instead. If the tenors are provided here, the `AtmTenors` node may be omitted.

- `OptionalQuotes` [Optional]: A boolean flag to indicate wheather market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are povided. If false, the curve will fail to build if any quotes are missing. This also applites to quotes for the `AtmTenors`. Default value is false.
- `IborIndex`: A valid interest rate index name giving the index underlying the cap floor quotes. Allowable values are given in the Table 18.
- `DiscountCurve`: A reference to a valid discount curve specification that will be used to discount cashflows during the stripping process. It should be of the form `Yield/Currency/curve_name` where `curve_name` is the name of a yield curve defined in the yield curve configurations.
- `AtmTenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturities to be used in the ATM curve. If omitted, the tenors for the ATM curve must be provided in the `Tenors` node instead. If the tenors are provided here, the `Tenors` node may be omitted.
- `SettlementDays` [Optional]: Any non-negative integer is allowed here. If omitted, it is assumed to be 0. If provided the reference date of the term volatility curve and the stripped optionlet volatility structure will be calculated by advancing the valuation date by this number of days using the configured calendar and business day convention. In general, this should be omitted or set to 0.
- `InterpolateOn`: As referenced above, the allowable values are `TermVolatilities` or `OptionletVolatilities`. As we are describing here an ATM curve with interpolation on term volatilities, this should be set to `TermVolatilities` as shown in Listing 64.
- `BootstrapConfig` [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's default values described in section 7.8.19 are used.

```xml
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <Extrapolation>...</Extrapolation>
  <InterpolationMethod>...</InterpolationMethod>
  <IncludeAtm>true</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>...</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <InterpolateOn>TermVolatilities</InterpolateOn>
  <BootstrapConfig>...</BootstrapConfig>
</CapFloorVolatility>
```

Listing 64: ATM cap floor configuration with interpolation on term volatilities.

Listing 65 shows the layout for parameterising an ATM cap volatility curve with interpolation on optionlet volatilities. Nodes that have no effect for this parameterisation but that are allowed by the schema are not referenced. The meaning of each of the nodes is as follows:

- `CurveId`: Unique identifier for the cap floor volatility structure.
- `CurveDescription` [Optional]: A description of the volatility structure. It is for information only and may be left blank.
- `VolatilityType`: Indicates the cap floor volatility type. It may be `Normal`, `Lognormal` or `ShiftedLognormal`. Note that this then determines which market data points are looked up in the market when creating the ATM cap floor curve and how they are interpreted when stripping the optionlets. In particular, the market will be searched for market data points of the form `CAPFLOOR/RATE_NVOL/Currency/Tenor/IndexTenor/1/1/0`, `CAPFLOOR/RATE_LNVOL/Currency/Tenor/IndexTenor/1/1/0` or `CAPFLOOR/RATE_SLNVOL/Currency/Tenor/IndexTenor/1/1/0` respectively.
- `Extrapolation`: The allowable values are `None`, `Flat` and `Linear`. If set to `None`, extrapolation is turned off and an exception is thrown if the optionlet surface is queried outside the allowable times. Otherwise, extrapolation is allowed and the type of extrapolation is determined by the `TimeInterpolation` node value described below.
- `IncludeAtm`: A boolean value indicating if an ATM curve should be used. Allowable boolean values are given in the Table 28. As we are describing an ATM curve here, this node should be set to `true` as shown in 65.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 17.
- `Calendar`: The calendar used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 16.
- `BusinessDayConvention`: The business day convention used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 14 under `Roll Convention`.
- `Tenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the ATM curve. If omitted, the tenors for the ATM curve must be provided in the `AtmTenors` node instead. If the tenors are provided here, the `AtmTenors` node may be omitted.
- `OptionalQuotes` [Optional]: A boolean flag to indicate wheather market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are povided. If false, the curve will fail to build if any quotes are missing. This also applites to quotes for the `AtmTenors`. Default value is false.
- `IborIndex`: A valid interest rate index name giving the index underlying the cap floor quotes. Allowable values are given in the Table 18.

- DiscountCurve: A reference to a valid discount curve specification that will be used to discount cashflows during the stripping process. It should be of the form Yield/Currency/curve_name where curve_name is the name of a yield curve defined in the yield curve configurations.
- AtmTenors [Optional]: A comma separated list of valid tenor strings giving the cap floor maturities to be used in the ATM curve. If omitted, the tenors for the ATM curve must be provided in the Tenors node instead. If the tenors are provided here, the Tenors node may be omitted.
- SettlementDays [Optional]: Any non-negative integer is allowed here. If omitted, it is assumed to be 0. If provided the reference date of the term volatility curve and the stripped optionlet volatility structure will be calculated by advancing the valuation date by this number of days using the configured calendar and business day convention. In general, this should be omitted or set to 0.
- InterpolateOn: As referenced above, the allowable values are TermVolatilities or OptionletVolatilities. As we are describing here an ATM curve with interpolation on optionlet volatilities, this should be set to OptionletVolatilities as shown in Listing 65.
- TimeInterpolation [Optional]: Indicates the interpolation and extrapolation, if allowed by the Extrapolation node, in the time direction. As InterpolateOn is set to OptionletVolatilities here, the interpolation is used to interpolate the optionlet volatilities only i.e. there is no interpolation on the term cap floor volatility curve. The allowable values are Linear, LinearFlat, BackwardFlat, Cubic and CubicFlat. If not set, LinearFlat is assumed. Note that Linear indicates linear interpolation and linear extrapolation. LinearFlat indicates linear interpolation and flat extrapolation. Analogous meanings apply for Cubic and CubicFlat.
- BootstrapConfig [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's default values described in section 7.8.19 are used.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <Extrapolation>...</Extrapolation>
  <IncludeAtm>true</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>...</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <InterpolateOn>OptionletVolatilities</InterpolateOn>
  <TimeInterpolation>...</TimeInterpolation>
  <BootstrapConfig>...</BootstrapConfig>
</CapFloorVolatility>
```

Listing 65: ATM cap floor configuration with interpolation on optionlet volatilities.

Listing 66 shows the layout for parameterising a cap tenor by absolute cap strike volatility surface with interpolation on term volatilities. This parameterisation also allows for the inclusion of a cap floor ATM curve in combination with the surface. Nodes that have no effect for this parameterisation but that are allowed by the schema are not referenced. The meaning of each of the nodes is as follows:

- `CurveId`: Unique identifier for the cap floor volatility structure.
- `CurveDescription` [Optional]: A description of the volatility structure. It is for information only and may be left blank.
- `VolatilityType`: Indicates the cap floor volatility type. It may be `Normal`, `Lognormal` or `ShiftedLognormal`. Note that this then determines which market data points are looked up in the market when creating the cap floor surface and how they are interpreted when stripping the optionlets. In particular, the market will be searched for market data points of the form `CAPFLOOR/RATE_NVOL/Currency/Tenor/IndexTenor/0/0/Strike`, `CAPFLOOR/RATE_LNVOL/Currency/Tenor/IndexTenor/0/0/Strike` or `CAPFLOOR/RATE_SLNVOL/Currency/Tenor/IndexTenor/0/0/Strike` respectively.
- `Extrapolation`: Indicates the extrapolation in the time and strike direction. The extrapolation occurs on the stripped optionlet volatilities. The allowable values are `None`, `Flat` and `Linear`. If set to `None`, extrapolation is turned off and an exception is thrown if the optionlet surface is queried outside the allowable times or strikes. If set to `Flat`, the optionlet volatility on the time strike boundary is used if the optionlet surface is queried outside the allowable times or strikes. If set to `Linear`, the interpolation method configured in `InterpolationMethod` is used to extrapolate either time or strike direction.
- `InterpolationMethod` [Optional]: Indicates the interpolation in the time and strike direction. As `InterpolateOn` is set to `TermVolatilities` here, the interpolation is used in the stripping process to interpolate the term cap floor volatility surface as explained above. It is also used to interpolate the optionlet volatilities when an optionlet volatility is queried from the stripped optionlet structure. The allowable values are `Bilinear` and `BicubicSpline`. If not set, `BicubicSpline` is assumed.
- `IncludeAtm`: A boolean value indicating if an ATM curve should be used in combination with the surface. Allowable boolean values are given in the Table 28. If set to `true`, the `AtmTenors` node needs to be populated with the ATM tenors to use. The ATM quotes that are searched for are as outlined in the previous two ATM sections above. The original stripped optionlet surface is amended by inserting the optionlet volatilities at the successive ATM strikes that reproduce the sequence of ATM cap volatilities.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 17.
- `Calendar`: The calendar used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the

configured cap tenors. Allowable values are given in the Table 16.

- BusinessDayConvention: The business day convention used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 14 under Roll Convention.
- Tenors: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the tenor by strike surface. In this case, i.e. configuring a surface, they must be provided.
- OptionalQuotes [Optional]: A boolean flag to indicate wheather market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are povided. If false, the curve will fail to build if any quotes are missing. This also applites to quotes for the AtmTenors. Default value is false.
- IborIndex: A valid interest rate index name giving the index underlying the cap floor quotes. Allowable values are given in the Table 18.
- DiscountCurve: A reference to a valid discount curve specification that will be used to discount cashflows during the stripping process. It should be of the form Yield/Currency/curve_name where curve_name is the name of a yield curve defined in the yield curve configurations.
- AtmTenors [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the ATM curve. It must be provided when IncludeAtm is true and omitted when IncludeAtm is false.
- SettlementDays [Optional]: Any non-negative integer is allowed here. If omitted, it is assumed to be 0. If provided the reference date of the term volatility curve and the stripped optionlet volatility structure will be calculated by advancing the valuation date by this number of days using the configured calendar and business day convention. In general, this should be omitted or set to 0.
- InterpolateOn: As referenced above, the allowable values are TermVolatilities or OptionletVolatilities. As we are describing here a surface with interpolation on term volatilities, this should be set to TermVolatilities as shown in Listing 66.
- BootstrapConfig [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's default values described in section 7.8.19 are used.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <Extrapolation>...</Extrapolation>
  <InterpolationMethod>...</InterpolationMethod>
  <IncludeAtm>...</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>...</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <InterpolateOn>TermVolatilities</InterpolateOn>
  <BootstrapConfig>...</BootstrapConfig>
</CapFloorVolatility>
```

Listing 66: Cap floor surface with interpolation on term volatilities.

Listing 67 shows the layout for parameterising a cap tenor by absolute cap strike volatility surface with interpolation on optionlet volatilities. This parameterisation also allows for the inclusion of a cap floor ATM curve in combination with the surface. Nodes that have no effect for this parameterisation but that are allowed by the schema are not referenced. The meaning of each of the nodes is as follows:

- `CurveId`: Unique identifier for the cap floor volatility structure.
- `CurveDescription` [Optional]: A description of the volatility structure. It is for information only and may be left blank.
- `VolatilityType`: Indicates the cap floor volatility type. It may be `Normal`, `Lognormal` or `ShiftedLognormal`. Note that this then determines which market data points are looked up in the market when creating the cap floor surface and how they are interpreted when stripping the optionlets. In particular, the market will be searched for market data points of the form
`CAPFLOOR/RATE_NVOL/Currency/Tenor/IndexTenor/0/0/Strike`,
`CAPFLOOR/RATE_LNVOL/Currency/Tenor/IndexTenor/0/0/Strike` or
`CAPFLOOR/RATE_SLNVOL/Currency/Tenor/IndexTenor/0/0/Strike` respectively.
- `Extrapolation`: The allowable values are `None`, `Flat` and `Linear`. If set to `None`, extrapolation is turned off and an exception is thrown if the optionlet surface is queried outside the allowable times or strikes. Otherwise, extrapolation is allowed and the type of extrapolation is determined by the `TimeInterpolation` and `StrikeInterpolation` node values described below.
- `IncludeAtm`: A boolean value indicating if an ATM curve should be used in combination with the surface. Allowable boolean values are given in the Table 28. If set to `true`, the `AtmTenors` node needs to be populated with the ATM tenors to use. The ATM quotes that are searched for are as outlined in the previous two ATM sections above. The original stripped optionlet surface is amended by inserting the optionlet volatilities at the configured ATM strikes that reproduce the configured ATM cap volatilities.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 17.
- `Calendar`: The calendar used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the Table 16.
- `BusinessDayConvention`: The business day convention used to advance dates by periods in the underlying structure. In particular, it is used in deriving the cap maturity dates from the configured cap tenors. Allowable values are given in the

Table 14 under `Roll Convention`.

- `Tenors`: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the tenor by strike surface. In this case, i.e. configuring a surface, they must be provided.
- `OptionalQuotes` [Optional]: A boolean flag to indicate wheather market data quotes for all tenors and strikes are required. If true, we attempt to build the curve from whatever quotes are povided. If false, the curve will fail to build if any quotes are missing. This also applites to quotes for the `AtmTenors`. Default value is false.
- `IborIndex`: A valid interest rate index name giving the index underlying the cap floor quotes. Allowable values are given in the Table 18.
- `DiscountCurve`: A reference to a valid discount curve specification that will be used to discount cashflows during the stripping process. It should be of the form `Yield/Currency/curve_name` where `curve_name` is the name of a yield curve defined in the yield curve configurations.
- `AtmTenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the ATM curve. It must be provided when `IncludeAtm` is `true` and omitted when `IncludeAtm` is `false`.
- `SettlementDays` [Optional]: Any non-negative integer is allowed here. If omitted, it is assumed to be 0. If provided the reference date of the term volatility curve and the stripped optionlet volatility structure will be calculated by advancing the valuation date by this number of days using the configured calendar and business day convention. In general, this should be omitted or set to 0.
- `InterpolateOn`: As referenced above, the allowable values are `TermVolatilities` or `OptionletVolatilities`. As we are describing here a surface with interpolation on optionlet volatilities, this should be set to `OptionletVolatilities` as shown in Listing 67.
- `TimeInterpolation`: Indicates the interpolation and extrapolation, if allowed by the `Extrapolation` node, in the time direction. As `InterpolateOn` is set to `OptionletVolatilities` here, the interpolation is used to interpolate the optionlet volatilities only i.e. there is no interpolation on the term cap floor volatility curve. The allowable values are `Linear`, `LinearFlat`, `BackwardFlat`, `Cubic` and `CubicFlat`. If not set, `LinearFlat` is assumed. Note that `Linear` indicates linear interpolation and linear extrapolation. `LinearFlat` indicates linear interpolation and flat extrapolation. Analogous meanings apply for `Cubic` and `CubicFlat`.
- `StrikeInterpolation`: Indicates the interpolation and extrapolation, if allowed by the `Extrapolation` node, in the strike direction. Again, as `InterpolateOn` is set to `OptionletVolatilities` here, the interpolation is used to interpolate the optionlet volatilities in the strike direction. The allowable values are `Linear`, `LinearFlat`, `Cubic` and `CubicFlat`. If not set, `LinearFlat` is assumed.
- `BootstrapConfig` [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 7.8.19. If omitted, this node's

default values described in section 7.8.19 are used.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <Extrapolation>...</Extrapolation>
  <InterpolationMethod>...</InterpolationMethod>
  <IncludeAtm>...</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>...</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <InterpolateOn>OptionletVolatilities</InterpolateOn>
  <TimeInterpolation>...</TimeInterpolation>
  <StrikeInterpolation>...</StrikeInterpolation>
  <BootstrapConfig>...</BootstrapConfig>
</CapFloorVolatility>
```

Listing 67: Cap floor surface with interpolation on optionlet volatilities.

7.8.7 FX Volatility Structures

Listings 68, 69, 70, 71, 72 shows examples of FX volatility structure configurations.

```
<FXVolatility>
  <CurveId>EURUSD</CurveId>
  <CurveDescription />
  <Dimension>ATM</Dimension>
  <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
  <FXSpotID>FX/EUR/USD</FXSpotID>
  <FXForeignCurveID>Yield/EUR/ERU-IN-USD</FXForeignCurveID>
  <FXDomesticCurveID>Yield/USD/USD1D</FXDomesticCurveID>
  <DayCounter>A365</DayCounter>
  <Calendar>US,TARGET<Calendar>
  <Conventions>EUR-USD-FXOPTION</Conventions>
</FXVolatility>
```

Listing 68: FX option volatility configuration ATM

```
<FXVolatility>
  <CurveId>USDJPY</CurveId>
  <CurveDescription />
  <Dimension>Smile</Dimension>
  <SmileType>VannaVolga</SmileType>
  <SmileInterpolation>VannaVolga2</SmileInterpolation>
  <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
  <SmileDelta>25</SmileDelta>
  <FXSpotID>FX/USD/JPY</FXSpotID>
  <FXForeignCurveID>Yield/USD/USD1D</FXForeignCurveID>
  <FXDomesticCurveID>Yield/JPY/JPY-IN-USD</FXDomesticCurveID>
  <DayCounter>A365</DayCounter>
  <Calendar>US,JP<Calendar>
  <Conventions>USD-JPY-FXOPTION</Conventions>
</FXVolatility>
```

Listing 69: FX option volatility configuration Smile / VannaVolga

```
<FXVolatility>
  <CurveId>USDJPY</CurveId>
  <CurveDescription />
  <Dimension>Smile</Dimension>
  <SmileType>Delta</SmileType>
  <SmileInterpolation>Linear</SmileInterpolation>
  <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
  <Deltas>10P,20P,30P,40P,ATM,40C,30C,20C,10C</Deltas>
  <FXSpotID>FX/USD/JPY</FXSpotID>
  <FXForeignCurveID>Yield/USD/USD1D</FXForeignCurveID>
  <FXDomesticCurveID>Yield/JPY/JPY-IN-USD</FXDomesticCurveID>
  <DayCounter>A365</DayCounter>
  <Calendar>US,JP<Calendar>
  <Conventions>USD-JPY-FXOPTION</Conventions>
</FXVolatility>
```

Listing 70: FX option volatility configuration Smile / Delta

```
<FXVolatility>
  <CurveId>USDJPY</CurveId>
  <CurveDescription />
```

```
    <Dimension>Smile</Dimension>
    <SmileType>BFRR</SmileType>
    <SmileInterpolation>Cubic</SmileInterpolation>
    <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
    <SmileDelta>10,25</SmileDelta>
    <FXSpotID>FX/USD/JPY</FXSpotID>
    <FXForeignCurveID>Yield/USD/USD1D</FXForeignCurveID>
    <FXDomesticCurveID>Yield/JPY/JPY-IN-USD</FXDomesticCurveID>
    <DayCounter>A365</DayCounter>
    <Calendar>US,JP<Calendar>
    <Conventions>USD-JPY-FXOPTION</Conventions>
  </FXVolatility>
```

Listing 71: FX option volatility configuration Smile / BFRR with 10 and 25 BF and RR

```
  <FXVolatility>
    <CurveId>EURJPY</CurveId>
    <CurveDescription />
    <Dimension>ATMTriangulated</Dimension>
    <FXSpotID>FX/EUR/JPY</FXSpotID>
    <DayCounter>A365</DayCounter>
    <Calendar>US,JP<Calendar>
    <BaseVolatility1>EURUSD</BaseVolatility1>
    <BaseVolatility2>USDJPY</BaseVolatility2>
  </FXVolatility>
```

Listing 72: FX option volatility configuration ATM Triangulated

The meaning of each of the elements in Listings 68, 69, 70, 71, 72 is given below.

- CurveId: Unique identifier of the FX volatility structure
- CurveDescription: A description of the volatility structure, may be left blank.
- Dimension: Distinguishes at-the-money volatility curves from volatility surfaces. An 'ATMTriangulated' value denotes a curve triangulated from two other surfaces.
  Allowable values: `ATM, Smile, ATMTriangulated`
- SmileType [Optional]: Required field in case of Dimension `Smile`, otherwise it can be omitted.
  Allowable values: `VannaVolga` as per (Castagna & Mercurio - 2006), `Delta`, `BFRR`, with default value `VannaVolga` if left blank.
- SmileInterpolation [Optional]: Smile interpolation method applied, required field in case of Dimension `Smile`, otherwise it can be omitted.
  Allowable values:
  - `VannaVolga1` or `VannaVolga2` in case of SmileType `VannaVolga` with default VannaVolga2 if left blank. VannaVolga1/VannaVolga2 refer to the first/second approximation in (eq. 13) and (eq. 14) of the reference above.
  - `Linear` or `Cubic` in case of SmileType `Delta` or `BFRR` with default `Linear` for SmileType Delta and `Cubic` for SmileType BFRR if left blank
- Expiries: Option expiries in period form. A wildcard may also be used. In the wildcard case, it will look for any matching quotes provided in the loader, and

construct the curve from these. This is currently only supported for `ATM` or `Delta` or `BFRR` curves.

- Deltas [Optional]: Strike grid, required in case of SmileType `Delta`
  Allowable values: `ATM, *P, *C`, see example in Listing 70
- SmileDeltas [Optional]: Strike grid for SmileType `VannaVolga` and `BFRR`, defaults to 25 for VannaVolga resp. 10,25 for BFRR
  Allowable values: a list of integers, see example in Listing 71
- FXSpotID: ORE representation of the relevant FX spot quote in the form FX/CCY1/CCY2
- FXForeignCurveID [Optional]: Yield curve, in ORE format Yield/CCY/ID, used as foreign yield curve in smile curves, may be omitted for ATM curves.
- FXDomesticCurveID [Optional]: Yield curve, in ORE format Yield/CCY/ID, used as domestic yield curve in smile curves, may be omitted for ATM curves.
- DayCounter: The term structure's day counter used in date to time conversions. Optional, defaults to A365.
- Calendar: The term structure's calendar used in option tenor to date conversions. Optional, defaults to source ccy + target ccy default calendars.
- Conventions [Optional]: FX conventions object ID that is used to determine the at-the-money type and delta type of the volatility quotes, these default to `AtmDeltaNeutral` and `Spot` for option tenors <= 2Y and `AtmDeltaNeutral` and `Fwd` for option tenors > 2Y if the conventions ID is omitted or left blank. Furthermore, the conventions hold information on the side of risk reversals (RiskReversalInFavorOf, defaults to `Call`) and the quote style of butterflies (ButterflyStyle, defaults to `Broker`), these latter two are relevant for BF, RR market data input. See 7.11.18 for more details.
- BaseVolatility1: For 'ATMTriangulated' this denotes one of the surfaces we want to triangulate from
- BaseVolatility2: For 'ATMTriangulated' this denotes one of the surfaces we want to triangulate from

7.8.8 Equity Curve Structures

Listing 73 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>
    <!-- DividendYield, ForwardPrice, OptionPremium, NoDividends, ForwardDividendPrice -->
    <Type>DividendYield</Type>
    <!-- Optional node, only used when Type is OptionPremium -->
    <ExerciseStyle>American</ExerciseStyle>
    <!-- Spot quote from the market data file -->
    <SpotQuote>EQUITY/PRICE/SP5/USD</SpotQuote>
    <!-- Note: do not provide Quotes if Type is NoDividends -->
    <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>
    <!-- Optional interpolation options, default Zero and Linear -->
    <!-- Note: do not provide DividendInterpolation if Type is NoDividends -->
    <DividendInterpolation>
      <!-- Zero, Discount -->
      <InterpolationVariable>Zero</InterpolationVariable>
      <!-- See \ref{tab:allow_interp_methods} -->
      <InterpolationMethod>LogLinear</InterpolationMethod>
    </DividendInterpolation>
    <!-- Optional node, defaults to true -->
    <Extrapolation>True</Extrapolation>
    <DayCounter>A365</DayCounter>
  </EquityCurve>
  <EquityCurve>
    ...
  </EquityCurve>
  </EquityCurves>
```

Listing 73: 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. If the index is a dividend futures index then curve type should be entered as ForwardDividendPrice. In this case the curve will be built from forward prices as normal, but excluded from the SIMM calculations as required by the SIMM methodology. 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.

7.8.9 Equity Volatility Structures

When configuring volatility structures for equities, there are four options:

1. a constant volatility for all expiries and strikes.
2. a volatility curve with a dependency on expiry but no strike dimension.
3. a volatility surface with an expiry and strike dimension.
4. a proxy surface - point to another surface as a proxy.

There are a number of fields common to all configurations:

- CurveId: Unique identifier for the curve.
- CurveDescription: A description of the curve. This field may be left blank.
- Currency: Currency of the equity.
- Calendar [Optional]: allowable value is any valid calendar. Defaults to `NullCalendar`.

- DayCounter [Optional]: allowable value is any valid day counter. Defaults to `A365`.
- OneDimSolverConfig [Optional]: A configuration for the one dimensional solver used in deriving volatilities from prices. This node is described in detail in Section 7.8.20. If not provided, a default step based configuration is used. This is only used when volatilities are stripped from prices.
- PreferOutOfTheMoney [Optional]: allowable value is any boolean value. Defaults to `false` for backwards compatibility. It is used, when building the volatility surface, to choose between a call and put option price or volatility when both are provided. When set to `true`, the out of the money option is chosen by comparing the quote's strike with the forward price at the associated expiry. Conversely, when set to `false`, the in the money option is chosen.

Listing 74 shows the configuration of equity volatility structures with constant volatility. A node `Constant` takes one `Quote`, as described in Section 10.24, which is held constant for all strikes and expiries.

```xml
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <Constant>
      <Quote>EQUITY_OPTION/RATE_LNVOL/SP5/USD/5Y/ATMF</Quote>
    </Constant>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 74: Equity option volatility configuration - constant

Secondly, the volatility curve configuration layout is given in Listing 75. With this curve construction the volatility is held constant in the strike direction, and quotes of varying expiry can be provided, for examlple if only ATM volatility quotes are available. The volatility quote IDs in the `Quotes` node should be Equity option volatility quotes as described in Section 10.24. An explicit list of quotes can be provided, or a single quote with a wildcard replacing the expiry/strike. In the wildcard case, it will look for any matching quotes provided in the loader, and construct the curve from these. The `Interpolation` node supports `Linear`, `Cubic` and `LogLinear` interpolation. The `Extrapolation` node supports either `None` for no extrapolation or `Flat` for flat extrapolation in the volatility.

```xml
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <Curve>
      <QuoteType>ImpliedVolatility</QuoteType>
      <VolatilityType>Lognormal</VolatilityType>
      <Quotes>
        <Quote>EQUITY_OPTION/RATE_LNVOL/SP5/USD/*</Quote>
      </Quotes>
      <Interpolation>LinearFlat</Interpolation>
      <Extrapolation>Flat</Extrapolation>
    </Curve>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 75: Equity option volatility configuration - curve

The volatility strike surface configuration layout is given in Listing 76. This allows a full surface of `Strikes` and `Expiries` to be defined. The following are the valid nodes:

- `QuoteType`: either `ImpliedVolatility` of `Premium`, indicating the type of quotes provided in the market.
- `ExerciseType` [Optional]: only valid when `QuoteType` is `Premium`. Valid types are `European` and `American`.
- `VolatilityType` [Optional]: only valid when `QuoteType` is `ImpliedVolatility`. Valid types are `Lognormal`, `ShiftedLognormal` and `Normal`.
- `Strikes`: comma separated list of strikes, representing the absolute strike values for the option. In other words, A single wildcard character, `*`, can be used here also to indicate that all strikes found in the market data for this equity volatility configuration should be used when building the equity volatility surface.
- `Expiries`: comma separated list of expiry tenors and or expiry dates. A single wildcard character, `*`, can be used here also to indicate that all expiries found in the market data for this equity volatility configuration should be used when building the equity volatility surface.
- `TimeInterpolation`: interpolation in the option expiry direction. If either `Strikes` or `Expiries` are configured with a wildcard character, `Linear` is used. If both `Strikes` and `Expiries` are configured explicitly, `Linear` or `Cubic` is allowed here but the value must agree with the value for `StrikeInterpolation`.
- `StrikeInterpolation`: interpolation in the strike direction. If either `Strikes` or `Expiries` are configured with a wildcard character, `Linear` is used. If both `Strikes` and `Expiries` are configured explicitly, `Linear` or `Cubic` is allowed here but the value must agree with the value for `TimeInterpolation`.
- `Extrapolation`: boolean value. If `true`, extrapolation is allowed. If `false`, extrapolation is not allowed.
- `TimeExtrapolation`: extrapolation in the option expiry direction. If both `Strikes` and `Expiries` are configured explicitly, the extrapolation in the time direction is flat in volatility regardless of the setting here. If either `Strikes` or `Expiries` are configured with a wildcard character, `Linear`, `UseInterpolator`, `Flat` or `None` are allowed. If `Linear` or `UseInterpolator` is specified, the

extrapolation is linear. If `Flat` is specified, the extrapolation is flat. If `None` is specified, it is ignored and the extrapolation is flat since extrapolation in the time direction cannot be turned off in isolation i.e. extrapolation can only be turned off for the surface as a whole using the `Extrapolation` flag.

- `StrikeExtrapolation`: extrapolation in the strike direction. The allowable values are `Linear`, `UseInterpolator`, `Flat` or `None`. If `Linear` or `UseInterpolator` is specified, the extrapolation uses the strike interpolation setting for extrapolation i.e. linear or cubic in this case. If `Flat` is specified, the extrapolation is flat. If `None` is specified, it is ignored and the extrapolation is flat since extrapolation in the strike direction cannot be turned off in isolation i.e. extrapolation can only be turned off for the surface as a whole using the `Extrapolation` flag.

When this configuration is used, the market is searched for quote strings of the form `EQUITY_OPTION/PRICE/[NAME]/[CURRENCY]/[EXPIRY]/[STRIKE]` or `EQUITY_OPTION/RATE_LNVOL/[NAME]/[CURRENCY]/[EXPIRY]/[STRIKE]`, depending on the `QuoteType`. When both the `Strikes` and `Expiries` are configured explicitly, it is clear that the `[EXPIRY]` field is populated from the list of expiries in turn and the `[STRIKE]` field is populated from the list of strikes in turn. If there are m expiries in the `Expiries` list and n strikes in the `Strikes` list, there will be $m \times n$ quotes created and searched for in the market data. If `Expiries` are configured via the wildcard character, `*`, all quotes in the market data matching the pattern `EQUITY_OPTION/RATE_LNVOL/[NAME]/[CURRENCY]/*/[STRIKE]`. Similarly for `Strikes` configured via the wildcard character, `*`.

```
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <StrikeSurface>
      <QuoteType>Premium</QuoteType>
      <ExerciseType>European</ExerciseType>
      <Strikes>*</Strikes>
      <Expiries>*</Expiries>
      <TimeInterpolation>Linear</TimeInterpolation>
      <StrikeInterpolation>Linear</StrikeInterpolation>
      <Extrapolation>true</Extrapolation>
      <TimeExtrapolation>UseInterpolator</TimeExtrapolation>
      <StrikeExtrapolation>Flat</StrikeExtrapolation>
    </StrikeSurface>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 76: Equity option volatility configuration - strike surface

A volatility surface can also be given in terms of moneyness levels as shown in listing 77. The nodes have the same meaning as in the case of a strike surface with the following exceptions:

- `QuoteType`: only `ImpliedVolatility` is allowed
- `VolatilityType` [Optional]: only `Lognormal` is allowed
- `MoneynessType`: `Spot` or `Fwd`, indicating the type of moneyness. See 10.24 for the definition of moneyness types.
- `MoneynessLevels`: comma separated list of moneyness levels, no wild cards are allowed.
- `Expiries`: comma separated list of expiry tenors and or expiry dates. A single wildcard character, `*`, can be used here also to indicate that all expiries found in the market data for this equity volatility configuration should be used when building the equity volatility surface.

Notice that the market data for the moneyness level 1.0 must be given as a moneyness quote, not an ATM or ATMF quote, see 10.24 for details of the market data.

```
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <MoneynessSurface>
      <QuoteType>ImpliedVolatility</QuoteType>
      <VolatilityType>Lognormal</VolatilityType>
      <MoneynessType>Fwd</MoneynessType>
      <MoneynessLevels>0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2,1.3,1.4,1.5</MoneynessLevels>
      <Expiries>*</Expiries>
      <TimeInterpolation>Linear</TimeInterpolation>
      <StrikeInterpolation>Linear</StrikeInterpolation>
      <Extrapolation>true</Extrapolation>
      <TimeExtrapolation>UseInterpolator</TimeExtrapolation>
      <StrikeExtrapolation>Flat</StrikeExtrapolation>
    </MoneynessSurface>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 77: Equity option volatility configuration - moneyness surface

Finally, the volatility proxy surface configuration layout is given in Listing 78. This allows us to use any other surface as a proxy, in cases where there is no option data for a given equity. We provide a name in the `ProxySurface` field, which must match the `CurveId` of another configuration. The proxy surface looks up the volatility in the reference surface bases on moneyness.

```
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>ABC</CurveId>
    <CurveDescription>Lognormal option implied vols for APC - proxied from SP5</CurveDescription>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <ProxySurface>SP5</PROXYSurface>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 78: Equity option volatility configuration - proxy surface

7.8.10 Inflation Curves

Listing 79 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>
               <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>
           <OverrideFactors>
               1.00,1.00,1.00,1.00,1.00,1.00,1.00,1.00,1.00,1.00,1.00,1.00
           </OverrideFactors>
       </Seasonality>
   </InflationCurve>
</InflationCurves>
```

Listing 79: 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.
- `OverrideFactors:` A numeric list of seaosonality correction factors, replacing the Factors. This allows to specify a static seasonality correction that does not require market data quotes. If both Factors and OverrideFactors are given, the OverrideFactors are used. Otherwise only one of Factors, OverrideFactors is required in a seasonality block.

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.

7.8.11 Inflation Cap/Floor Volatility Surfaces

Listing 80 shows the configuration of an zero coupon inflation cap floor price surface.

```
<InflationCapFloorVolatility>
    <CurveId>EUHICPXT_ZC_CF</CurveId>
    <CurveDescription>
        EUHICPXT CPI Cap/Floor vol surface derived from price quotes
    </CurveDescription>
    <Type>ZC</Type>
    <QuoteType>Price</QuoteType>
    <VolatilityType>Normal</VolatilityType>
    <Extrapolation>Y</Extrapolation>
    <DayCounter>ACT</DayCounter>
    <Calendar>TARGET</Calendar>
    <BusinessDayConvention>MF</BusinessDayConvention>
    <Tenors>1Y,2Y,3Y,4Y,5Y,6Y,7Y,8Y,9Y,10Y,12Y,15Y,20Y,30Y</Tenors>
    <!-- QuoteType 'Volatility' requires <Strikes>: -->
    <!-- <Strikes>-0.02,-0.01,-0.005,0.00,0.01,0.015,0.02,0.025,0.03</Strikes> -->
    <!-- QuoteType 'Price' requires <CapStrikes> and/or <FloorStrikes>: -->
    <CapStrikes/>
    <FloorStrikes>-0.02,-0.01,-0.005,0.00,0.01,0.015,0.02,0.025,0.03</FloorStrikes>
    <Index>EUHICPXT</Index>
    <IndexCurve>Inflation/EUHICPXT/EUHICPXT_ZC_Swaps</IndexCurve>
    <IndexInterpolated>false</IndexInterpolated>
    <ObservationLag>3M</ObservationLag>
    <YieldTermStructure>Yield/EUR/EUR1D</YieldTermStructure>
    <QuoteIndex>...</QuoteIndex>
</InflationCapFloorVolatility>
```

Listing 80: Inflation ZC cap floor volatility surface configuration

- `Type`: The type of the surface, `ZC` for zero coupon, `YY` for year on year.
- `QuoteType`: The type of the quotes used to build the surface, `Volatility` for volatility quotes, `Price` for bootstrap from option premia.
- `VolatilityType`: If QuoteType is `Volatility`, this specifies whether the input is `Normal`, `Lognormal` or `ShiftedLognormal`.
- `Extrapolation`: Boolean to indictae whether the surface should allow extrapolation.
- `DayCounter`: The term structure's day counter
- `Calendar`: The term structure's calendar
- `BusinessDayConvention`: The term structure's business day convention
- `Tenors`: The maturities for which cap and floor prices are quoted
- `Strikes`: In the case of QuoteType `Volatility`, the strikes for which floor prices are quoted (may, and will usually, overlap with the cap strike region); neither CapStrikes nor FloorStrikes are expected in this case.
- `CapStrikes`: The strikes for which cap prices are quoted (may, and will usually, overlap with the floor strike region); if the QuoteType above is `Price`, either CapStrikes or FloorStrikes or both are required.
- `FloorStrikes`: The strikes for which floor prices are quoted (may and will usually) overlap with the cap strike region); if the QuoteType above is `Price`, either CapStrikes or FloorStrikes or both are required.
- `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.
- `Observation Lag`: The observation lag applicable to the term structure.
- `YieldTermStructure`: The nominal term structure.

- QuoteIndex: An optional node allowing the user to provide an alternative index name for forming the quotes that will be used in building the cap floor surface. If this node is not provided, the Index node value is used in quote construction. For example, quotes must be created from each strike and each tenor and these quotes are subsequently looked up in the market data when building the cap floor volatility surface. The quotes are formed by concatenating [Type]_INFLATIONCAPFLOOR, PRICE or RATE_[Vol_Type]VOL, [Index_Name], [Tenor], C or F and [Strike] delimited by /. If QuoteIndex is provided, it is used as the [Index_Name] token. If it is not provided Index is used as usual.

7.8.12 CDS Volatilities

When configuring volatility structures for CDS and index CDS options, there are three options:

1. a constant volatility for all expiries and strikes.
2. a volatility curve with a dependency on expiry but no strike dimension.
3. a volatility surface with an expiry and strike dimension.

Firstly, the constant volatility configuration layout is given in Listing 81. The single volatility quote ID, constant_quote_id, in the Quote node should be a CDS option volatility quote as described in Section 10.15. The DayCounter node is optional and defaults to A365F. The Calendar node is optional and defaults to NullCalendar. The DayCounter and Calendar nodes are common to all three CDS volatility configurations.

```
<CDSVolatility>
  <CurveId>..<CurveId>
  <CurveDescription>...</CurveDescription>
  <Constant>
    <Quote>constant_quote_id</Quote>
  </Constant>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
</CDSVolatility>
```

Listing 81: Constant CDS volatility configuration

Secondly, the volatility curve configuration layout is given in Listing 82. The volatility quote IDs, quote_id_1, quote_id_2, etc., in the Quotes node should be CDS option volatility quotes as described in Section 10.15. The Interpolation node supports Linear, Cubic and LogLinear interpolation. The Extrapolation node supports either None for no extrapolation or Flat for flat extrapolation in the volatility.

The optional boolean parameter EnforceMontoneVariance should be set to true to raise an exception if the curve implied variance is not montone increasing with time and should be set to false if you want to suppress such an exception. The default value for EnforceMontoneVariance is true.

```
<CDSVolatility>
  <CurveId>..<CurveId>
  <CurveDescription>...</CurveDescription>
  <Curve>
    <Quotes>
      <Quote>quote_id_1</Quote>
      <Quote>quote_id_2</Quote>
      ...
    </Quotes>
    <Interpolation>...</Interpolation>
    <Extrapolation>...</Extrapolation>
    <EnforceMontoneVariance></EnforceMontoneVariance>
  </Curve>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
</CDSVolatility>
```

Listing 82: CDS volatility curve configuration

For backwards compatibility, the volatility curve configuration can also be given using a single `Expiries` node as shown in Listing 83. Note that this configuration is deprecated and the configuration in 82 is preferred. The `Expiries` node should take a comma separated list of tenors and or expiry dates e.g. `<Expiries>1M,3M,6M</Expiries>`. The list of expiries are extracted and a set of quotes are created of the form `INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[EXPIRY]` or `INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[TERM]/[EXPIRY]`. There is one quote for each expiry in the list where the `[EXPIRY]` field is understood to be replaced with the expiry string extracted from the list.

The `[NAME]` field, and possible `[TERM]` field, in the quote depends on the values of the optional string node `QuoteName` and the optional boolean node `ParseTerm`. The following logic is used to populate the `[NAME]` field and possibly the `[TERM]` field:

- if `QuoteName` is provided, it is used directly to populate the `[NAME]` field. In other words, the quotes will be of the first form i.e. `INDEX_CDS_OPTION/RATE_LNVOL/QuoteName/[EXPIRY]` and the `[TERM]` field is not added to the quote string. Note that there is nothing stopping the user from adding the term directly to the `QuoteName` i.e. using a `QuoteName` of `SubQuoteName/Term`. In this way, the quotes will be of the second form i.e. `INDEX_CDS_OPTION/RATE_LNVOL/SubQuoteName/Term/[EXPIRY]` and `[NAME]` has been populated with `SubQuoteName` and `[TERM]` has been populated with `Term`. Obviously, for the quote to be parsed properly, `Term` needs to be a valid tenor string.

- if `QuoteName` is not provided and `ParseTerm` is provided and is `true`, an attempt is made to parse a `SubCurveId` and `Term` from the `CurveId` node. In particular, the parsing routine expects the `CurveId` to be of the form `SubCurveId-Term`. If this is the case, the `[NAME]` field is populated with `SubCurveId/Term`. The reason for this logic is to easily allow different volatility structures for different terms of the same index CDS series or the same CDS reference entity.

  As a concrete example, we could have two volatility structures with `CurveIds` equal to `CDXHYS34V1-5Y` and `CDXHYS34V1-10Y`. Assume that `QuoteName` is not provided and `ParseTerm` is provided and is `true` in both configurations. For `CDXHYS34V1-5Y`, quotes will be of the form `INDEX_CDS_OPTION/RATE_LNVOL/CDXHYS34V1/5Y/[EXPIRY]` and for

CDXHYS34V1-10Y, quotes will be of the form
INDEX_CDS_OPTION/RATE_LNVOL/CDXHYS34V1/10Y/[EXPIRY].

If the attempt to parse a SubCurveId and Term from the CurveId node fails, the [NAME] field is simply populated with the CurveId node value.

- if QuoteName is not provided and ParseTerm is not provided or is provided and is false, the [NAME] field is simply populated with the CurveId node value.

If only one expiry is provided in the list, there is only one quote and a constant volatility structure is configured as in Listing 81. If more than one expiry is provided, a curve is configured as in 82. The interpolation is Linear, the extrapolation is Flat and EnforceMontoneVariance is true.

```
<CDSVolatility>
  <CurveId>..<CurveId>
  <CurveDescription>...</CurveDescription>
  <Expiries>...</Expiries>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <QuoteName>...</QuoteName>
  <ParseTerm>...</ParseTerm>
</CDSVolatility>
```

Listing 83: Legacy deprecated CDS volatility curve configuration

Thirdly, the volatility surface configuration layout is given in Listing 84. The nodes have the following meanings and supported values:

- Strikes: comma separated list of strikes. The strikes may be in terms of spread or price. However, it is important to ensure that the trade XML for a CDS option or index CDS option provides the strike in the same way. In other words, if the strike is in terms of spread on the trade XML, the strike must be in terms of spread in the CDS volatility configuration here. Similarly for strikes in terms of price. A single wildcard character, *, can be used here also to indicate that all strikes found in the market data for this CDS volatility configuration should be used when building the CDS volatility surface.

- Expiries: comma separated list of expiry tenors and or expiry dates. A single wildcard character, *, can be used here also to indicate that all expiries found in the market data for this CDS volatility configuration should be used when building the CDS volatility surface.

- TimeInterpolation: interpolation in the option expiry direction. If either Strikes or Expiries are configured with a wildcard character, Linear is used. If both Strikes and Expiries are configured explicitly, Linear or Cubic is allowed here but the value must agree with the value for StrikeInterpolation.

- StrikeInterpolation: interpolation in the strike direction. If either Strikes or Expiries are configured with a wildcard character, Linear is used. If both Strikes and Expiries are configured explicitly, Linear or Cubic is allowed here but the value must agree with the value for TimeInterpolation.

- Extrapolation: boolean value. If true, extrapolation is allowed. If false, extrapolation is not allowed.

- `TimeExtrapolation`: extrapolation in the option expiry direction. If both `Strikes` and `Expiries` are configured explicitly, the extrapolation in the time direction is flat in volatility regardless of the setting here. If either `Strikes` or `Expiries` are configured with a wildcard character, `Linear`, `UseInterpolator`, `Flat` or `None` are allowed. If `Linear` or `UseInterpolator` is specified, the extrapolation is linear. If `Flat` is specified, the extrapolation is flat. If `None` is specified, it is ignored and the extrapolation is flat since extrapolation in the time direction cannot be turned off in isolation i.e. extrapolation can only be turned off for the surface as a whole using the `Extrapolation` flag.
- `StrikeExtrapolation`: extrapolation in the strike direction. The allowable values are `Linear`, `UseInterpolator`, `Flat` or `None`. If `Linear` or `UseInterpolator` is specified, the extrapolation uses the strike interpolation setting for extrapolation i.e. linear or cubic in this case. If `Flat` is specified, the extrapolation is flat. If `None` is specified, it is ignored and the extrapolation is flat since extrapolation in the strike direction cannot be turned off in isolation i.e. extrapolation can only be turned off for the surface as a whole using the `Extrapolation` flag.
- `DayCounter`: allowable value is any valid day count fraction. As stated above, this node is optional and defautls to `A365F`.
- `Calendar`: allowable value is any valid calendar. As stated above, this node is optional and defaults to `NullCalendar`.
- `StrikeType`: allowable value is either `Price` or `Spread`. This flag denotes if the strikes are in terms of spread or price. Currently, this is merely informational and as outlined in the `Strikes` section above, it is the responsibility of the user to ensure that the strike type in trades aligns with the configured strike type in the CDS volatility surfaces.
- `QuoteName`: this node is optional and the allowable value is any string. This value can be used in determining the name and term that appears in the quote strings that are searched for in the market data to feed into the CDS volatility surface construction. How it is used has been outlined above when describing the derprecated CDS volatility curve configuration.
- `StrikeFactor`: this node is optional and the allowable value is any postive real number. It defaults to 1. The strikes configured and found in the market data quote strings may not be in absolute terms. For example, a quote string such as `INDEX_CDS_OPTION/RATE_LNVOL/CDXIGS33V1/5Y/1M/115` could be given to indicate an index CDS option volatility quote for CDX IG Series 33 Version 1, with underlying index term 5Y expiring in 1M with a strike spread of 115 bps. The strike here is in bps and needs to be divided by 10,000 before being used in the ORE volatility objects. The `StrikeFactor` would be set to `10000` here.
- `ParseTerm`: this node is an optional boolean and defaults to `false`. This value can be used in determining the name and term that appears in the quote strings that are searched for in the market data to feed into the CDS volatility surface construction. How it is used has been outlined above when describing the derprecated CDS volatility curve configuration.

When the CDS volatility surface is configured as in Listing 84, the market is searched for quote strings of the form
`INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[EXPIRY]/[STRIKE]` or
`INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[TERM]/[EXPIRY]/[STRIKE]`. The population of the `[NAME]` field, and possibly the `[TERM]` field, and how they depend on the `QuoteName` and `ParseTerm` nodes has been discussed at length above when describing the derprecated CDS volatility curve configuration. When both the `Strikes` and `Expiries` are configured explicitly, it is clear that the `[EXPIRY]` field is populated from the list of expiries in turn and the `[STRIKE]` field is populated from the list of strikes in turn. If there are m expiries in the `Expiries` list and n strikes in the `Strikes` list, there will be $m \times n$ quotes created and searched for in the market data. If `Expiries` are configured via the wildcard character, `*`, all quotes in the market data matching the pattern `INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/*/[STRIKE]` will be used if `[TERM]` has not been populated and all quotes in the market data matching the pattern `INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[TERM]/*/[STRIKE]` will be used if `[TERM]` has been populated. Similarly for `Strikes` configured via the wildcard character, `*`.

```xml
<CDSVolatility>
  <CurveId/>
  <CurveDescription/>
  <StrikeSurface>
    <Strikes>...</Strikes>
    <Expiries>...</Expiries>
    <TimeInterpolation>...</TimeInterpolation>
    <StrikeInterpolation>...</StrikeInterpolation>
    <Extrapolation>...</Extrapolation>
    <TimeExtrapolation>...</TimeExtrapolation>
    <StrikeExtrapolation>...</StrikeExtrapolation>
  </StrikeSurface>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <StrikeType>...</StrikeType>
  <QuoteName>...</QuoteName>
  <StrikeFactor>..</StrikeFactor>
  <ParseTerm>...</ParseTerm>
</CDSVolatility>
```

Listing 84: CDS volatility surface configuration

7.8.13 Base Correlations

Listing 85 shows the configuration of a Base Correlation curve.

```xml
  <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 85: Base Correlation Configuration

The meaning of each of the elements in Listing 85 is given below.

- CurveId: Unique identifier of the base correlation structure
- CurveDescription: A description of the base correlation structure, may be left blank.
- Terms: Comma-separated list of tenors, sorted in increasing order, possibly a single term to represent a flat term structure in time-direction
- DetachmentPoints: Comma-separated list of equity tranche detachment points, sorted in increasing order
  Allowable values: Any positive number less than one
- SettlementDays: The floating term structure's settlement days argument used in the reference date calculation
- DayCounter: The term structure's day counter used in date to time conversions
- Calendar: The term structure's calendar used in tenor to date conversions
- BusinessDayConvention: The term structure's business day convention used in tenor to date conversion
- Extrapolate: Boolean to indicate whether the correlation curve shall be extrapolated or not

7.8.14 FXSpots

Listing 86 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 86: FXSpot Configuration

7.8.15 Securities

Listing 87 shows the configuration of the Securities. Each Security name is associated with

- an optional SpreadQuote
- an optional RecoveryRateQuote. Usually a pricer will fall back on the recovery rate associated to the credit curve involved in the pricing if no security specific recovery rate is given. If no credit curve is given either, a zero recovery rate will be assumed.

If no configuration is given for a security, in general a pricer will assume as zero spread and recovery rate. Notice that in this case the spread and recovery will not be simulated and therefore also be excluded from the sensitvitiy and stress analysis.

```
<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 87: Security Configuration

7.8.16 Correlations

Listing 88 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. OptionTenors can be a comma separaed list of periods, 1Y,2Y etc, or a * to indicate a wildcard. If a wildcard is provided, all relevant market data quotes are used.

```
  <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 88: Correlation Configuration

7.8.17 Commodity Curves

Commodity Curves are setup as price curves in ORE, meaning that they return a price for a given time t rather than a rate or discount factor, these curves are common in commodities and can be populated with futures prices directly.

Listing 89 shows the configuration of Commodity curves built from futures prices, in this example WTI Oil prices, note there is no spot price in this configuration, rather we have a set of futures prices only.

```
<CommodityCurve>
  <CurveId>WTI_USD</CurveId>
  <CurveDescription>WTI USD price curve</CurveDescription>
  <Currency>USD</Currency>
  <Quotes>
    <Quote>COMMODITY_FWD/PRICE/WTI/USD/2016-06-30</Quote>
    <Quote>COMMODITY_FWD/PRICE/WTI/USD/2016-07-31</Quote>
    ...
  </Quotes>
  <DayCounter>A365</DayCounter>
  <InterpolationMethod>Linear</InterpolationMethod>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 89: Commodity Curve Configuration for WTI Oil

Listing 90 shows the configuration for a Precious Metal curve using FX style spot and forward point quotes, note that SpotQuote is used in this case. The different interpretation of the forward quotes is controlled by the conventions.

```
<CommodityCurve>
  <CurveId>XAU</CurveId>
  <CurveDescription>Gold USD price curve</CurveDescription>
  <Currency>USD</Currency>
  <SpotQuote>COMMODITY/PRICE/XAU/USD</SpotQuote>
  <Quotes>
    <Quote>COMMODITY_FWD/PRICE/XAU/USD/ON</Quote>
    <Quote>COMMODITY_FWD/PRICE/XAU/USD/TN</Quote>
    <Quote>COMMODITY_FWD/PRICE/XAU/USD/SN</Quote>
    <Quote>COMMODITY_FWD/PRICE/XAU/USD/1W</Quote>
    ...
  </Quotes>
  <DayCounter>A365</DayCounter>
  <InterpolationMethod>Linear</InterpolationMethod>
  <Conventions>XAU</Conventions>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 90: Commodity Curve Configuration for Gold in USD

The meaning of each of the top level elements 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 curve.
- CurveDescription: A description of the curve. This field may be left blank.
- Currency: The commodity curve currency.

- SpotQuote [Optional]: The spot price quote, if omitted then the spot value will be interpolated.
- Quotes: forward price quotes. These can be a futures price or forward points.
- DayCounter: The day count basis used internally by the curve to calculate the time between dates.
- InterpolationMethod [Optional]: The variable on which the interpolation is performed. The allowable values are Linear, LogLinear, Cubic, Hermite, LinearFlat, LogLinearFlat, CubicFlat, HermiteFlat, BackwardFlat. This is different to yield curves above in that Flat versions of the standard methods are defined, with each of these if there is no Spot price than any extrapolation between T_0 and the first future price will be flat (i.e. the first future price will be copied back "Flat" to T_0). If the element is omitted or left blank, then it defaults to *Linear*.
- Conventions [Optional]: The conventions to use, if omited it is assumed that these quotes are Outright prices.
- 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*.

Alternatively commodity curves can be set up as a commodity curve times the ratio of two yield curves as shown in listing 91. This can be used to setup commodity curves in different currencies, for example Gold in EUR (XAUEUR) can be built from a Gold in USD curve and then the ratio of the EUR and USD discount factors at each pillar. This is akin to crossing FX forward points to get FX forward prices for any pair.

```
<CommodityCurve>
  <CurveId>XAUEUR</CurveId>
  <CurveDescription>Gold EUR price curve</CurveDescription>
  <Currency>EUR</Currency>
  <BasePriceCurve>XAU</BasePriceCurve>
  <BaseYieldCurve>USD-FedFunds</BaseYieldCurve>
  <YieldCurve>EUR-IN-USD</YieldCurve>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 91: Commodity Curve Configuration for Gold in EUR

Commodity curves may also be set up using a base future price curve and a set of future basis quotes to give an outright price curve. There are a number of options here depending on whether the base future and basis future are averaging or not averaging. Whether or not the base future and basis future is averaging is determined from the conventions supplied in the `BasePriceConventions` and `BasisConventions` fields.

- The base future is not averaging and the basis future is not averaging. The commodity curve that is built gives the outright price of the non-averaging future. An example of this is the CME Henry Hub future contract, symbol NG, and the various locational gas basis future contracts, e.g. ICE Waha Basis Future, symbol WAH. Listing 92 demonstrates an example set-up for this curve. The curve that is built will give the ICE Waha outright price on the basis contract's expiry dates.

- The base future is not averaging and the basis future is averaging. The commodity curve that is built gives the outright price of the averaging future. In this case, if the `AverageBase` field is `true` the base price will be averaged from and excluding one basis future expiry to and including the next basis future expiry. An example of this is the CME Light Sweet Crude Oil future contract, symbol CL, and the various locational oil basis future contracts, e.g. CME WTI Midland (Argus) Future, symbol FF. Listing 93 demonstrates an example set-up for this curve. The curve that is built will give the outright average price of WTI Midland (Argus) over the calendar month. If the `AverageBase` field is `false`, the base price is not averaged and the basis is added to the base price to give a curve that can be queried on an expiry date for an average price. An example of this is a curve built for the average of the daily prices published by Gas Daily using the ICE futures that reference the difference between the Inside FERC price and the average Gas Daily price.
- The base future is averaging and the basis future is averaging. The commodity curve that is built gives the outright price of the averaging future. The set-up is identical to that outlined in Listings 92 and 93.
- The base future is averaging and the basis future is not averaging. This combination is not currently supported.

```
<CommodityCurve>
  <CurveId>ICE:WAH</CurveId>
  <Currency>USD</Currency>
  <BasisConfiguration>
    <BasePriceCurve>NYMEX:NG</BasePriceCurve>
    <BasePriceConventions>NYMEX:NG</BasePriceConventions>
    <BasisQuotes>
      <Quote>COMMODITY_FWD/PRICE/ICE:WAH/*</Quote>
    </BasisQuotes>
    <BasisConventions>ICE:WAH</BasisConventions>
    <DayCounter>A365</DayCounter>
    <InterpolationMethod>LinearFlat</InterpolationMethod>
    <AddBasis>true</AddBasis>
  </BasisConfiguration>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 92: Commodity curve configuration for ICE Waha

```
<CommodityCurve>
  <CurveId>NYMEX:FF</CurveId>
  <Currency>USD</Currency>
  <BasisConfiguration>
    <BasePriceCurve>NYMEX:CL</BasePriceCurve>
    <BasePriceConventions>NYMEX:CL</BasePriceConventions>
    <BasisQuotes>
      <Quote>COMMODITY_FWD/PRICE/NYMEX:FF/*</Quote>
    </BasisQuotes>
    <BasisConventions>NYMEX:FF</BasisConventions>
    <DayCounter>A365</DayCounter>
    <InterpolationMethod>LinearFlat</InterpolationMethod>
    <AddBasis>true</AddBasis>
    <AverageBase>true</AverageBase>
  </BasisConfiguration>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 93: Commodity curve configuration for WTI Midland (Argus)

The meaning of the fields in the BasisConfiguration node in Listings 92 and 93 are as follows:

- BasePriceCurve: The identifier for the base future commodity price curve.
- BasePriceConventions: The identifier for the base future contract conventions.
- BasisQuotes: The set of basis quotes to look for in the market. Note that this can be a single wildcard string as shown in the Listings or a list of explicit COMMODITY_FWD PRICE quote strings.
- BasisConventions: The identifier for the basis future contract conventions.
- DayCounter: Has the meaning given previously in this section.
- InterpolationMethod [Optional]: Has the meaning given previously in this section.
- AddBasis [Optional]: This is a boolean flag where true, the default value, indicates that the basis value should be added to the base price to give the outright price and false indicates that the basis value should be subtracted from the base price to give the outright price.
- MonthOffset [Optional]: This is an optional non-negative integer value. In general, the basis contract period and the base contract period are equal, i.e. the value of the March basis contract for ICE Waha will be added to the value of thr March contract for CME NG. If for contracts with a monthly cycle or greater, the base contract month is n months prior to the basis contract month, the MonthOffset should be set to n. The default value if omitted is 0.

A commodity curve may also be set up as a piecewise price curve involving sets of quotes e.g. direct future price quotes, future price quotes that are the average of other future prices over a period, future price quotes that are the average of spot price over a period etc. This is particularly useful for cases where there are future contracts with different cycles. For example, in the power markets, there are daily future contracts at the short end and monthly future contracts that average the daily prices over the month at the long end. An example of such a set-up is shown in Listing 94.

```
<CommodityCurve>
  <CurveId>ICE:PDQ</CurveId>
  <Currency>USD</Currency>
  <PriceSegments>
    <PriceSegment>
      <Type>Future</Type>
      <Priority>1</Priority>
      <Conventions>ICE:PDQ</Conventions>
      <Quotes>
        <Quote>COMMODITY_FWD/PRICE/ICE:PDQ/*</Quote>
      </Quotes>
    </PriceSegment>
    <PriceSegment>
      <Type>AveragingFuture</Type>
      <Priority>2</Priority>
      <Conventions>ICE:PMI</Conventions>
      <Quotes>
        <Quote>COMMODITY_FWD/PRICE/ICE:PMI/*</Quote>
      </Quotes>
    </PriceSegment>
  </PriceSegments>
  <DayCounter>A365</DayCounter>
  <InterpolationMethod>LinearFlat</InterpolationMethod>
  <Extrapolation>true</Extrapolation>
  <BootstrapConfig>...</BootstrapConfig>
</CommodityCurve>
```

Listing 94: Commodity curve configuration for PJM Real Time Peak

The BootstrapConfig node is described in Section 7.8.19. The remaining nodes in Listing 94 have been described already in this section. The meaning of each of the fields in the PriceSegment node in Listing 94 is as follows:

- Type: The type of the future contract for which quotes are being provided in the current PriceSegment. The allowable values are:
  - Future: This indicates that the quote is a straight future contract price quote.
  - AveragingFuture: This indicates that the quote is for a future contract price that is the average of other future contract prices over a given period. The averaging period for each quote and other conventions are given in the associated conventions determined by the Conventions node.
  - AveragingSpot: This indicates that the quote is for a future contract price that is the average of spot prices over a given period. The averaging period for each quote and other conventions are given in the associated conventions determined by the Conventions node.
- Priority [Optional]: An optional non-negative integer giving the priority of the current PriceSegment relative to the other PriceSegments when there are quotes for contracts with the same expiry dates in those segments. Values closer to zero indicate higher priority i.e. quotes in this segment are given priority in the event of clashes. If omitted, the PriceSegments are currently read in the order that they are provided in the XML so that quotes in segments appearing earlier in the XML will be given preference in the case of clashes.
- Conventions: The identifier for the future contract conventions associated with the quotes in the PriceSegment. Details on these conventions are given in Section 7.11.20.
- Quotes: The set of future price quotes to look for in the market. Note that this can be a single wildcard string as shown in the Listing 94 or a list of explicit COMMODITY_FWD PRICE quote strings.

7.8.18 Commodity Volatilities

The following types of commodity volatility structures are supported in ORE:

- A constant volatility structure giving the same single volatility for all expiry times and strikes.
- A one-dimensional expiry dependent volatility structure i.e. the volatility returned is dependent on the time to option expiry but does not change with option strike.
- A two-dimensional volatility structure with a dependence on both expiry and strike. There is support for absolute strikes, delta strikes and moneyness strikes.
- An average price option (APO) volatility surface. In particular, this structure returns the volatility of an average price that can then be used directly in the Black 76 formula to give the value of the APO.

Listing 95 outlines the configuration for a constant volatility structure.

```
<CommodityVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>...</Currency>
  <Constant>
    <Quote>...</Quote>
  </Constant>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
</CommodityVolatility>
```

Listing 95: Constant commodity volatility configuration

The meaning of each of the elements is as follows:

- `CurveId`: Unique identifier for the curve.
- `CurveDescription`: A description of the curve. This field may be left blank.
- `Currency`: The commodity curve currency.
- `Quote`: The single quote giving the constant volatility.
- `DayCounter` [Optional]: The day count basis used internally by the curve to calculate the time between dates. If omitted it defaults to `A365`.
- `Calendar` [Optional]: The calendar used internally by the volatility structure to amend dates generated from option tenors i.e. if a volatility is requested from the surface using an expiry tenor. If omitted it defaults to `NullCalendar` meaning there is no adjustment to the expiry on applying the option tenor.

Listing 96 outlines the configuration for the one-dimensional expiry dependent volatility curve.

```
<CommodityVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>...</Currency>
  <Curve>
    <QuoteType>...</QuoteType>
    <VolatilityType>...</VolatilityType>
    <ExerciseType>...</ExerciseType>
    <Quotes>
      <Quote>...</Quote>
      <Quote>...</Quote>
      ...
    </Quotes>
    <Interpolation>...</Interpolation>
    <Extrapolation>...</Extrapolation>
    <EnforceMontoneVariance>...</EnforceMontoneVariance>
  </Curve>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <FutureConventions>...</FutureConventions>
  <OptionExpiryRollDays>...</OptionExpiryRollDays>
</CommodityVolatility>
```

Listing 96: Commodity volatility curve configuration

The meaning of each of the elements is given below. Elements that were defined for the previous configuration and are common to all of the configurations are not repeated.

- `QuoteType` [Optional]: The allowable values in general for `QuoteType` are `ImpliedVolatility` and `Premium`. Currently, only `ImpliedVolatility` is supported for commodity volatility curves. This is the default for `QuoteType` so this node may be omitted.
- `VolatilityType` [Optional]: The allowable values in general for `VolatilityType` are `Lognormal`, `ShiftedLognormal` and `Normal`. Currently, only `Lognormal` is supported for commodity volatility curves. This is the default for `VolatilityType` so this node may be omitted.
- `ExerciseType` [Optional]: This node is described below in the context of surfaces. For commodity volatility curves, it is ignored and should be omitted.
- `Quotes`: A list of commodity option volatility quotes with different expiries to use in the commodity curve building. The commodity option volatility quotes are explained in Section 10.28. As indicated above, any quote string used here much start with `COMMODITY_OPTION/RATE_LNVOL`. A single regular expression `Quote` is also supported here in place of a list of explicit `Quote` strings. Note that if a list of explicit `Quote` strings are provided, it is an error to have a duplicated option expiry date. If a regular expression is used, the first quote found is used and subsequent qutoes with the same expiry are discarded with a warning issued.
- `Interpolation`: The interpolation to use to give volatilities between option expiry times. The allowable values are `Linear`, `Cubic` and `LogLinear`. Note that the interpolation here is on the variance.
- `Extrapolation`: The extrapolation to use to give volatilities after the last expiry date in the variance curve. The allowable values are `None`, `UseInterpolator`, `Linear` and `Flat`. However, all options except `None` yield the same extrapolation i.e. flat extrapolation in the volatility. `None` disables extrapolation so that an exception is raised if the curve is queried after the last expiry for a volatility.

Note that as the curve is parameterised in variance, interpolation is used to interpolate between time zero where the variance is zero and the first expiry time.

- `EnforceMontoneVariance` [Optional]: Boolean parameter that should be set to `true` to raise an exception if the implied variance curve is not montone increasing with time. It should be set to `false` to suppress such an exception. The default value if omitted is `true`.

- `FutureConventions` [Optional]: Depending on the quotes that are provided in the `Quotes` section, a `CommodityFuture` convention may be needed in order to derive an option expiry date from the *Expiry* portion of the commodity option quote. In particular, as outlined in Section 10.28, the *Expiry* portion of a commodity option quote allows for continuation expiries of the form `cN`. The `N` is a positive integer meaning the `N`-th next expiry after the valuation date on which we are building the commodity volatility curve. When a continuation expiry is used in a quote, the `FutureConventions` is needed and gives the ID of the conventions associated with the commodity for which we are trying to build the volatility curve. These conventions are used to determine the explicit expiry date for the given option quote from the continuation expiry.

- `OptionExpiryRollDays` [Optional]: The `OptionExpiryRollDays` can be any non-negative integer and may be needed when deriving an option expiry date from the *Expiry* portion of the commodity option quote. If the *Expiry* portion of the commodity option quote is a continuation expiry, an explicit expiry date is deduced as explained in the previous bullet point. Additionally, in some cases, the option quotes for the next option expiry may stop a number of business days before that option expiry and the `cN` expiry in this period begins referring to the `N+1`-th next option expiry. As an example, assume d_v is the valuation date and $e_1 = d_v$ is the next option expiry date. If `OptionExpiryRollDays` is `0` then a commodity option quote with an *Expiry* portion equal to `c1` on d_v indicates that the option quote is for an option with expiry date equal to e_1. However, if `OptionExpiryRollDays` is `1`, a commodity option quote with an *Expiry* portion equal to `c1` on d_v indicates that the option quote is for an option with expiry date equal to e_2 where e_2 is the next option expiry date after e_1. In other words, with `OptionExpiryRollDays` set to `1` the option quotes for expiry date e_1 stopped on the business day before e_1. If omitted, `OptionExpiryRollDays` defaults to `0`.

Listing 97 outlines the configuration for the two-dimensional expiry and absolute strike commodity option surface.

```
<CommodityVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>...</Currency>
  <StrikeSurface>
    <QuoteType>...</QuoteType>
    <VolatilityType>...</VolatilityType>
    <ExerciseType>...</ExerciseType>
    <Strikes>...</Strikes>
    <Expiries>...</Expiries>
    <TimeInterpolation>...</TimeInterpolation>
    <StrikeInterpolation>...</StrikeInterpolation>
    <Extrapolation>...</Extrapolation>
    <TimeExtrapolation>...</TimeExtrapolation>
    <StrikeExtrapolation>...</StrikeExtrapolation>
  </StrikeSurface>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <FutureConventions>..</FutureConventions>
  <OptionExpiryRollDays>...</OptionExpiryRollDays>
  <PriceCurveId>...</PriceCurveId>
  <YieldCurveId>...</YieldCurveId>
  <OneDimSolverConfig>
    <MaxEvaluations>100</MaxEvaluations>
    <InitialGuess>0.35</InitialGuess>
    <Accuracy>0.0001</Accuracy>
    <MinMax>
      <Min>0.0001</Min>
      <Max>2.0</Max>
    </MinMax>
  </OneDimSolverConfig>
  <PreferOutOfTheMoney>...</PreferOutOfTheMoney>
  <QuoteSuffix>...</QuoteSuffix>
</CommodityVolatility>
```

Listing 97: Expiry and absolute strike commodity option surface configuration

The meaning of each of the elements is given below. Again, nodes explained in the previous configuration are not repeated here.

- `QuoteType` [Optional]: As above, the allowable values for `QuoteType` are `ImpliedVolatility` and `Premium`. If omitted, the default is `ImpliedVolatility`. If the `QuoteType` is `Premium`, a volatility surface will be stripped from option premium quotes. Note that `Premium` is only allowed if one or both of `Strikes` or `Expiries` below is set to the single wildcard value `*`. In other words, if we explicitly specify all of the strikes and expiries, we can only build a volatility surface directly and the `QuoteType` must be `ImpliedVolatility`.

- `VolatilityType` [Optional]: As above, the allowable values for `VolatilityType` are `Lognormal`, `ShiftedLognormal` and `Normal`. This is only needed if `QuoteType` is `ImpliedVolatility`. Currently, only `Lognormal` is supported for commodity volatility surfaces. This is the default for `VolatilityType` so this node may be omitted.

- `ExerciseType` [Optional]: The allowable values for `ExerciseType` are `European` and `American`. This is only needed if `QuoteType` is `Premium` and indicates if the option premium quotes are American or European exercise. If omitted the default is `European`.

- `Strikes`: This can be a single wildcard value `*` or a comma separated list of explicit strike prices. We explain below how these strikes are combined with the other parameters in the configuration to give a list of commodity option quotes to search for in the market data.

- `Expiries`: This can be a single wildcard value `*` or a comma separated list of expiry strings. We explain below how these expiries are combined with the other parameters in the configuration to give a list of commodity option quotes to search for in the market data. Note that as outlined in Section 10.28, the *Expiry* portion of the commodity option quote may be an explicit expiry date, an expiry tenor or a continuation expiry of the form `cN` explained in the volatility curve section above.

- `TimeInterpolation`: Indicates the interpolation in the time direction. There are quite a number of restrictions here. If either `Strikes` or `Expiries` use the single wildcard value `*`, the interpolation in both the time and strike direction is linear regardless of the value passed here in the `TimeInterpolation` node. If neither `Strikes` nor `Expiries` use the single wildcard value `*`, `TimeInterpolation` may be set to `Linear` or `Cubic` but `StrikeInterpolation` must have the same value. If it does not, then `Linear` is used for both. In other words, if neither `Strikes` nor `Expiries` use the single wildcard value `*`, we can configure bilinear or bicubic interpolation. Again, in all cases, the interpolation is carried out on the variance.
- `StrikeInterpolation`: Indicates the interpolation in the strike direction. The requirements are exactly as outlined for the `TimeInterpolation` node.
- `Extrapolation`: A boolean value indicating if extrapolation is allowed.
- `TimeExtrapolation`: Indicates the extrapolation in the time direction. The allowable values are `None`, `UseInterpolator`, `Linear` and `Flat`. If neither `Strikes` nor `Expiries` use the single wildcard value `*`, the extrapolation in the time direction is flat regardless of the value passed here. If either `Strikes` or `Expiries` use the single wildcard value `*`, both `Flat` and `None` give flat extrapolation in the time direction whereas `UseInterpolator` and `Linear` indicate that the configured interpolation (linear or cubic) should be continued in the time direction in order to extrapolate. `Linear` is only allowable here for backward compatibility and `UseInterpolator` should be preferred for clarity.
- `StrikeExtrapolation`: Indicates the extrapolation in the strike direction. The allowable values are `None`, `UseInterpolator`, `Linear` and `Flat`. Both `Flat` and `None` give flat extrapolation in the strike direction. `UseInterpolator` and `Linear` indicate that the configured interpolation (linear or cubic) should be continued in the strike direction in order to extrapolate. `Linear` is only allowable here for backward compatibility and `UseInterpolator` should be preferred for clarity.
- `PriceCurveId` [Optional]: The ID of a price curve for the commodity of the form `Commodity/{CCY}/{NAME}`. This is needed if the `QuoteType` is `Premium`. It is also needed when the `QuoteType` is `ImpliedVolatility` if either `Strikes` or `Expiries` use the single wildcard value `*` and both call and put quotes are found in the market for the same expiry and strike pair. In this case, it is needed to determine which quotes to use based on the value of the `PreferOutOfTheMoney` node.
- `YieldCurveId` [Optional]: The ID of a yield curve in the currency of the commodity of the form `Yield/{CCY}/{NAME}`. This is needed if the `QuoteType` is `Premium` in the stripping of the volatilities from premia.
- `OneDimSolverConfig` [Optional]: This is used if the `QuoteType` is `Premium`. It provides the options for the root search in the stripping of the volatilities from premia. If omitted, the default set of options shown in Listing 97 are used. The

`MinMax` node can be replaced with a single `Step` node that accepts a double giving the step size to use in the root search.

- `PreferOutOfTheMoney` [Optional]: A node accepting a boolean value. If set to `true`, quotes for out of the money options are preferred where a call and a put quote are found for the same expiry strike pair. If set to `false`, quotes for in the money options are preferred where a call and a put quote are found for the same expiry strike pair. If omitted, `true` is assumed.

- `PreferOutOfTheMoney` [Optional]: A node accepting a boolean value. If set to `true`, quotes for out of the money options are preferred where a call and a put quote are found for the same expiry strike pair. If set to `false`, quotes for in the money options are preferred where a call and a put quote are found for the same expiry strike pair. If omitted, `true` is assumed.

- `QuoteSuffix` [Optional]: The allowable values are `C` and `P` indicating `Call` and `Put` respectively. If given, they are used in the construction of the commodity option quote strings as explained below. They are useful in cases where the market data contains both call and put volatility quotes for the same expiry strike pair and you want to use only the calls (set `QuoteSuffix` to `C`) or the puts (set `QuoteSuffix` to `P`).

As mentioned above, a number of parameters from the two-dimensional expiry and absolute strike configuration are used in constructing the commodity option quote strings that are looked up in the market data. There are two cases:

1. Both the `Strikes` and `Expiries` node provide a comma separated list of values. As mentioned above, we can only use a `QuoteType` of `ImpliedVolatility` in this case where we have explicit expiries and strikes and the `VolatilityType` must be `Lognormal`. For example, assume the `Expiries` node has the set of values `e_1,e_2,...,e_N` and that the `Strikes` node has the set of values `s_1,s_2,...,s_M`. For each of the $N \times M$ expiry strike pairs (e_n, s_m), a quote of the form `COMMODITY_OPTION/RATE_LNVOL/{N}/{C}/e_n/s_m[/{S}]` is created to be looked up in the market data. `{N}` is the value in the `CurveId` node, `{C}` is the value in the `Currency` node and `{S}` is the value in the `QuoteSuffix` node if given. This explicit grid of volatility quotes must be present in the market for the commodity volatility surface to be constructed.

2. One or both of the `Strikes` and `Expiries` node use a single wildcard value `*`. As mentioned above, the `QuoteType` can be `ImpliedVolatility` or `Premium` in this case. As above, assume the `Expiries` node has the set of values `e_1,e_2,...,e_N` and that the `Strikes` node has the set of values `s_1,s_2,...,s_M`. The additional constraint here is that $N = 1$ and `e_1` is `*` or that $M = 1$ and `s_1` is `*`, or both. For each of the $N \times M$ expiry strike pairs (e_n, s_m), a quote of the form `COMMODITY_OPTION/{T}/{N}/{C}/e_n/s_m[/{S}]` is created to be looked up in the market data. `{T}` is `PRICE` when `QuoteType` is `Premium` and is `RATE_LNVOL` when `QuoteType` is `ImpliedVolatility`, `{N}` is the value in the `CurveId` node, `{C}` is the value in the `Currency` node and `{S}` is the value in the `QuoteSuffix` node if given. Any quote in the market with a name matching any of the quote strings formed in this way are then included in the commodity volatility curve building. Note that the `QuoteSuffix` has no effect in

this case and should be omitted i.e. it is only used in the case of an explicit grid of quotes above.

Listing 98 outlines the configuration for the two-dimensional expiry and moneyness strike commodity option surface. This is similar to the absolute strike surface configuration above but currently only supports a `QuoteType` of `ImpliedVolatility` i.e. `QuoteType` of `Premium` is not supported. Also, the `VolatilityType` must be `Lognormal`. Both forward and spot moneyness is supported.

```
<CommodityVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>...</Currency>
  <MoneynessSurface>
    <QuoteType>...</QuoteType>
    <VolatilityType>...</VolatilityType>
    <ExerciseType>...</ExerciseType>
    <MoneynessType>...</MoneynessType>
    <MoneynessLevels>...</MoneynessLevels>
    <Expiries>...</Expiries>
    <TimeInterpolation>...</TimeInterpolation>
    <StrikeInterpolation>...</StrikeInterpolation>
    <Extrapolation>...</Extrapolation>
    <TimeExtrapolation>...</TimeExtrapolation>
    <StrikeExtrapolation>...</StrikeExtrapolation>
    <FuturePriceCorrection>...</FuturePriceCorrection>
  </MoneynessSurface>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <FutureConventions>..</FutureConventions>
  <OptionExpiryRollDays>...</OptionExpiryRollDays>
  <PriceCurveId>...</PriceCurveId>
  <YieldCurveId>...</YieldCurveId>
</CommodityVolatility>
```

Listing 98: Expiry and moneyness strike commodity option surface configuration

The meaning of each of the elements is given below. Again, nodes explained in the previous configuration are not repeated here.

- `MoneynessType`: The allowable values are `Spot` for spot moneyness and `Fwd` for forward moneyness.
- `MoneynessLevels`: This can be a single wildcard value `*` or a comma separated list of moneyness values. A moneyness value of 1 indicates a strike equal to spot or forward depending on the value given in the `MoneynessType` node.
- `TimeInterpolation`: Only `Linear` is currently supported here.
- `StrikeInterpolation`: Only `Linear` is currently supported here.
- `Extrapolation`: A boolean value indicating if extrapolation is allowed.
- `TimeExtrapolation`: Only `Flat` is currently supported here giving flat extrapolation of the volatility.
- `StrikeExtrapolation`: Indicates the extrapolation in the strike direction. The

allowable values are None, UseInterpolator, Linear and Flat. Both Flat and None give flat extrapolation in the strike direction. UseInterpolator and Linear indicate that the configured interpolation (linear) should be continued in the strike direction in order to extrapolate.

- FuturePriceCorrection [Optional]: This is a boolean flag that defaults to true. In most cases, for options on futures, the option expiry date is a short period before the future expiry. If there is an arbitrary interpolation applied to the future price curve, the future price on the option expiry date may not equal the associated future price. If FuturePriceCorrection is true, this is corrected i.e. the future price on option expiry is the associated future price for the future expiry date. Note that a valid FutureConventions is needed for the correction to be applied.
- PriceCurveId: This is required for both a spot and forward moneyness surface.
- YieldCurveId: This is required for a forward moneyness surface.

Note that, similar to the procedure outlined above for the absolute strike surface, quote strings of the form COMMODITY_OPTION/RATE_LNVOL/{N}/{C}/e_n/MNY/{T}/l_m are created from the moneyness configuration to be looked up in the market. Here, l_m are the moneyness levels for $m = 1, \ldots, M$ and {T} is the moneyness type i.e. either Spot or Fwd.

Listing 99 outlines the configuration for the two-dimensional expiry and delta strike commodity option surface. Similar to the moneyness strike surface configuration above, this currently only supports a QuoteType of ImpliedVolatility i.e. QuoteType of Premium is not supported. Also, the VolatilityType must be Lognormal. Various delta and ATM types are supported.

```
<CommodityVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <Currency>...</Currency>
  <DeltaSurface>
    <QuoteType>...</QuoteType>
    <VolatilityType>...</VolatilityType>
    <ExerciseType>...</ExerciseType>
    <DeltaType>...</DeltaType>
    <AtmType>...</AtmType>
    <AtmDeltaType>...</AtmDeltaType>
    <PutDeltas>...</PutDeltas>
    <CallDeltas>...</CallDeltas>
    <Expiries>...</Expiries>
    <TimeInterpolation>...</TimeInterpolation>
    <StrikeInterpolation>...</StrikeInterpolation>
    <Extrapolation>...</Extrapolation>
    <TimeExtrapolation>...</TimeExtrapolation>
    <StrikeExtrapolation>...</StrikeExtrapolation>
    <FuturePriceCorrection>...</FuturePriceCorrection>
  </DeltaSurface>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <FutureConventions>..</FutureConventions>
  <OptionExpiryRollDays>...</OptionExpiryRollDays>
  <PriceCurveId>...</PriceCurveId>
  <YieldCurveId>...</YieldCurveId>
</CommodityVolatility>
```

Listing 99: Expiry and delta strike commodity option surface configuration

The meaning of each of the elements is given below. Again, nodes explained in the previous configuration are not repeated here.

- `DeltaType`: The allowable supported values are `Spot` for spot delta `Fwd` for forward delta.
- `AtmType`: The allowable supported values are `AtmSpot`, `AtmFwd` and `AtmDeltaNeutral`.
- `AtmDeltaType` [Optional]: This is only needed if the `AtmType` is `AtmDeltaNeutral`.
- `PutDeltas`: A comma separated list of one or more put deltas to use in the volatility surface. Note that the put deltas should be given without a sign e.g. `<PutDeltas>0.10,0.20,0.30,0.40</PutDeltas>` would be an example.
- `PutDeltas`: A comma separated list of one or more call deltas to use in the volatility surface.
- `TimeInterpolation`: Only `Linear` is currently supported here.
- `StrikeInterpolation`: Allowable values are `Linear`, `NaturalCubic` and `FinancialCubic`.
- `Extrapolation`: A boolean value indicating if extrapolation is allowed.
- `TimeExtrapolation`: Only `Flat` is currently supported here giving flat extrapolation of the volatility.
- `StrikeExtrapolation`: Indicates the extrapolation in the strike direction. The allowable values are `None`, `UseInterpolator`, `Linear` and `Flat`. Both `Flat` and `None` give flat extrapolation in the strike direction. `UseInterpolator` and `Linear` indicate that the configured interpolation should be continued in the strike direction in order to extrapolate.
- `PriceCurveId`: This is required for a delta surface.
- `YieldCurveId`: This is required for a delta surface.

Note that, similar to the procedure outlined above for the absolute strike surface, quote strings are created from the configuration to be looked up in the market. For the put deltas, quote strings of the form `COMMODITY_OPTION/RATE_LNVOL/{N}/{C}/e_n/DEL/{T}/Put/d_m` are created. Here, `d_m` are the `PutDeltas` and `{T}` is the delta type i.e. either `Spot` or `Fwd`. Similarly for the call deltas, quote strings of the form `COMMODITY_OPTION/RATE_LNVOL/{N}/{C}/e_n/DEL/{T}/Call/d_j` are created where `d_j` are the `CallDeltas`. For ATM, quote strings of the form `COMMODITY_OPTION/RATE_LNVOL/{N}/{C}/e_n/DEL/ATM/{A}[/DEL/{T}]` are created where `{A}` is the `AtmType` i.e. `AtmSpot`, `AtmFwd` or `AtmDeltaNeutral` and `{T}` is the optional delta type.

Also, it is worth adding a note here on the interpolation for a delta based surface. Assume we want the volatility at time t and absolute strike s i.e. at the (t, s) node. For the maturity time t, a delta "slice" i.e. a set of (delta, vol) pairs for that time t, is obtained by interpolating, or extrapolating, the variance in the time direction on each delta column. Then for each (delta, vol) pair at time t, an absolute strike value is deduced to give a slice at time t in terms of absolute strike i.e. a set of (strike, vol) pairs at time t. This strike versus volatility curve is then interpolated, or extrapolated, to give the vol at the (t, s).

Listing 100 outlines the configuration for the APO volatility surface. This currently only supports a `QuoteType` of `ImpliedVolatility` and `VolatilityType` must be `Lognormal`. This configuration takes a base commodity volatility surface and builds a surface that can be queried for volatilities to price APOs directly i.e. using the volatility directly in a Black 76 formula along with the average future price. It uses the approach described in the Section entitled *Commodity Average Price Option - Future Settlement Prices* in the Product Catalogue to go from future option volatilities to APO volatilities.

We describe here briefly a motivating example encountered in practice. We have commodity APOs where the underlying is WTI Midland Argus averaged over the calendar month. We do not have direct volatilities for these APO contracts. We have a price curve for the average of WTI Midland Argus over the calendar month from the futures market. We can use the volatility surface that we have for CME WTI to build an APO surface for WTI Midland Argus. Listing 100 shows the configuration used in this context.

```xml
<CommodityVolatility>
  <CurveId>WTI_MIDLAND</CurveId>
  <CurveDescription>WTI Midland (CAL) APO surface</CurveDescription>
  <Currency>USD</Currency>
  <ApoFutureSurface>
    <QuoteType>ImpliedVolatility</QuoteType>
    <VolatilityType>Lognormal</VolatilityType>
    <MoneynessLevels>0.50,0.75,1.00,1.25,1.50</MoneynessLevels>
    <VolatilityId>CommodityVolatility/USD/WTI</VolatilityId>
    <PriceCurveId>Commodity/USD/WTI</PriceCurveId>
    <FutureConventions>WTI</FutureConventions>
    <TimeInterpolation>Linear</TimeInterpolation>
    <StrikeInterpolation>Linear</StrikeInterpolation>
    <Extrapolation>true</Extrapolation>
    <TimeExtrapolation>Flat</TimeExtrapolation>
    <StrikeExtrapolation>Flat</StrikeExtrapolation>
    <MaxTenor>2Y</MaxTenor>
    <Beta>0</Beta>
  </ApoFutureSurface>
  <DayCounter>A365</DayCounter>
  <Calendar>CME</Calendar>
  <FutureConventions>WTI_MIDLAND</FutureConventions>
  <PriceCurveId>Commodity/USD/WTI_MIDLAND</PriceCurveId>
  <YieldCurveId>Yield/USD/USD-FedFunds</YieldCurveId>
</CommodityVolatility>
```

Listing 100: APO surface configuration

The meaning of each of the elements is given below.

- `MoneynessLevels`: A comma separated list of the moneyness levels used in the APO surface construction. Forward moneyness is assumed with a value of 1 indicating a strike equal to the future price.
- `VolatilityId`: The ID of an existing commodity option surface for options on the future settlement price referenced in the APOs used in the generation of the volatilities for this surface.
- `PriceCurveId`: The ID of an existing commodity price curve for the future settlement price referenced in the APOs used in the generation of the volatilities for this surface.
- `FutureConventions`: This ID of the commodity future conventions for the future settlement price referenced in the APOs used in the generation of the volatilities for this surface.
- `TimeInterpolation`: Only `Linear` is currently supported here. Note that the interpolation is in terms of variance.
- `StrikeInterpolation`: Only `Linear` is supported here. Note that the interpolation is in terms of variance.
- `Extrapolation`: A boolean value indicating if extrapolation is allowed.
- `TimeExtrapolation`: Only `Flat` is currently supported here. The flat extrapolation is in terms of the volatility.
- `StrikeExtrapolation`: Indicates the extrapolation in the strike direction. The allowable values are `None`, `UseInterpolator`, `Linear` and `Flat`. Both `Flat` and `None` give flat extrapolation in the strike direction. `UseInterpolator` and `Linear` indicate that the configured interpolation should be continued in the strike direction in order to extrapolate.
- `PriceCurveId`: The ID of an existing commodity price curve giving the average price for the APO period.
- `YieldCurveId`: This ID of a yield curve in the currency of the commodity used for discounting.

7.8.19 Bootstrap Configuration

The BootstrapConfig node, outlined in listing 101, can be added to curve configurations that use a bootstrap algorithm to alter the default behaviour of the bootstrap algorithm.

```
<BootstrapConfig>
  <Accuracy>...</Accuracy>
  <GlobalAccuracy>...</GlobalAccuracy>
  <DontThrow>...</DontThrow>
  <MaxAttempts>...</MaxAttempts>
  <MaxFactor>...</MaxFactor>
  <MinFactor>...</MinFactor>
  <DontThrowSteps>...</DontThrowSteps>
</BootstrapConfig>
```

Listing 101: BootstrapConfig node outline

The meaning of each of the elements is:

- Accuracy [Optional]: The accuracy with which the implied quote must match the market quote for each instrument in the curve bootstrap. This node should hold a positive real number. If omitted, the default value is 1.0×10^{-12}.
- GlobalAccuracy [Optional]: If the interpolation method in the bootstrap is global, e.g. cubic spline, the bootstrap routine needs to perform multiple iterative bootstraps of the curve to converge. After each such bootstrap of the full curve, the absolute value of the change between the current bootstrap and previous bootstrap for the curve value at each pillar is calculated. The global bootstrap is deemed to have converged if the maximum of these changes is less than the global accuracy or the accuracy from the Accuracy node. This node should hold a positive real number. If omitted, the global accuracy is set equal to the accuracy from the Accuracy node. This node is useful in some cases where a slightly less restrictive accuracy, than that given by the Accuracy node, is needed for the global bootstrap.
- DontThrow [Optional]: If this node is set to true, the curve bootstrap does not throw an error when the bootstrap fails at a pillar. Instead, a curve value is sought at the failing pillar that minimises the absolute value of the difference between the implied quote and the market quote at that pillar. The minimum is sought between the minimum and maximum curve value that was used in the root finding routine that failed at the pillar. The number of steps used in this search is given by the DontThrowSteps node below. This node should hold a boolean value. If omitted, the default value is false i.e. the bootstrap throws an exception at the first pillar where the bootstrap fails.
- MaxAttempts [Optional]: At each pillar, the bootstrap routine searches between a minimum curve value and a maximum curve value for a curve value that gives an implied quote that matches the market quote at that pillar. In some cases, the minimum curve value and maximum curve value are too restrictive and the bootstrap fails at a pillar. This node determines how many times the bootstrap should be attempted at each pillar. For example, if the node is set to 1, the bootstrap uses the minimum curve value and maximum curve value implied in

the code and fails if a root is not found. If this node is set to 2 and the first attempt fails, the minimum curve value is reduced by a factor specified in the node `MinFactor`, the maximum curve value is increased by a factor specified in the node `MaxFactor` and a second attempt is made to find a root between the enlarged bounds. If no root is found, the bootstrap then fails at this pillar. This node should hold a positive integer. If omitted, the default value is 5.

- `MaxFactor` [Optional]: This node is used only if `MaxAttempts` is greater than 1. The meaning of this node is given in the description of the `MaxAttempts` node. This node should hold a positive real number. If omitted, the default value is 2.0.
- `MinFactor` [Optional]: This node is used only if `MaxAttempts` is greater than 1. The meaning of this node is given in the description of the `MaxAttempts` node. This node should hold a positive real number. If omitted, the default value is 2.0.
- `DontThrowSteps` [Optional]: This node is used only if `DontThrow` is `true`. The meaning of this node is given in the description of the `DontThrow` node. This node should hold a positive integer. If omitted, the default value is 10.

7.8.20 One Dimensional Solver Configuration

The `OneDimSolverConfig` node, outlined in Listing 102, can be added to certain curve configurations that lead to a one dimensional solver being used in the curve construction. For example, the `EquityVolatility` curve configuration can lead to equity volatilities being stripped from equity option premiums. In this case, the `OneDimSolverConfig` node can be added to the `EquityVolatility` curve configuration to indicate how the solver should behave i.e. maximum number of evaluations, initial guess, accuracy etc. The various options are outlined below.

```xml
<OneDimSolverConfig>
  <MaxEvaluations>...</MaxEvaluations>
  <InitialGuess>...</InitialGuess>
  <Accuracy>...</Accuracy>
  <MinMax>
    <Min>...</Min>
    <Max>...</Max>
  </MinMax>
  <!-- Step only needed if MinMax not provided. -->
  <Step>...</Step>
  <LowerBound>...</LowerBound>
  <UpperBound>...</UpperBound>
</OneDimSolverConfig>
```

Listing 102: `OneDimSolverConfig` node outline

The meaning of each of the elements is:

- `MaxEvaluations`: This node should hold a positive integer. The maximum number of function evaluations that can be made by the solver.
- `InitialGuess`: This node should hold a real number. The initial guess used by the solver.
- `Accuracy`: This node should hold a positive real number. The accuracy used by the solver in the root find.

- `MinMax` [Optional]: A node that holds a `Min` and a `Max` node each of which should hold a real number. This indicates that the solver should search for a root between the value in `Min` and the value in `Max`. The value in `Min` should obviously be less than the value in `Max`. This node is optional. If not provided, the `Step` node below should be provided to set up a step based solver.
- `Step` [Optional]: This node should hold a real number. The validation is a choice between `MinMax` and `Step` so that `Step` can only be provided if `MinMax` is not and vice versa. The value in `Step` provides the solver with a step size to use in its search for a root.
- `LowerBound` [Optional]: This node should hold a real number. It provides a lower bound for the search domain. If omitted, no lower bound is applied to the search domain.
- `UpperBound` [Optional]: This node should hold a real number. It provides an upper bound for the search domain. If omitted, no upper bound is applied to the search domain. Obviously, if both `LowerBound` and `UpperBound` are provided, the value in `LowerBound` should be less than the value in `UpperBound`.

7.9 Reference Data `referencedata.xml`

Reference Data is used to ease the burden on portfolio xml representation, by taking common elements and storing them as static data. Currently this can be used for *Bond Derivatives* that require bond static information.

Bond reference data is also used to build yield curves fitted to liquid bond prices, see 7.8.1.

The allowable types for ReferenceData is

1. **Bond** static data consists of the Leg data for a given bond.

```
<ReferenceData>
<!-- Bond reference datum -->
<ReferenceDatum id="SECURITY_1">
  <Type>Bond</Type>
  <BondReferenceData>
    <IssuerId>CPTY_C</IssuerId>
    <CreditCurveId>ZERO</CreditCurveId>
    <ReferenceCurveId>EURBENCHMARK-EUR-6M</ReferenceCurveId>
    <SettlementDays>2</SettlementDays>
    <Calendar>TARGET</Calendar>
    <IssueDate>20110215</IssueDate>
    <LegData>
      <LegType>Fixed</LegType>
      <Payer>false</Payer>
      <Currency>EUR</Currency>
      <Notionals>
        <Notional>1</Notional>
      </Notionals>
      <DayCounter>ActActISMA</DayCounter>
      <PaymentConvention>F</PaymentConvention>
      <FixedLegData>
        <Rates>
          <Rate>0.02</Rate>
        </Rates>
      </FixedLegData>
      <ScheduleData>
        <Rules>
          <StartDate>20190103</StartDate>
          <EndDate>20200103</EndDate>
          <Tenor>1Y</Tenor>
          <Calendar>TARGET</Calendar>
          <Convention>U</Convention>
          <TermConvention>U</TermConvention>
          <Rule>Forward</Rule>
          <EndOfMonth/>
          <FirstDate/>
          <LastDate/>
          </Rules>
        </ScheduleData>
      </LegData>
    </BondReferenceData>
  </ReferenceDatum>
</ReferenceData>
```

7.10 Ibor Fallback Config

The Ibor Fallback Configuration represents the rules for replacing Ibor reference rates by risk free rates. If no configuration is specified, a standard configuration is used. Specifying a custom configuration mainly serves testing purposes. The fields are:

- EnableIborFallbacks: If false, Ibor fallbacks are disabled.
- UseRfrCurveInTodaysMarket: If true, the todays market Ibor forwarding curve for a replaced Ibor index is built using the RfrIndex OIS curve and the Spread.
- UseRfrCurveInSimulationMarket: If true, the simulation market Ibor forward curve for a replaced Ibor index is built using the RfrIndex OIS curve and the Spread.
- Fallback: Each Ibor index to be replaced is declared by
  - IborIndex: the Ibor index name
  - RfrIndex: the rfr index name
  - Spread: the spread to apply to the rfr rate
  - SwitchDate: the date on which the fallback is used

```
<IborFallbackConfig>
        <GlobalSettings>
                <EnableIborFallbacks>true</EnableIborFallbacks>
                <UseRfrCurveInTodaysMarket>true</UseRfrCurveInTodaysMarket>
                <UseRfrCurveInSimulationMarket>true</UseRfrCurveInSimulationMarket>
        </GlobalSettings>
        <Fallbacks>
                <Fallback>
                        <IborIndex>CHF-LIBOR-12M</IborIndex>
                        <RfrIndex>CHF-SARON</RfrIndex>
                        <Spread>0.0020479999999999999</Spread>
                        <SwitchDate>2022-01-01</SwitchDate>
                </Fallback>
                <Fallback>
                        <IborIndex>CHF-LIBOR-1M</IborIndex>
                        <RfrIndex>CHF-SARON</RfrIndex>
                        <Spread>-0.000571</Spread>
                        <SwitchDate>2022-01-01</SwitchDate>
        </Fallback>
        ....
```

7.11 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.

7.11.1 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 103. The node for a tenor based zero rate is shown in Listing 104.

Listing 103: Zero conventions

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

Listing 104: 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, ThirdThursday, ThirdFriday, MondayAfterThirdFriday, TuesdayAfterThirdFriday, LastWednesday*).
- 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.

7.11.2 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 105. Alternatively, all the necessary conventions can be given explicitly without reference to an index family. The structure of this node is shown in Listing 106.

Listing 105: Deposit conventions

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

Listing 106: 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.

7.11.3 Future Conventions

A node with name *Future* is used to store conventions for money market (MM) or overnight index (OI) interest rate future quotes, for example futures on Euribor 3M or SOFR 3M underlyings. The structure of this node is shown in Listing 107. The fields have the following meaning:

- Id: The name of the convention.
- Index: The underlying index of the futures, this is either a MM (i.e. Ibor) index like e.g. EUR-EURIBOR-3M or an overnight index like e.g. USD-SOFR.
- DateGenerationRule [Optional]: This should be set to 'IMM' when the start and end dates of the future are following the IMM date logic or 'FirstDayOfMonth' when the start and end date are the first day of a month. If not given this field defaults to 'IMM'.
  - For MM futures only 'IMM' is allowed and the expiry date is determined as the next 3rd Wednesday of the expiry month of a future.
  - For an overnight index future 'IMM' means that the end date of the future is set to the 3rd Wednesday of the expiry month and the start date is set to the 3rd Wednesday of the expiry month minus the future tenor. The setting 'IMM' applies to SOFR-3M futures for example. 'FirstDayOfMonth' on the other hand means that the end date of the future is set to the first day in the month following the future's expiry month and the start date is set to the first day of the month lying n months before the end date's month where n is the number of months of the future's underlying tenor. The setting 'FirstDayOfMonth' applies to SOFR-1M futures for example. This tenor is derived from the market quote, see 10.8.
- OvernightIndexFutureNettingType [Optional]: Only relevant for OI futures. Can be 'Compounding' (which is also the default value if no value is given) or 'Averaging'. For example, SOFR 3M futures are compounding while SOFR 1M futures are averaging the daily overnight fixings over the calculation period of the future.

Listings 108, 109, 110 show examples for Euribor-3M, SOFR-3M and SOFR-1M future conventions.

Listing 107: Future conventions

```
<Future>
  <Id> </Id>
  <Index> </Index>
  <DateGenerationRule> </DateGenerationRule>
  <OvernightIndexFutureNettingType> </OvernightIndexFutureNettingType>
</Future>
```

Listing 108: Euribor 3M MM Future conventions

```
<Future>
  <Id>EURIBOR-3M-FUTURES</Id>
  <Index>EUR-EURIBOR-3M</Index>
</Future>
```

Listing 109: USD SOFR 3M OI Future conventions

```
  <Future>
    <Id>USD-SOFR-3M-FUTURES</Id>
    <Index>USD-SOFR</Index>
    <DateGenerationRule>IMM</DateGenerationRule>
    <OvernightIndexFutureNettingType>Compounding</OvernightIndexFutureNettingType>
  </Future>
```

Listing 110: USD SOFR 1M OI Future conventions

```
  <Future>
    <Id>USD-SOFR-1M-FUTURES</Id>
    <Index>USD-SOFR</Index>
    <DateGenerationRule>FirstDayOfMonth</DateGenerationRule>
    <OvernightIndexFutureNettingType>Averaging</OvernightIndexFutureNettingType>
  </Future>
```

7.11.4 FRA Conventions

A node with name *FRA* is used to store conventions for FRA quotes. The structure of this node is shown in Listing 111. 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 111: FRA conventions

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

7.11.5 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 112.

Listing 112: 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.

7.11.6 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 113.

Listing 113: 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".

7.11.7 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 114.

Listing 114: 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.

7.11.8 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 115.

Listing 115: 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*.

7.11.9 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 116.

Listing 116: Tenor basis two swap conventions

```
<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*.

7.11.10 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 117.

Listing 117: 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.

7.11.11 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 118.

Listing 118: 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*.

7.11.12 Inflation Swap 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 119

Listing 119: 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 or 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.
- `AdjustInflationObservationDates`: Flag indicating whether index observation dates should be adjusted or not.
- `InflationCalendar`: The calendar for the inflation leg of the swap.
- `InflationConvention`: The rolling convention for the inflation leg of the swap.
- `PublicationRoll`: This is an optional node taking the values `None`, `OnPublicationDate` or `AfterPublicationDate`. If omitted, the value `None` is used. Currently, our only known use case for a value other than `None` is for Australian zero coupon inflation indexed swaps (ZCIIS). Here, the index is published quarterly on the last Wednesday of the month following the end of the reference quarter. The start date and maturity date of the market quoted ZCIIS roll to the next quarterly date after the publication date of the index. For example, the AU CPI value for Q3 2020, i.e. 1 Jul 2020 to 30 Sep 2020 was

released on 28 Oct 2020. On 27 Oct 2020, before the index publication date, the market 5Y ZCIIS would start on 15 Sep 2020 and end on 15 Sep 2025 and reference the Q2 inflation index value. On 29 Oct 2020, after the index publication date, the market 5Y ZCIIS would start on 15 Dec 2020 and end on 15 Dec 2025 and reference the Q3 inflation index value. On the release date, i.e. 28 Oct 2020, the market ZCIIS that is set up is determined by whether the `PublicationRoll` value is `OnPublicationDate` or `AfterPublicationDate`. If it is set to `OnPublicationDate`, the swap rolls on this date and hence the market 5Y ZCIIS would start on 15 Dec 2020 and end on 15 Dec 2025 and reference the Q3 inflation index value. If it is set to `AfterPublicationDate`, the swap does not roll on the publication date and instead rolls on the next day, and hence the market 5Y ZCIIS would start on 15 Sep 2020 and end on 15 Sep 2025 and reference the Q2 inflation index value. The publication schedule for the index must be provided in the `PublicationSchedule` node if `PublicationRoll` is not `None`. An example of the AU CPI conventions set up is given in Listing 120.

- `PublicationSchedule`: This is an optional node and is not used if `PublicationRoll` is `None`. If `PublicationRoll` is not `None`, it must be provided and gives the publication dates for the inflation index. The node fields are the same fields that are described in the Section 8.3.4, i.e. they are `ScheduleData` elements. An example of the AU CPI conventions set up is given in Listing 120. The `PublicationSchedule` must cover the dates on which you intend to perform valuations, i.e. the first publication schedule date must be less than the smallest valuation date that you intend to use and the last publication schedule date must be greater than the largest valuation date that you intend to use.

Listing 120: AU CPI inflation swap conventions

```xml
<InflationSwap>
  <Id>AUCPI_INFLATIONSWAP</Id>
  <FixCalendar>AUD</FixCalendar>
  <FixConvention>F</FixConvention>
  <DayCounter>30/360</DayCounter>
  <Index>AUCPI</Index>
  <Interpolated>false</Interpolated>
  <ObservationLag>3M</ObservationLag>
  <AdjustInflationObservationDates>false</AdjustInflationObservationDates>
  <InflationCalendar>AUD</InflationCalendar>
  <InflationConvention>F</InflationConvention>
  <PublicationRoll>AfterPublicationDate</PublicationRoll>
  <PublicationSchedule>
    <Rules>
      <StartDate>2001-01-24</StartDate>
      <EndDate>2030-01-30</EndDate>
      <Tenor>3M</Tenor>
      <Calendar>AUD</Calendar>
      <Convention>Preceding</Convention>
      <TermConvention>Unadjusted</TermConvention>
      <Rule>LastWednesday</Rule>
    </Rules>
  </PublicationSchedule>
</InflationSwap>
```

7.11.13 CMS Spread Option Conventions

A node with name *CmsSpreadOption* is used to store the conventions.

Listing 121: 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.

7.11.14 Ibor Index Conventions

A node with name *IborIndex* is used to store conventions for Ibor indices. This can be used to define new Ibor indices without the need of adding them to the C++ code, or also to override the conventions of existing Ibor indices.

Listing 122: Ibor index convention

```
<IborIndex>
  <Id>EUR-EURIBOR_ACT365-3M</Id>
  <FixingCalendar>TARGET</FixingCalendar>
  <DayCounter>A365F</DayCounter>
  <SettlementDays>2</SettlementDays>
  <BusinessDayConvention>MF</BusinessDayConvention>
  <EndOfMonth>true</EndOfMonth>
</IborIndex>
```

The meaning of the elements is as follows:

- Id: The index name. This must be of the form “CCY-NAME-TENOR” with a currency “CCY”, an index name “NAME” and a string “TENOR” representing a period. The name should not be “GENERIC”, since this is reserved.

- FixingCalendar: The fixing calendar of the index.
- DayCounter: The day count convention used by the index.
- SettlementDays: The settlement days for the index. This must be a non-negative whole number.
- BusinessDayConvention: The business day convention used by the index.
- EndOfMonth: A flag indicating whether the index employs the end of month convention.

Notice that if another convention depends on an Ibor index convention (because it contains the Ibor index name defined in the latter convention), the Ibor index convention must appear before the convention that depends on it in the convention input file.

Also notice that customised indices can not be used in cap / floor volatility surface configurations.

7.11.15 Overnight Index Conventions

A node with name *OvernightIndex* is used to store conventions for Overnight indices. This can be used to define new Overnight indices without the need of adding them to the C++ code, or also to override the conventions of existing Overnight indices.

Listing 123: Overnight index convention

```
<OvernightIndex>
  <Id>EUR-ESTER</Id>
  <FixingCalendar>TARGET</FixingCalendar>
  <DayCounter>A360</DayCounter>
  <SettlementDays>0</SettlementDays>
</OvernightIndex>
```

The meaning of the elements is as follows:

- Id: The index name. This must be of the form “CCY-NAME” with a currency “CCY” and an index name “NAME”. The name should not be “GENERIC”, since this is reserved.
- FixingCalendar: The fixing calendar of the index.
- DayCounter: The day count convention used by the index.
- SettlementDays: The settlement days for the index. This must be a non-negative whole number.

Notice that if another convention depends on an Overnight index convention (because it contains the Overnight index name defined in the latter convention), the Overnight index convention must appear before the convention that depends on it in the convention input file.

Also notice that customised indices can not be used in cap / floor volatility surface configurations.

7.11.16 Inflation Index Conventions

A node with the name `ZeroInflationIndex` is used to store data for the creation of a new inflation index. This avoids having to add the index definition to the C++ code and recompile. Note that the `ZeroInflationIndex` node should be placed before its use in any other convention, e.g. in an `InflationSwap` convention, to avoid an error due to the new index itself not being created. If the `Id` node matches an existing inflation index, the newly created index will take precedence and its defintion will be used in the code for the given `Id`.

Listing 124: ZeroInflationIndex *node*

```
<ZeroInflationIndex>
  <Id>...</Id>
  <RegionName>...</RegionName>
  <RegionCode>...</RegionCode>
  <Revised>...</Revised>
  <Frequency>...</Frequency>
  <AvailabilityLag>...</AvailabilityLag>
  <Currency>...</Currency>
</ZeroInflationIndex>
```

The meaning of each element is as follows:

- `Id`: The new inflation index name.
- `RegionName`: The name of the region with which the inflation index is associated.
- `RegionCode`: A code for the region with which the inflation index is associated.
- `Revised`: A boolean flag indicating whether the index is a revised index or not. This is generally set to `false` but is left as an option to align with the C++ `InflationIndex` class definition.
- `Frequency`: A valid frequency indicating the publication frequency of the inflation index, generally `Monthly`, `Quarterly` or `Annual`.
- `AvailabilityLag`: A valid period indicating the lag between the inflation index publication for a given period and the period itself. For example, if March's inflation index value is published in April, the `AvailabilityLag` would be `1M`.
- `Currency`: The ISO currency code of the currency associated with the inflation index, generally the currency of the region.

7.11.17 Swap Index Conventions

A node with name *SwapIndex* is used to store conventions for Swap indices (also known as "CMS" indices).

Listing 125: Swap index convention

```
  <SwapIndex>
    <Id>EUR-CMS-2Y</Id>
    <Conventions>EUR-EURIBOR-6M-SWAP</Conventions>
    <FixingCalendar>TARGET</FixingCalendar>
  </SwapIndex>
```

The meaning of the elements is as follows:

- Id: The index name. This must be of the form "CCY-CMS-TENOR" with a currency "CCY" and a string "TENOR" representing a period. The index name can contain an optional tag "CCY-CMS-TAG-TENOR" which is an arbitrary label that allows to define more than one swap index per currency.
- Conventions: A swap convention defining the index conventions.
- FixingCalendar [Optional]: The fixing calendar for the swap index fixings publication. If not given, the fixed leg calendar from the swap conventions will be used as a fall back.

7.11.18 FX Option Conventions

A node with name *FxOption* is used to store conventions for FX option quotes for a given currency pair. The structure of this node is shown in Listing 126.

Listing 126: FX option conventions

```
<FxOption>
  <Id>EUR-USD-FXOPTION</Id>
  <FXConventionID>EUR-USD-FX</FXConventionID>
  <AtmType>AtmDeltaNeutral</AtmType>
  <DeltaType>Spot</DeltaType>
  <SwitchTenor>2Y</SwitchTenor>
  <LongTermAtmType>AtmDeltaNeutral</LongTermAtmType>
  <LongTermDeltaType>Fwd</LongTermDeltaType>
  <RiskReversalInFavorOf>Call</RiskReversalInFavorOf>
  <ButterflyStyle>Broker</ButterflyStyle>
</FxOption>
```

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

- FXConventionID: The FX convention for the currency pair (see 7.11.10). Optional, if not given, the FX spot days default to 2 and the advance calendar defaults to source ccy + target ccy default calendars.
- AtmType: Convention of ATM option quote (Choices are *AtmNull, AtmSpot, AtmFwd, AtmDeltaNeutral, AtmVegaMax, AtmGammaMax, AtmPutCall50*).
- DeltaType: Convention of Delta option quote (Choices are *Spot, Fwd, PaSpot, PaFwd*).

- SwitchTenor [Optional]: If given, different ATM and Delta conventions will be used if the option tenor is greater or equal the switch tenor ("long term" atm and delta type)
- LongTermAtmType [Mandatory if and only if SwitchTenor is given]: ATM type to use for options with tenor > switch point, if SwitchTenor is given
- LongTermDeltaType [Mandatory if and only if SwitchTenor is given]: Delta type to use for options with tenor > switch point, if SwitchTenor is given
- RiskReversalInFavorOf [Optional]: Call (default), Put. Only relevant for BF, RR market data input.
- ButterflyStyle [Optional]: Broker (default), Smile. Only relevant for BF, RR market data input.

7.11.19 Commodity Forward Conventions

A node with name `CommodityForward` is used to store conventions for commodity forward price quotes. The structure of this node is shown in Listing 127.

Listing 127: Commodity forward conventions

```
<CommodityForward>
  <Id>...</Id>
  <SpotDays>...</SpotDays>
  <PointsFactor>...</PointsFactor>
  <AdvanceCalendar>...</AdvanceCalendar>
  <SpotRelative>...</SpotRelative>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Outright>...</Outright>
</CommodityForward>
```

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

- `Id`: The identifier for the commodity forward convention. The identifier here should match the `Name` that would be provided for the commodity in the trade XML as described in Table 24.
- `SpotDays` [Optional]: The number of business days to spot for the commodity. Any non-negative integer is allowed here. If omitted, this takes a default value of 2.
- `PointsFactor` [Optional]: This is only used if `Outright` is `false`. Any positive real number is allowed here. When `Outright` is `false`, the commodity forward quotes are provided as points i.e. a number that should be added to the commodity spot to give the outright commodity forward rate. The `PointsFactor` is the number by which the points quote should be divided before adding it to the spot quote to obtain the forward price. If omitted, this takes a default value of 1.
- `AdvanceCalendar` [Optional]: The business day calendar(s) used for advancing dates for both spot and forwards. The allowable values are given in Table 16. If omitted, it defaults to `NullCalendar` i.e. a calendar where all days are considered good business days.

- `SpotRelative` [Optional]: The allowable values are `true` and `false`. If `true`, the forward tenor is interpreted as being relative to the spot date. If `false`, the forward tenor is interpreted as being relative to the valuation date. If omitted, it defaults to `True`.
- `BusinessDayConvention` [Optional]: The business day roll convention used to adjust dates when getting from the valuation date to the spot date and the forward maturity date. The allowable values are given in Table 14. If omitted, it defaults to `Following`.
- `Outright` [Optional]: The allowable values are `true` and `false`. If `true`, the forward quotes are interpreted as outright forward prices. If `false`, the forward quotes are interpreted as points i.e. as a number that must be added to the spot price to get the outright forward price. If omitted, it defaults to `true`.

7.11.20 Commodity Future Conventions

A node with name `CommodityFuture` is used to store conventions for commodity future contracts and options on them. These conventions are used in commodity derivative trades and commodity curve construction to calculate contract expiry dates. The structure of this node is shown in Listing 128.

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

- `Id`: The identifier for the commodity future convention. The identifier here should match the `Name` that would be provided for the commodity in the trade XML as described in Table 24.
- `AnchorDay` [Optional]: This node is not applicable for daily future contracts and hence is optional. It is necessary for future contracts with a monthly cycle or greater. This node is used to give a date in the future contract month to use as a base date for calculating the expiry date. It can contain a `DayOfMonth` node, a `CalendarDaysBefore` node or an `NthWeekday` node:
  - The `DayOfMonth` This node can contain any integer in the range $1, \ldots, 31$ indicating the day of the month. A value of 31 will guarantee that the last day in the month is used a base date.
  - The `CalendarDaysBefore` This node can contain any non-negative integer. The contract expiry date is this number of calendar days before the first calendar day of the contract month.
  - The `NthWeekday` This node has the elements shown in Listing 129. This node is used to indicate a date in a given month in the form of the n-th named weekday of that month e.g. 3rd Wednesday. The allowable values for `Nth` are $1, 2, 3, 4$. The `Weekday` node takes a weekday in the form of the first three characters of the weekday with the first character capitalised.
- `ContractFrequency`: This node indicates the frequency of the commodity future contracts. The value here is usually `Monthly` or `Quarterly`.
- `Calendar`: The business day trading calendar(s) applicable for the commodity future contract.

Listing 128: Commodity future conventions

```
<CommodityFuture>
  <Id>...</Id>
  <AnchorDay>
    ...
  </AnchorDay>
  <ContractFrequency>...</ContractFrequency>
  <Calendar>...</Calendar>
  <ExpiryCalendar>...</ExpiryCalendar>
  <ExpiryMonthLag>...</ExpiryMonthLag>
  <OneContractMonth>...</OneContractMonth>
  <OffsetDays>...</OffsetDays>
  <BusinessDayConvention>...</BusinessDayConvention>
  <AdjustBeforeOffset>...</AdjustBeforeOffset>
  <IsAveraging>...</IsAveraging>
  <OptionExpiryOffset>...</OptionExpiryOffset>
  <ProhibitedExpiries>
    <Dates>
      <Date forFuture="true" convention="Preceding" forOption="true"
      ↪  optionConvention="Preceding">...</Date>
        ...
    </Dates>
  </ProhibitedExpiries>
  <OptionExpiryMonthLag>...</OptionExpiryMonthLag>
  <OptionExpiryDay>...</OptionExpiryDay>
  <OptionBusinessDayConvention>...</OptionBusinessDayConvention>
  <FutureContinuationMappings>
    <ContinuationMapping>
      <From>...</From>
      <To>...</To>
    </ContinuationMapping>
    ...
  </FutureContinuationMappings>
  <OptionContinuationMappings>
    <ContinuationMapping>
      <From>...</From>
      <To>...</To>
    </ContinuationMapping>
    ...
  </OptionContinuationMappings>
  <AveragingData>
    ...
  </AveragingData>
  <HoursPerDay>...</HoursPerDay>
</CommodityFuture>
```

- `ExpiryCalendar` [Optional]: The business day expiry calendar(s) applicable for the commodity future contract. This calendar is used when deriving expiry dates. If omitted, this defaults to the trading day calendar specified in the `Calendar` node.
- `ExpiryMonthLag` [Optional]: The allowable values are any integer. This value indicates the number of months from the month containing the expiry date to the contract month. If 0, the commodity future contract expiry date is in the contract month. If the value of `ExpiryMonthLag` is $n > 0$, the commodity future

contract expires in the n-th month prior to the contract month. If the value of ExpiryMonthLag is $n < 0$, the commodity future contract expires in the n-th month after the contract month. The value of ExpiryMonthLag is generally 0, 1 or 2. For example, NYMEX:CL has an ExpiryMonthLag of 1 and ICE:B has an ExpiryMonthLag of 2. If omitted, it defaults to 0.

- OneContractMonth [Optional]: This node takes a calendar month in the form of the first three characters of the month with the first character capitalised. The month provided should be an arbitrary valid future contract month. It is used in cases where the ContractFrequency is not Monthly in order to determine the valid contract months. If omitted, it defaults to January.
- OffsetDays [Optional]: The number of business days that the expiry date is before the base date where the base date is implied by the AnchorDay node above. Any non-negative integer is allowed here. If omitted, this takes a default value of zero.
- BusinessDayConvention [Optional]: The business day roll convention used to adjust the expiry date. The allowable values are given in Table 14. If omitted, it defaults to Preceding.
- AdjustBeforeOffset [Optional]: The allowable values are true and false. If true, if the base date implied by the AnchorDay node above is not a good business day according to the calendar provided in the Calendar node, this date is adjusted before the offset specified in the OffsetDays is applied. If false, this adjustment does not happen. If omitted, it defaults to true.
- IsAveraging [Optional]: The allowable values are true and false. This node indicates if the future contract is based on the average commodity price of the contract period. If omitted, it defaults to false.
- OptionExpiryOffset [Optional]: The number of business days that the option expiry date is before the future expiry date. Any non-negative integer is allowed here. If omitted, this takes a default value of zero and the expiry date of an option on the future contract is assumed to equal the expiry date of the future contract.
- ProhibitedExpiries [Optional]: This node can be used to specify explicit dates which are not allowed as future contract expiry dates or as option expiry dates. A useful example of this is the ICE Brent contract which has the following constraint on expiry dates: *If the day on which trading is due to cease would be either: (i) the Business Day preceding Christmas Day, or (ii) the Business Day preceding New Year's Day, then trading shall cease on the next preceding Business Day.* Each Date node can take optional attributes. The default values of these attributes is shown in Listing 128. The convention attribute accepts a valid business day convention in the list Preceding, ModifiedPreceding, Following and ModifiedFollowing. This convention indicates how the future expiry date should be adjusted if it lands on the prohibited expiry Date. If ommitted, the default is Preceding. Both Preceding and ModifiedPreceding indicate that the next available business day before the date is tested. Following and ModifiedFollowing indicate that the next available business day after the date is tested. The optionConvention attribute allows the same values and behaves

in the same way to determine how the option expiry date should be adjusted if it lands on the prohibited expiry `Date`. The `forFuture` and `forOption` boolean attributes enable the prohibited expiry to apply only for the future expiry date or the option expiry date respectively by setting the value to `false`.

- `OptionExpiryMonthLag` [Optional]: The allowable values are any integer. This value indicates the number of months from the month containing the option expiry date to the month containing the expiry date. If 0, the commodity future option contract expiry date is anchored in the same month as the commodity future contract expiry date. If the value of `OptionExpiryMonthLag` is $n > 0$, the commodity option future contract expires in the n-th month prior to the commodity future contract expiry month. If the value of `OptionExpiryMonthLag` is $n < 0$, the commodity option future contract expires in the n-th month after the the commodity future contract expiry month. The value of `OptionExpiryMonthLag` should be equal to `ExpiryMonthLag` when `OptionExpiryOffset` is used. The `OptionExpiryMonthLag` is rarely used. An example is the Crude Palm Oil contract `XKLS:FCPO` where the future contract expiry is in the delivery month and the option expiry is in the month that is 2 months prior to this. In this case, `OptionExpiryMonthLag` is 2. If omitted, `OptionExpiryMonthLag` defaults to 0.

- `OptionExpiryDay` [Optional]: This node can contain any integer in the range $1, \ldots, 31$ indicating the day of the month on which an option expiry date is anchored. A value of 31 will guarantee that the last day in the month is used a base date. If omitted, this is not used. Setting this field takes precedence over `OptionExpiryOffset`.

- `OptionBusinessDayConvention` [Optional]: The business day convention used to adjust the option expiry date to a good business day if `OptionExpiryDay` is used.

- `FutureContinuationMappings` [Optional]: When building future curves, we may use market data that has a continuation expiry, i.e. `c1`, `c2`, etc. , as opposed to an explicit expiry date or tenor. In some cases, the continuation expiries coming from the market data provider may skip serial months and therefore we use the mapping here to map from the market data provider index to the relevant serial month.

- `OptionContinuationMappings` [Optional]: When building option volatility structures, we may use market data that has a continuation expiry, i.e. `c1`, `c2`, etc. , as opposed to an explicit expiry date or tenor. In some cases, the continuation expiries coming from the market data provider may skip serial months and therefore we use the mapping here to map from the market data provider index to the relevant serial month. For example, for the Crude Palm Oil contract `XKLS:FCPO`, the option expiry months are serial up to the 9th month and then alternate months. So, we would add a mapping from 10 to 11, 11 to 13 and so on so that the correct option expiry is arrived at when reading the market data quotes and constructing the option volatility structure.

- `AveragingData` [Optional]: This node is needed for future contracts that are used in a piecewise commodity curve `PriceSegment` and whose underlying is the average of other future prices or spot prices over a given period. An example is the ICE PMI power contract with contract specifications outlined here. It is described in detail below.

- `HoursPerDay` [Optional]: For power derivatives, quantities are sometimes given as a quantity per hour. To deduce the quantity for the day which is multiplied by that day's future price, one needs to know the number of hours in the day associated with the future price. For example ICE PDQ is the daily PJM Western Hub Real Time Peak future contract. The price each day for this contract is the average of the locational marginal prices (LMPs) for all hours ending 08:00 to 23:00 Eastern Pacific Time. In other words, there are 16 hours in the day that feed in to the average yielding this settlement price. For this contract, `HoursPerDay` would be `16`. This field is only needed if a trade XML references this commodity contract, has `CommodityQuantityFrequency` set to `PerHour` and has no `HoursPerDay` value set directly in the XML.

Listing 129: `NthWeekday` *node outline*

```
<NthWeekday>
  <Nth>...</Nth>
  <Weekday>...</Weekday>
</NthWeekday>
```

An example `CommodityFuture` node for the NYMEX WTI future contract, specified here, is provided in Listing 130.

Listing 130: NYMEX WTI `CommodityFuture` *node*

```
<CommodityFuture>
  <Id>NYMEX:CL</Id>
  <AnchorDay>
    <DayOfMonth>25</DayOfMonth>
  </AnchorDay>
  <ContractFrequency>Monthly</ContractFrequency>
  <Calendar>US-NYSE</Calendar>
  <ExpiryMonthLag>1</ExpiryMonthLag>
  <OffsetDays>3</OffsetDays>
  <BusinessDayConvention>Preceding</BusinessDayConvention>
  <IsAveraging>false</IsAveraging>
</CommodityFuture>
```

The `AveragingData` node referenced above has the structure shown in Listing 131. The meaning of each of the fields is as follows:

- `CommodityName`: The name of the commodity being averaged.
- `Conventions`: The identifier for the conventions associated with the commodity being averaged.
- `Period`: This indicates the averaging period relative to the future expiry date. The allowable values are:
  - `PreviousMonth`: The calendar month prior to the month in which the (top level) future contract's expiry date falls is used as the averaging period.

  - `ExpiryToExpiry`: Given a (top level) future contract's expiry date, the averaging period is from and excluding the previous expiry date to and including the expiry date.

- `PricingCalendar`: The pricing calendar(s) used to determine the pricing dates in the averaging period.

- `UseBusinessDays` [Optional]: A boolean flag that defaults to `true` if omitted. When set to `true`, the pricing dates in the averaging period are the set of `PricingCalendar` good business days. When set to `false`, the pricing dates in the averaging period are the complement of the set of `PricingCalendar` good business days. This may be useful in certain situations. For example, the contract ICE PW2 with specifications here averages the PJM Western Hub locational marginal prices over each day in the averaging period that is a Saturday, Sunday or NERC holiday. So, in this case, `UseBusinessDays` would be `false` and `PricingCalendar` would be `US-NERC`.

- `DeliveryRollDays` [Optional]: This node allows any non-negative integer value. When averaging a commodity future contract price over the averaging period, the averaging period may include an underlying future contract expiry date. This node's value indicates when we should begin using the next future contract's price in the averaging. If the value is zero, we should include the future contract prices up to and including the contract expiry. If the value is one, we should include the contract prices up to and including the day that is one business day before the contract expiry and then switch to using the next future contract's price thereafter. Similarly for other non-negative integer values. If this node is omitted, it is set to zero.

- `FutureMonthOffset` [Optional]: This node allows any non-negative integer value. If this node is omitted, it is set to zero. This node indicates which future contract is being referenced on each *Pricing Date* in the averaging period by acting as an offset from the next available expiry date. If `FutureMonthOffset` is zero, the settlement price of the next available monthly contract that has not expired with respect to the *Pricing Date* is used as the price on that *Pricing Date*. If `FutureMonthOffset` is one, the settlement price of the second available monthly contract that has not expired with respect to the *Pricing Date* is used as the price on that *Pricing Date*. Similarly for other positive values of `FutureMonthOffset`.

- `DailyExpiryOffset` [Optional]: This node allows any non-negative integer value. It should only be used where the `CommodityName` being averaged has a daily contract frequency. If this node is omitted, it is set to zero. This node indicates which future contract is being referenced on each *Pricing Date* in the averaging period by acting as a business day offset, using the `CommodityName`'s expiry calendar, from the *Pricing Date*. It is useful in the base metals market where the future contract being averaged on each *Pricing Date* is the cash contract on that *Pricing Date* i.e. the contract with expiry date two business days after the *Pricing Date*.

Listing 131: ***AveragingData*** *node structure*

```
<AveragingData>
  <CommodityName>...</CommodityName>
  <Conventions>...</Conventions>
  <Period>...</Period>
  <PricingCalendar>...</PricingCalendar>
  <UseBusinessDays>...</UseBusinessDays>
  <DeliveryRollDays>...</DeliveryRollDays>
  <FutureMonthOffset>...</FutureMonthOffset>
  <DailyExpiryOffset>...</DailyExpiryOffset>
</AveragingData>
```

7.11.21 Credit Default Swap Conventions

A node with name `CDS` is used to store conventions for credit default swaps. The structure of this node is shown in Listing 132.

Listing 132: CDS conventions

```
<CDS>
  <Id>...</Id>
  <SettlementDays>...</SettlementDays>
  <Calendar>...</Calendar>
  <Frequency>...</Frequency>
  <PaymentConvention>...</PaymentConvention>
  <Rule>...</Rule>
  <DayCounter>...</DayCounter>
  <SettlesAccrual>...</SettlesAccrual>
  <PaysAtDefaultTime>...</PaysAtDefaultTime>
</CDS>
```

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

- `Id`: The identifier for the CDS convention.
- `SettlementDays`: The number of days after the CDS trade date when protection starts i.e. the *Protection effective date* or *step-in date.* Any non-negative integer is allowed here. For standard CDS after, this is generally set to 1.
- `Calendar`: The calendar associated with the CDS. For non-JPY currencies, this is generally `WeekendsOnly` to agree with the ISDA standard. For JPY CDS, the ISDA standard calendar is `TYO` documented at `https://www.cdsmodel.com/cdsmodel`. This could be set up as an additional calendar or `JPN` could be used as a proxy. Allowable calendar values are given in Table 16.
- `Frequency`: The frequency of fee leg payments for the CDS. The ISDA standard is `Quarterly` but any valid frequency is allowed.
- `PaymentConvention`: The business day convention for payments on the CDS. The ISDA standard is `Following` but any valid business day convention from Table 14 is allowed.

- `Rule`: The date generation rule for the fee leg on the CDS. The ISDA standard is `CDS2015` but any valid date generation rule is allowed.
- `DayCounter`: The day counter for fee leg payments on the CDS. The ISDA standard is `A360` but any valid day counter from Table 17 is allowed.
- `SettlesAccrual`: A boolean value indicating if an accrued fee is due on the occurrence of a credit event. Allowable boolean values are given in the Table 28. In general, this is set `true`.
- `PaysAtDefaultTime`: A boolean value indicating if the accrued fee, on the occurrence of a credit event, is payable at the credit event date or the end of the fee period. A value of `true` indicates that the accrued is payable at the credit event date and a value of `false` indicates that it is payable at the end of the fee period. In general, this is set `true`.

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 133.

Listing 133: 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 8.3.3. An example structure of a two-legged `SwapData` node is shown in Listing 134.

- Settlement [Optional]: Delivery type applicable to cross currency swaps, and ignored for all other swap types. Delivery type does not impact pricing in ORE.

  Settlement *Cash* indicates that principal exchanges on the cross currency swap should be included in Initial Margin (IM). According to ISDA non-deliverable (*Cash*) trades are excluded from the exemption from IM for the principal exchange, i.e. the principal exchanges are included in IM.

  Settlement *Physical* indicates that principal exchanges on the cross currency swap should be excluded in IM (the ISDA exemption applies).

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

Listing 134: Swap data

```
<SwapData>
  <Settlement>Cash</Settlement>
  <LegData>
    ...
  </LegData>
  <LegData>
    ...
  </LegData>
</SwapData>
```

8.2.2 Zero Coupon Swap

A Zero Coupon swap is set up as a swap (trade type *Swap*) , with one leg of type `ZeroCouponFixed`. Listing 135 shows an example. The ZeroCouponFixed leg contains an additional `ZeroCouponFixedLegData` block. See 8.3.18 for details on the ZeroCouponFixed leg specification.

Listing 135: Zero Coupon Swap Data

```
<SwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Payer>true</Payer>
    ...
  </LegData>
  <LegData>
    <LegType>ZeroCouponFixed</LegType>
    <Payer>false</Payer>
    ...
    <ZeroCouponFixedLegData>
        <Rates>
            <Rate>0.02</Rate>
        </Rates>
        <Compounding>Simple</Compounding>
    </ZeroCouponFixedLegData>
  </LegData>
</SwapData>
```

8.2.3 Cap/Floor

The `CapFloorData` node is the trade data container for the *CapFloor* trade type. 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 can be set to *Floating*, *CMS*, *DurationAdjustedCMS*, *CPI* or *YY*, plus elements for the Cap and Floor rates. An example structure with Cap rates is shown in in Listing 136. A `CapFloorData` node must have either `Caps` or `Floors` elements, or both. In the case of both (I.e. a collar with long cap and short floor) the sequence is that `Caps` elements must be above the `Floors` elements. Note that the `Caps` and `Floors` elements must be outside the `LegData` sub-node, i.e. a *CapFloor* can't have a capped or floored *Floating* or *CMS* leg. THe *Payer* flag in the LegData subnode is ignored for this instrument. Notice that the signs in the definition of a collar (long cap, short floor) for the CapFloor instruments is exactly opposite to 8.3.6.

Listing 136: Cap/Floor data

```
<CapFloorData>
  <LongShort>Long</LongShort>
  <LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
     ...
  </LegData>
  <Caps>
    <Cap>0.05</Cap>
  </Caps>
  <Premiums>
    <Premium>
      <Amount>1000</Amount>
      <Currency>EUR</Currency>
      <PayDate>2021-01-27</PayDate>
    </Premium>
  </Premiums>
</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 8.3.3. Exactly one `LegData` node is allowed, and the LegType element must be set to *Floating* (Ibor and OIS), *CMS*, *DurationAdjustedCMS*, *CPI* or *YY*.
- Caps: This node has child elements of type `Cap` capping the floating leg (after applying spread if any). 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 (after applying spread if any). 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%
- Premiums [Optional]: Option premium amounts paid by the option buyer to the

option seller.

Allowable values: See section 8.3.2

8.2.4 Forward Rate Agreement

A forward rate agreement (trade type *ForwardRateAgreement* is set up using a `ForwardRateAgreementData` block as shown in listing 137. The forward rate agreement specific elements are:

- StartDate: A FRA expires/settles on the startDate.
  Allowable values: See `Date` in Table 14.

- 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 14.

- Currency: The currency of the FRA notional.
  Allowable values: See `Currency` in Table 14.

- 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 18.

- 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 137: 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>
```

8.2.5 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 8.3.3.

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. There must be at least one full coupon period after the exercise date for European Swaptions, and after the last exercise date for Bermudan Swaptions. See Table 11 for further details on requirements for swaptions.

The structure of an example `SwaptionData` node of a European swaption is shown in Listing 138.

Listing 138: Swaption data

```
<SwaptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <Style>European</Style>
        <Settlement>Physical</Settlement>
        <ExerciseDates>
          <ExerciseDate>2027-03-02</ExerciseDate>
        </ExerciseDates>
        ...
        <Premiums>
          <Premium>
            <Amount>807000</Amount>
            <Currency>GBP</Currency>
            <PayDate>2021-06-15</PayDate>
          </Premium>
        </Premiums>
    </OptionData>
    <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        <Currency>GBP</Currency>
        ...
    </LegData>
    <LegData>
        <LegType>Floating</LegType>
        <Payer>true</Payer>
        <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. Note that the exercise date of a *European* swaption must be before or on the start date of the underlying swap.
`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 11: Requirements for Swaptions

The `OptionData` trade component sub-node is outlined in section 8.3.1. The relevant fields in the `OptionData` node for a Swaption are:

- `LongShort`: The allowable values are *Long* or *Short*. Note that the payer and receiver legs in the underlying swap are always from the perspective of the party that is *Long*. E.g. for a *Short* swaption with a fixed leg where the Payer flag is set to *false*, it means that the counterparty receives the fixed flows.

`LongShort`	`Payer` for Fixed leg on underlying Swap	`Payer` for Floating leg on underlying Swap	Resulting Set Up and Flows
Long	*true*	*false*	The Party to the trade buys an option to enter a swap where the Party pays fixed and receives floating
Short	*true*	*false*	The Party to the trade sells an option to the Counterparty to enter a swap where the Counterparty pays fixed and receives floating
Long	*false*	*true*	The Party to the trade buys an option to enter a swap where the Party receives fixed and pays floating
Short	*false*	*true*	The Party to the trade sells an option to the Counterparty to enter a swap where the Counterparty receives fixed and pays floating

Table 12: Swaption set up and resulting flows

- `OptionType`[Optional]: This flag is optional for swaptions, and even if set, has no impact. Whether a swaption is a payer or receiver swaption is determined by the Payer flags on the legs of the underlying swap.

- `Style`: The exercise style of the Swaption. The allowable values are *European* or *Bermudan*.
- `NoticePeriod`[Optional]: The notice period defining the date (relative to the exercise date) on which the exercise decision has to be taken. If not given the notice period defaults to *0D*, i.e. the notice date is identical to the exercise date. Allowable values: A number followed by *D, W, M, or Y*
- `NoticeCalendar`[Optional]: The calendar used to compute the notice date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 16 `Calendar`.
- `NoticeConvention`[Optional]: The convention used to compute the notice date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See `Roll Convention` in Table 14.
- `Settlement`: Delivery Type. The allowable values are *Cash* or *Physical*.
- `SettlementMethod`[Optional]: Specifies the method to calculate the settlement amount for Swaptions and CallableSwaps. 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]
- `ExerciseFees`[Optional]: This node contains child elements of type `ExerciseFee`. Similar to a list of notionals (see 8.3.3) the fees can be given either
  - as a list where each entry corresponds to an exercise date and the last entry is used for all remaining exercise dates if there are more exercise dates than exercise fee entries, or
  - using the `startDate` attribute to specify a change in a fee from a certain day on (w.r.t. the exercise date schedule)

  Fees can either be given as an absolute amount or relative to the current notional of the period immediately following the exercise date using the `type` attribute together with specifiers `Absolute` resp. `Percentage`. If not given, the type defaults to `Absolute`. `Percentage` fees are expressed in decimal form, e.g. 0.05 is a fee of 5% of notional.

  If a fee is given as a positive number the option holder has to pay a corresponding amount if they exercise the option. If the fee is negative on the other hand, the option holder receives an amount on the option exercise.

  Only supported for Swaptions and Callable Swaps currently.

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

- `ExerciseFeeSettlementPeriod`[Optional]: The settlement lag for exercise fee payments. Defaults to 0D if not given. This lag is relative to the exercise date (as opposed to the notice date). Allowable values: A number followed by *D, W, M, or Y*
- `ExerciseFeeSettlementCalendar`[Optional]: The calendar used to compute the exercise fee settlement date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 16 Calendar.
- `ExerciseFeeSettlementConvention`[Optional]: The convention used to compute the exercise fee settlement date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See `Roll Convention` in Table 14.
- An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given for *European* style swaptions, and for *Bermudan* style swaptions at least two `ExerciseDate` date elements must be given.
- Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 8.3.2

8.2.6 FX Forward

The `FXForwardData` node is the trade data container for the *FxForward* trade type. The structure - including example values - of the `FXForwardData` node is shown in Listing 139.

Listing 139: FX Forward data

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

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

- ValueDate: The value date of the FX Forward.
  Allowable values: See `Date` in Table 14.
- BoughtCurrency: The currency to be bought on value date.
  Allowable values: See `Currency` in Table 14.
- 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 14.

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

- SettlementData [Optional]: This node is used to specify the settlement of the cash flows on the value date.

A `SettlementData` node is shown in Listing 140, and the meanings and allowable values of its elements follow below.

- Currency: The currency in which the FX Forward is settled. This field is only used if settlement is *Cash*.
  Allowable values: See `Currency` in Table 14. Defaults to the sold currency if left blank or omitted.

- FXIndex: The FX reference index for determining the FX fixing at the value date. This field is required if settlement is *Cash* and the payment date is greater than the value date. Otherwise, it is ignored.
  Allowable values: The format of the `FXIndex` is "FX-FixingSource-CCY1-CCY2" as described in Table 20.

- Date [Optional]: If specified, this will be the payment date.
  Allowable values: See `Date` in Table 14. If left blank or omitted, defaults to the value date with some adjustments applied from the `Rules` sub-node.

- Rules [Optional]: If `Date` is left blank or omitted, this node will be used to derive the payment date from the value date. The `Rules` sub-node is shown in Listing 140, and the meanings and allowable values of its elements follow below.

  – PaymentLag [Optional]: The lag between the value date and the payment date.
    Allowable values: Any valid period, i.e. a non-negative whole number, followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted.

  – PaymentCalendar [Optional]: The calendar to be used when applying the payment lag.
    Allowable values: See Table 16 `Calendar`. Defaults to *NullCalendar* (no holidays) if left blank or omitted.

  – PaymentConvention [Optional]: The business day convention to be used when applying the payment lag.
    Allowable values: See `Roll Convention` in Table 14. Defaults to *Unadjusted* if left blank or omitted.

Note that FX Forwards also cover Precious Metals forwards, i.e. with currencies XAU, XAG, XPT, XPD.

Listing 140: Example `SettlementData` node with `Rules` sub-node

```
<SettlementData>
  <Currency>USD</Currency>
  <FXIndex>FX-ECB-EUR-USD</FXIndex>
  <Date>2020-09-03</Date>
  <Rules>
    <PaymentLag>2D</PaymentLag>
    <PaymentCalendar>USD</PaymentCalendar>
    <PaymentConvention>Following</PaymentConvention>
  </Rules>
</SettlementData>
```

8.2.7 FX Swap

The `FXSwapData` node is the trade data container for the *FxSwap* trade type. The structure - including example values - of the `FXSwapData` node is shown in Listing 141. It contains no sub-nodes.

Listing 141: 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>
            <Settlement>Cash</Settlement>
        </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 14.
- 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 14.
- 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 14.
- 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 14.

- 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.
- Settlement [Optional]: Delivery type. Note that Non-Deliverable FX Swaps can be represented by *Cash* settlement, and that deliverable FX Swaps will be excluded from the CRIF output. Delivery type does not impact pricing in ORE.

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

Note that FX Swaps also cover Precious Metals Swaps, i.e. with currencies XAU, XAG, XPT, XPD.

8.2.8 FX Option

The `FXOptionData` node is the trade data container for the *FxOption* trade type. FX options with exercise styles *European* or *American* are supported. The `FXOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the FX Option. The structure of an `FXOptionData` node for an FX Option is shown in Listing 142.

Listing 142: FX Option data

```xml
<FxOptionData>
  <OptionData>
    <LongShort>Long</LongShort>
    <OptionType>Call</OptionType>
    <Style>European</Style>
    <Settlement>Cash</Settlement>
    <PayOffAtExpiry>false</PayOffAtExpiry>
    <ExerciseDates>
       <ExerciseDate>2026-03-01</ExerciseDate>
     </ExerciseDates>
     <Premiums>
       <Premium>
         <Amount>10900</Amount>
         <Currency>EUR</Currency>
         <PayDate>2020-03-01</PayDate>
       </Premium>
     </Premiums>
  </OptionData>
  <BoughtCurrency>EUR</BoughtCurrency>
  <BoughtAmount>1000000</BoughtAmount>
  <SoldCurrency>USD</SoldCurrency>
  <SoldAmount>1700000</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. The relevant fields in the `OptionData` node for an FxOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. For option type *Put*, Bought and Sold currencies/amounts are switched compared to the trade data node. For example, a holder of BoughtCurrency EUR SoldCurrency USD FX Call Option has the right to buy EUR using USD, while holder of the Put counterpart has the right to buy USD using EUR, or equivalently sell EUR for USD.
  - `Style` The allowable values are *European* or *American*.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - `PayOffAtExpiry` [Optional] The allowable values are *true* for payoff at expiry, or *false* for payoff at exercise (relevant for *American* style FxOptions). Defaults to *true* if left blank or omitted.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given. For *American* style FxOptions the ExerciseDate represents the Expiry date, i.e. they can be exercised up until this date.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 8.3.2

  See 8.3.1 for further specifications of the `OptionData` node.

- BoughtCurrency: The bought currency of the FX option. See OptionData above for more details.

  Allowable values: See Currency in Table 14.

- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

- SoldCurrency: The sold currency of the FX option. See OptionData above for more details.

  Allowable values: See Currency in Table 14.

- SoldAmount: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

- FXIndex [Optional]: If the option *European*, has cash settlement and is subject to *Automatic Exercise*, as indicated by the `AutomaticExercise` node under `OptionData`, this node must be populated with a valid FX index. The FX index is used to retrieve an FX rate on the expiry date that is in turn used to determine the payoff on the cash settlement date. The payoff is in the `SoldCurrency` i.e. the domestic currency.

  Allowable values: A valid FX index from the Table 20.

Note that FX Options also cover Precious Metals Options, i.e. with currencies XAU, XAG, XPT, XPD.

8.2.9 Equity Option

The `EquityOptionData` node is the trade data container for the *EquityOption* trade type. Equity options with exercise styles *European* and *American* are supported. For a Quanto payoff, only *European* exercise is supported. 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 143.

Listing 143: Equity Option data

```
<EquityOptionData>
    <OptionData>
         <LongShort>Long</LongShort>
         <OptionType>Call</OptionType>
         <Style>American</Style>
         <Settlement>Cash</Settlement>
         <PayOffAtExpiry>true</PayOffAtExpiry>
         <ExerciseDates>
             <ExerciseDate>2022-03-01</ExerciseDate>
         </ExerciseDates>
         ...
    </OptionData>
    <Name>RIC:.SPX</Name>
    <Currency>USD</Currency>
    <Strike>2147.56</Strike>
    <StrikeCurrency>USD</StrikeCurrency>
    <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. The relevant fields in the `OptionData` node for an EquityOption are:
  - `LongShort`: The allowable values are *Long* or *Short*.
  - `OptionType`: The allowable values are *Call* or *Put*. *Call* means that the option holder has the right to buy the given quantity of the underlying equity at the strike price. *Put* means that the option holder has the right to sell the given quantity of the underlying equity at the strike price.
  - `Style`: The allowable values are *European* and *American*. For Quanto payoffs, i.e. if `Curreny` and underlying equity currency are different, this must be set to *European*.
  - `Settlement`: The allowable values are *Cash* or *Physical*. If `Currency` and underlying equity currency are different, i.e. Quanto payoff, this must be set to *Cash*.

  - `PayOffAtExpiry` [Optional]: The allowable values are *true* for payoff at expiry, or *false* for payoff at exercise. This field is relevant for *American* style EquityOptions, and defaults to *true* if left blank or omitted.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 8.3.2

- Name: The identifier of the underlying equity or equity index.
  Allowable values: See `Name` for equity trades in Table 22.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 8.3.20.
- Currency: The payment currency of the equity option.
  Allowable values: See `Currency` and `Minor Currencies` in Table 14. If this is different to the currency that the underlying equity is quoted in, then a Quanto payoff will be applied. Using the corresponding major currency for an equity quoted in the minor currency will not correspond to a Quanto payoff.
- Strike: The option strike price.
  Allowable values: Any positive real number.
- StrikeCurrency [Optional]: The currency that the `Strike` is quoted in. If the option is Quanto, then this field must not be left blank, and must equal the currency that the underlying equity is quoted in, up to the minor/major currency. For example, if the underlying equity is quoted in GBP, then`StrikeCurrency` must be either *GBP* or *GBp*.
  Allowable values: See `Currency` and `Minor Currencies` in Table 14. Must be the major or minor currency of the `Currency` field above.
- Quantity: The number of units of the underlying covered by the transaction.
  Allowable values: Any positive real number.

8.2.10 Equity Futures Option

The `EquityFutreOptionData` node is the trade data container for the *EquityFutureOption* trade type. Equity options with exercise styles *European* and *American* are supported. The `EquityFutureOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the equity future option. The structure of an example `EquityFutureOptionData` node for an equity option is shown in Listing 144.

Listing 144: Equity Future Option data

```
<EquityFutureOptionData>
    <OptionData>
         <LongShort>Long</LongShort>
         <OptionType>Call</OptionType>
         <Style>American</Style>
         <Settlement>Cash</Settlement>
         <PayOffAtExpiry>true</PayOffAtExpiry>
         <ExerciseDates>
             <ExerciseDate>2022-03-01</ExerciseDate>
         </ExerciseDates>
         ...
    </OptionData>
    <Name>RIC:.SPX</Name>
    <Currency>USD</Currency>
    <Strike>2147.56</Strike>
    <Quantity>17000</Quantity>
    <FutureExpiryDate>2021-01-29</FutureExpiryDate>
</EquityFutureOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 8.3.1 Option Data. The relevant fields in the `OptionData` node for an EquityOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. *Call* means that the option holder has the right to buy the given quantity of the underlying equity at the strike price. *Put* means that the option holder has the right to sell the given quantity of the underlying equity at the strike price.
  - `Style` The allowable value is *European*.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - `PayOffAtExpiry` [Optional] The allowable values are *true* for payoff at expiry, or *false* for payoff at exercise. This field is relevant for *American* style EquityOptions, and defaults to *true* if left blank or omitted.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 8.3.2
- Name: The identifier of the underlying equity or equity index.
  Allowable values: See `Name` for equity trades in Table 22.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 8.3.20.

- Currency: The currency of the equity option.
  Allowable values: See `Currency` in Table 14.

- 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.

- FutureExpiryDate [Optional]: If `IsFuturePrice` is `true` and the underlying is a future contract settlement price, this node allows the user to specify the expiry date of the underlying future contract.

  Allowable values: This should be a valid date as outlined in Table 14. If not provided, it is assumed that the future contract's expiry date is equal to the option expiry date provided in the `OptionData` node.

8.2.11 Equity Forward

The `EquityForwardData` node is the trade data container for the *EquityForward* trade type. Vanilla equity forwards are supported. The structure of an example `EquityForwardData` node for an equity forward is shown in Listing 145.

Listing 145: Equity Forward data

```
<EquityForwardData>
  <LongShort>Long</LongShort>
  <Maturity>2018-06-30</Maturity>
  <Name>RIC:.SPX</Name>
  <Currency>USD</Currency>
  <Strike>2147.56</Strike>
  <StrikeCurrency>USD</StrikeCurrency>
  <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 14.

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

  Allowable values: See `Name` for equity trades in Table 22.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 8.3.20.

- Currency: The currency of the equity forward.
  Allowable values: See `Currency` and `Minor Currencies` in Table 14.
- Strike: The agreed buy/sell price of the equity forward.
  Allowable values: Any positive real number.
- StrikeCurrency: [Optional] The currency of the strike value. Defaults to `Currency` field
  Allowable values: See `Currency` and `Minor Currencies` in Table 14.
- Quantity: The number of units of the underlying equity to be bought/sold.
  Allowable values: Any positive real number.

8.2.12 Equity Swap

An Equity Swap uses its own trade type *EquitySwap*, and is set up using a `EquitySwapData` node with one leg of type *Equity* and one more leg that can be either *Fixed* or *Floating*. Listing 146 shows an example. The Equity leg contains an additional `EquityLegData` block. See 8.3.15 for details on the Equity leg specification.

Listing 146: Equity Swap Data

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

If the equity swap has a resetting notional, typically the funding leg's notional will be aligned with the equity leg's notional. To achieve this, indexings on the floating leg can be used, see 8.3.8. In the context of equity swaps the indexings can be defined in a simplified way by adding a Indexings node with a subnode FromAssetLeg set to true to the funding leg's LegData node. The notionals node is not required either in the funding leg's LegData in this case. An example is shown in listing 147.

Listing 147: Equity Swap Data with notional reset and FX indexing

```
<EquitySwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Currency>USD</Currency>
    ...
    <!-- Notionals node is not required, set to 1 internally -->
    ...
    <Indexings>
      <!-- derive the indexing information (equity price, FX) from the Equity leg -->
      <FromAssetLeg>true</FromAssetLeg>
    </Indexings>
  </LegData>
  <LegData>
    <LegType>Equity</LegType>
      <Currency>USD</Currency>
      ...
      <EquityLegData>
        <Quantity>1000</Quantity>
        <Name>RIC:.STOXX50E</Name>
        <InitialPrice>2937.36</InitialPrice>
        <NotionalReset>true</NotionalReset>
        <FXTerms>
          <EquityCurrency>EUR</EquityCurrency>
          <FXIndex>FX-ECB-EUR-USD</FXIndex>
          <FXIndexFixingDays>2</FXIndexFixingDays>
          <FXIndexCalendar>EUR,USD</FXIndexCalendar>
        </FXTerms>
      </EquityLegData>
      ...
  </LegData>
</EquitySwapData>
```

8.2.13 CPI Swap

A CPI swap is set up as a swap, with one leg of type `CPI`. Listing 148 shows an example. The CPI leg contains an additional `CPILegData` block. See 8.3.16 for details on the CPI leg specification.

Listing 148: CPI Swap Data

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

8.2.14 Year on Year Inflation Swap

A Year on Year inflation swap is set up as a swap, with one leg of type `YY`. Listing 149 shows an example. The YY leg contains an additional `YYLegData` block. See 8.3.17 for details on the YY leg specification.

Listing 149: 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>
```

8.2.15 Bond

A Bond is set up using a `BondData` block, and can be both a stand-alone instrument with trade type *Bond*, or a trade component used by multiple bond derivative instruments.

A Bond can be set up in a short version referencing an underlying bond static, or in a long version where the underlying bond details are specified explicitly, including a full LegData block. The short version is shown in listing 150. The details of the bond are read from the reference data in this case using the SecurityId as a key. The bond trade is fully specified by

- SecurityId: The id identifying the bond
  Allowable Values: A valid bond identifier, typically the ISIN of the reference bond with the ISIN: prefix, e.g.: `ISIN:XXNNNNNNNNNN`
- BondNotional: The notional of the position in the reference bond, expressed in the currency of the bond
  Allowable Values: Any non-negative real number
- CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product. If set to *false*, the product class will not be set to *Credit*, and there will be no credit sensitivities.
  Allowable Values: *true* or *false* Defaults to *true* if left blank or omitted.

in this case.

Listing 150: Bond Data

```
<BondData>
  <SecurityId>ISIN:XS0982710740</SecurityId>
  <BondNotional>100000000.0</BondNotional>
  <CreditRisk>true</CreditRisk>
</BondData>
```

For the long version, the bond details are inlined in the trade as shown in listing 151. The bond specific elements are

- IssuerId [Optional]: A text description of the issuer of the bond. This is for informational purposes and not used for pricing.

  Allowable values: Any string. If left blank or omitted, the bond will not have any issuer description.

- CreditCurveId [Optional]: The unique identifier of the bond. This is used for pricing, and is required for bonds for which a credit - related margin component should be generated, and otherwise left blank. If left blank, the bond (and any bond derivatives using the bond as a trade component) will be plain IR rather than a IR/CR.

  Allowable values: A valid bond identifier, typically the ISIN of the reference bond with the ISIN: prefix, e.g.: `ISIN:XXNNNNNNNNNN`

- SecurityId: The unique identifier of the bond. This defines the security specific spread to be used for pricing.

  Allowable values: A valid bond identifier, typically the ISIN of the reference bond with the ISIN: prefix, e.g.: `ISIN:XXNNNNNNNNNN`

- ProxySecurityId [Optional]: Only applicable to exotic bonds, which have the BondData block embedded as one of the components typically. An identifier of a proxy security. If given, the security curve configuration, i.e. the security spread and recovery rate of the proxy security will be used for the pricing of the exotic bond. Typically the ISIN of a liquid vanilla bond of the same issuer and with comparable maturity as the convertible bond. The proxy security must be a vanilla bond.

Allowable values: A valid bond identifier, typically the ISIN of the reference bond with the ISIN: prefix, e.g.: `ISIN:XXNNNNNNNNNN`

- ReferenceCurveId: The benchmark curve to be used for pricing. This is typically the main ibor index for the currency of the bond, and if no ibor index is available for the currency in question, a currency-specific benchmark curve can be used.

  Allowable values: For currencies with available ibor indices:
  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 18.

  For currencies without available ibor indices:
  An alphanumeric string of the form [CCY]BENCHMARK-[CCY]-TERM, matching a benchmark curve set up in the market data configuration.

  Examples: IDRBENCHMARK-IDR-3M, EGPBENCHMARK-EGP-3M, UAHBENCHMARK-UAH-3M, NGNBENCHMARK-NGN-3M

- SettlementDays: The settlement lag in number of business days applicable to the security.

  Allowable values: A non-negative integer.

- Calendar: The calendar associated to the settlement lag.

  Allowable values: See Table 16 Calendar.

- IssueDate: The issue date of the security.

  See `Date` in Table 14.

A LegData block then defines the cashflow structure of the bond, this can be of type fixed, floating etc. Note that a LegData block should only be included in the long version.

Listing 151: Bond Data

```
<BondData>
    <IssuerId>Ineos Group Holdings SA</IssuerId>
    <CreditCurveId>ISIN:XS0982710740</CreditCurveId>
    <SecurityId>ISIN:XS0982710740</SecurityId>
    <ProxySecurityId>ISIN:XS2000000000</ProxySecurityId>
    <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 trade type supports perpetual schedules, i.e. perpetual bonds can be represented by omitting the EndDate in the leg data schedule definition. Only rule based schedules can be used to indicate perpetual schedules.

8.2.16 Forward Bond

A Forward Bond (or Bond Forward) is a contract that establishes an agreement to buy or sell (determined by `LongInForward`) an underlying bond at a future point in time (the `ForwardMaturityDate`) at an agreed price (the settlement `Amount`).

A T-Lock is a Forward Bond with a US Treasury Bond as underlying, whereas a J-Lock is a Forward Bond with a Japanese Government Bond as underlying. T-Locks can be specified in terms of a lock-in yield rather then a settlement amount. The cash settlement amount is given by (bond yield at maturity - lock rate) x DV01 in this case.

Listing 152 shows an example for a physically settled forward bond. Listing 153 shows an example for a cash settled T-Lock transaction specified by a lock-in yield.

A Forward Bond is set up using a `ForwardBondData` block as shown below and the trade type is *ForwardBond*. The specific elements are

- BondData: A `BondData` block specifying the underlying bond as described in section 8.2.15. A long position must be taken in the bond, i.e. (`Payer`) flag must be set to (`true`). The bond data block contains an additional field for forward bonds
  - IncomeCurveId: The benchmark curve to be used for compounding, this must match a name of a curve in the yield curves or index curve block in `todaysmarket.xml`. It is optional to provide this curve. If left out the market reference yield curve from `todaysmarket.xml` is used for compounding.
- SettlementData: The entity defining the terms of settlement:
  - ForwardMaturityDate: The date of maturity of the forward contract.
    Allowable values: See `Date` in Table 14.
  - Settlement [Optional]: Cash or Physical. Option, defaults to Physcial, except in case the settlement is defined by LockRate, in which case it defaults to Cash.
    Allowable values: Cash, Physical
  - Amount [Optional]: The settlement amount (also called strike) transferred at forward maturity in return for the bond (physical delivery) or a cash amount equal to the dirty price of the bond (cash settlement). This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond. Exactly one of the fields Amount, LockRate must be given.
    Allowable values: Any non-negative real number.
  - LockRate [Optional]: The payoff is given by (yield at forward maturity - LockRate) x DV01 (LongInForward = true). Exactly one of the fields Amount, LockRate must be given. In case the LockRate is given, the

Settlement must be set to Cash. If Settlement is not given, it defaults to Cash in this case.
Allowable values: Any non-negative real number.

  - LockRateDayCounter [Optional]: The day counter w.r.t. which the lock rate is expressed. Optional, defaults to A360.
    Allowable values: see table 17

  - SettlementDirty [Optional]: A flag that determines whether the settlement amount (`Amount`) reflects a clean (*false*) or dirty (*true*) price. In either case, the dirty amount is actually paid on the forward maturity date, i.e. if SettlementDirty = *false*, the (forward) accruals are computed internally and added to the given amount to get the actual settlement amount. Optional, defaults to true.
    Allowable values: *true*, *false*

- PremiumData: The entity defining the terms of a potential premium payment. This node is optional. If left out it is assumed that no premium is paid.

  - Date: The date when a premium is paid.
    Allowable values: See `Date` in Table 14.

  - Amount: The amount transferred as a premium. This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond.
    Allowable values: Any non-negative real number.

- LongInForward: A flag that determines whether the forward contract is entered in long (*true*) or short (*false*) position.
  Allowable values: *true*, *false*

Listing 152: Forward Bond Data

```
<ForwardBondData>
  <BondData>
   ...
   <IncomeCurveId>BENCHMARKINCOME-EUR<IncomeCurveId>
  </BondData>
  <SettlementData>
    <ForwardMaturityDate>20160808</ForwardMaturityDate>
    <Settlement>Physcial</Settlement>
    <ForwardSettlementDate>20160810</ForwardSettlementDate>
    <Amount>1000000.00</Amount>
    <SettlementDirty>true</SettlementDirty>
  </SettlementData>
  <PremiumData>
    <Amount>1000.00</Amount>
    <Date>20160808</Date>
  </PremiumData>
  <LongInForward>true</LongInForward>
</ForwardBondData>
```

Listing 153: Forward Bond Date (T-Lock)

```
<ForwardBondData>
  <BondData>
   ...
  </BondData>
  <SettlementData>
    <ForwardMaturityDate>20160808</ForwardMaturityDate>
    <ForwardSettlementDate>20160810</ForwardSettlementDate>
    <LockRate>0.02365</LockRate>
  </SettlementData>
  <LongInForward>true</LongInForward>
</ForwardBondData>
```

As for the ordinary bond the forward bond pricing requires a recovery rate that can be specified in ORE per SecurityId.

Forward Bond - Pricing Engine configuration

The configuration for the pricing engine of the forward bond is identical to the ordinary bond.The pricing engine called by forward bond products is the `DiscountingForwardBondEngine`, see below for a configuration example.

```
<Product type="ForwardBond">
<Model>DiscountedCashflows</Model>
<ModelParameters></ModelParameters>
<Engine>DiscountingForwardBondEngine</Engine>
<EngineParameters>
 <Parameter name="TimestepPeriod">3M</Parameter>
</EngineParameters>
</Product>
```

8.2.17 Credit Default Swap

A credit default swap, trade type `CreditDefaultSwap`, is set up using a `CreditDefaultSwapData` block as shown in listing 154 or 155. The `CreditDefaultSwapData` block must include either a `CreditCurveId` element or a `ReferenceInformation` node. The elements have the following meaning:

- IssuerId [Optional]: An identifier for the reference entity of the CDS. For informational purposes and not used for pricing.
- CreditCurveId: The identifier of the reference entity defining the default curve used for pricing. For the allowable values, see `CreditCurveId` for credit trades - single name in Table 22. A `ReferenceInformation` node may be used in place of this `CreditCurveId` node.
- ReferenceInformation: This node may be used as an alternative to the `CreditCurveId` node to specify the reference entity, tier, currency and documentation clause for the CDS. This in turn defines the credit curve used for

pricing. The `ReferenceInformation` node is described in further detail in Section 8.3.19.

- SettlesAccrual [Optional]: Whether or not the accrued coupon is due in the event of a default. This defaults to `true` if not provided.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 28.

- ProtectionPaymentTime [Optional]: Controls the payment time of protection and premium accrual payments in case of a default event. Defaults to `atDefault`.

  Allowable values: `atDefault`, `atPeriodEnd`, `atMaturity`. Overrides the `PaysAtDefaultTime` node

- PaysAtDefaultTime [Deprecated]: *true* is equivalent to ProtectionPaymentTime = atDefault, *false* to ProtectionPaymentTime = atPeriodEnd. Overridden by the `ProtectionPaymentTime` node if set

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 28.

- ProtectionStart [Optional]: The first date where a credit event will trigger the contract. This defaults to the first date in the schedule if it is not provided. Must be set to a date before or on the first date in the schedule if the `LegData` has a rule that is not one of `CDS` or `CDS2015`. In general, for standard CDS traded after the CDS Big Bang in 2009, the protection start date is equal to the trade date. Therefore, typically the `ProtectionStart` should be set to the trade date of the CDS.

- UpfrontDate [Optional]: Settlement date for the UpfrontFee if an UpfrontFee is provided. If an UpfrontFee is provided and it is non-zero, `UpfrontDate` is required. The `UpfrontDate`, if provided, must be on or after the ProtectionStart date. Typically, it is 3 business days after the CDS contract trade date.

- UpfrontFee [Optional]: The upfront payment, expressed as a percentage in decimal form, to be multiplied by notional amount. If an UpfrontDate is provided, an UpfrontFee must also be provided. The UpfrontFee can be omitted but cannot be left blank. The UpfrontFee can be negative. The UpfrontFee is treated as an amount payable by the protection buyer to the protection seller. A negative value for the UpfrontFee indicates that the UpfrontFee is being paid by the protection seller to the protection buyer.

  Allowable values: Any real number, expressed in decimal form as a percentage of the notional. E.g. an UpfrontFee of *0.045* and a notional of 10M, would imply an upfront fee amount of 450K.

- FixedRecoveryRate [Optional]: This node holds the fixed recovery rate if the CDS is a fixed recovery CDS. For a standard CDS, this field should be omitted.

- TradeDate [Optional]: The CDS trade date. If omitted, the trade date is deduced from the protection start date. If the schedule provided in the `LegData` has a rule that is either `CDS` or `CDS2015`, the trade date is set equal to the protection start date. This is the standard for CDS traded after the CDS Big

Bang in 2009. Otherwise, the trade date is set equal to the protection start date minus 1 day as it was standard before the CDS Big Bang to have protection starting on the day after the trade date.

- CashSettlementDays [Optional]: The number of business days between the trade date and the cash settlement date. For standard CDS, this is 3 business days. If omitted, this defaults to 3.

The `LegData` block then defines the CDS premium leg structure. This premium leg must be be of type `Fixed` as described in Section 8.3.5.

Listing 154: CreditDefaultSwap Data

```
    <CreditDefaultSwapData>
      <IssuerId>CPTY_A</IssuerId>
      <CreditCurveId>RED:008CA0|SNRFOR|USD|MR14</CreditCurveId>
      <SettlesAccrual>Y</SettlesAccrual>
      <ProtectionPaymentTime>atDefault</ProtectionPaymentTime>
      <ProtectionStart>20160206</ProtectionStart>
      <UpfrontDate>20160208</UpfrontDate>
      <UpfrontFee>0.0</UpfrontFee>
      <LegData>
            <LegType>Fixed</LegType>
            <Payer>false</Payer>
            ...
      </LegData>
    </CreditDefaultSwapData>
```

Listing 155: `CreditDefaultSwapData` with `ReferenceInformation`

```
<CreditDefaultSwapData>
  <ReferenceInformation>
    <ReferenceEntityId>RED:008CA0</ReferenceEntityId>
    <Tier>SNRFOR</Tier>
    <Currency>USD</Currency>
    <DocClause>MR14</DocClause>
  </ReferenceInformation>
  <LegData>
    ...
  </LegData>
</CreditDefaultSwapData>
```

8.2.18 Commodity Option

The `CommodityOptionData` node is the trade data container for the *CommodityOption* trade type. Vanilla commodity options are supported. The exercise style may be *European* or *American*. The `CommodityOptionData` node includes exactly one `OptionData` trade component sub-node plus elements specific to the commodity option. The structure of a `CommodityOptionData` node for a commodity option is shown in Listing 156.

Listing 156: Commodity Option data

```
<CommodityOptionData>
  <OptionData>
    ...
  </OptionData>
  <Name>...</Name>
  <Currency>...</Currency>
  <Strike>...</Strike>
  <Quantity>...</Quantity>
  <IsFuturePrice>...<IsFuturePrice>
  <FutureExpiryDate>...<FutureExpiryDate>
</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* or *American* option style. If the option style is *American*, *PayOffAtExpiry* needs to be *false* as payoff at expiry is not currently supported.

- Name: The name of the underlying commodity.
  Allowable values: See `Name` for commodity trades in Table 24.

- Currency: The currency of the commodity option.
  Allowable values: See `Currency` in Table 14.

- 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.

- IsFuturePrice [Optional]: A boolean indicating if the underlying is a future contract settlement price, `true`, or a spot price, `false`.

  Allowable values: A boolean value given in Table 28. If not provided, the default value is `true`.

- FutureExpiryDate [Optional]: If `IsFuturePrice` is `true` and the underlying is a future contract settlement price, this node allows the user to specify the expiry date of the underlying future contract.

  Allowable values: This should be a valid date as outlined in Table 14. If not provided, it is assumed that the future contract's expiry date is equal to the option expiry date provided in the `OptionData` node.

Note that a Precious Metal Option should be represented as an FX Option using the appropriate commodity "currency" (XAU, XAG, XPT, XPD).

8.2.19 Commodity Forward

The `CommodityForwardData` node is the trade data container for the `CommodityForward` trade type. The structure of an example `CommodityForwardData` node is shown in Listings 157 and 158.

Listing 157: Commodity Forward data

```
<CommodityForwardData>
  <Position>Long</Position>
  <Maturity>2018-06-30</Maturity>
  <Name>PM:XAUUSD</Name>
  <Currency>USD</Currency>
  <Strike>1355</Strike>
  <Quantity>1000</Quantity>
  <IsFuturePrice>...</IsFuturePrice>
  <FutureExpiryDate>...</FutureExpiryDate>
  <FutureExpiryOffset>...</FutureExpiryOffset>
  <FutureExpiryOffsetCalendar>...</FutureExpiryOffsetCalendar>
  <PhysicallySettled>...</PhysicallySettled>
  <PaymentDate>...</PaymentDate>
</CommodityForwardData>
```

Listing 158: `CommodityForwardData` for forward on LME Aluminium 3M future.

```
<CommodityForwardData>
  <Position>Long</Position>
  <Maturity>2021-08-16</Maturity>
  <Name>XLME:AH</Name>
  <Currency>USD</Currency>
  <Strike>2160</Strike>
  <Quantity>1000</Quantity>
  <IsFuturePrice>true</IsFuturePrice>
  <FutureExpiryDate>2021-11-16</FutureExpiryDate>
  <PhysicallySettled>true</PhysicallySettled>
</CommodityForwardData>
```

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

- Position: 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 14.
- Name: The name of the underlying commodity.
  Allowable values: See `Name` for commodity trades in Table 24.
- Currency: The currency of the commodity forward.
  Allowable values: See `Currency` in Table 14.

- 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.

- `IsFuturePrice` [Optional]: This should be set to `true` if the forward contract underlying is the settlement price of a commodity future contract. If omitted, it defaults to `false`.
  Allowable values: Any string that evaluates to true or false as outlined in Table 28.

- `FutureExpiryDate` [Optional]: If `IsFuturePrice` is set to `true`, this gives the expiration date of the underlying commodity future contract. If omitted, the expiration date of the underlying commodity future contract is set equal to the value in the `Maturity` node. If `FutureExpiryDate` is provided, it takes precedence over any value provided in the `FutureExpiryOffset` and `FutureExpiryOffsetCalendar` fields.
  Allowable values: Any date string, see `Date` in Table 14.

- `FutureExpiryOffset` [Optional]: If `IsFuturePrice` is set to `true` and `FutureExpiryDate` is not explicitly specified, this gives the offset period that should be applied to the `Maturity` date to generate the underlying commodity future contract expiration date. If omitted, the expiration date of the underlying commodity future contract is set equal to the value in the `Maturity` node.
  Allowable values: Any string that can be parsed as a period e.g. `2D`, `3M`, etc.

- `FutureExpiryOffsetCalendar` [Optional]: If `FutureExpiryOffset` is provided and is being used, this gives the calendar that should be used when generating the underlying commodity future contract expiration date from the `Maturity` date. If omitted, all days are considered good business days when generating the commodity future contract expiration date which is generally not what is desired.
  Allowable values: Any calendar string, see `Calendar` in Table 16.

- `PhysicallySettled` [Optional]: A value of `true` indicates that the forward contract is physically settled e.g. if the underlying is a future contract, that future contract is entered into on the `Maturity` date. A value of `false` indicates that the forward contract is cash settled e.g. if the underlying is a future contract, that future contract settlement price is observed on the `Maturity` date and the net amount due is exchanged on the cash settlement date. If omitted, it defaults to `true`.
  Allowable values: Any string that evaluates to true or false as outlined in Table 28.

- `PaymentDate` [Optional]: If `PhysicallySettled` is set to `false`, this gives the cash settlement date. It must be greater than or equal to the `Maturity` date. If omitted and the forward is cash settled, the `Maturity` date is used.
  Allowable values: Any date string, see `Date` in Table 14.

Note that a Precious Metal Forward should be represented as an FX Forward using the appropriate commodity "currency" (XAU, XAG, XPT, XPD).

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 159.

Listing 159: 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 160.

Listing 160: Option data

```
<OptionData>
  <LongShort>Long</LongShort>
  <OptionType>Call</OptionType>
  <Style>Bermudan</Style>
  <NoticePeriod>5D</NoticePeriod>
  <NoticeCalendar>TARGET</NoticeCalendar>
  <NoticeConvention>F</NoticePeriod>
  <Settlement>Cash</Settlement>
  <SettlementMethod>CollateralizedCashPrice</SettlementMethod>
  <PayOffAtExpiry>true</PayOffAtExpiry>
  <ExerciseFees>
    <ExerciseFee type="Percentage">0.0020</ExerciseFee>
    <ExerciseFee type="Absolute" startDate="2020-04-20">25000</ExerciseFee>
  </ExerciseFees>
  <ExerciseFeeSettlementPeriod>2D</ExerciseFeeSettlementPeriod>
  <ExerciseFeeSettlementConvention>F</ExerciseFeeSettlementConvention>
  <ExerciseFeeSettlementCalendar>TARGET</ExerciseFeeSettlementCalendar>
  <ExerciseDates>
    <ExerciseDate>2019-04-20</ExerciseDate>
    <ExerciseDate>2020-04-20</ExerciseDate>
  </ExerciseDates>
  <Premiums>
    <Premium>
      <Amount>100000</Amount>
      <Currency>EUR</Currency>
      <PayDate>2018-05-07</PayDate>
    </Premium>
  </Premiums>
  <AutomaticExercise>...</AutomaticExercise>
  <ExerciseData>
    <Date>...</Date>
    <Price>...</Price>
  </ExerciseData>
  <PaymentData>...</PaymentData>
</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*. Note that for Swaptions, Callable Swaps, and Index CDS Options setting `LongShort` to *short* makes the `Payer` indicator on the underlying Swap / Index CDS to be set from the perspective of the Counterparty.

  Allowable values: *Long, L* or *Short, S*

- OptionType: Specifies whether it is a call or a put option. Optional for trade types Swaption and CallableSwap.

  Allowable values: *Call* or *Put*

  The meaning of Call and Put values depend on the trade type and asset class of the option, see Table 13.

Asset Class and Trade Type	**Call / Put Specifications**
Equity/ Commodity/Bond Option	*Call*: The right to buy the underlying equity/commodity/bond at the strike price. *Put*: The right to sell the underlying equity/commodity/bond at the strike price.
IR Swaption, CallableSwap, Commodity Swaption	*Call/Put* values are ignored, and the OptionType field is optional. Payer/Receiver swaption is determined by the `Payer` fields in the Leg Data nodes of the underlying swap.
FX Options (all variants, except Touch, Digital, Asian)	*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. Note that barriers are not switched / unaffected.
Index CDS Option	*Call/Put* values are ignored, and the OptionType field is optional. The `Payer` field in the underlying Index CDS leg determines if the option is to buy or sell protection.
Asian FX Options	*Call*: The right to buy/receive the underlying currency at the strike price. *Put*: The right to sell/pay the underlying currency at the strike price.
Digital FX Options	*Call*: The digital payout will occur if the fx rate at the expiry date is above the given strike, *Put*: The digital payout will occur if the fx rate at the expiry date is below the given strike.
FX Single Touch Options	The OptionType should be *Call* for Up and *Put* for Down barrier types - this is used to check whether the barrier has been hit in the past.
FX Double Touch Options	*Call/Put* values are ignored, and the OptionType field is optional.

Table 13: Specification of Option Type Call / Put

- PayoffType [Optional, except for trade types detailed below]: Specifies a detailed payoff type for exotic options. Only applicable to specific trade types as indicated in parentheses:

  Allowable values:

  - *Accumulator, Decumulator* (applies to trade types EquityAccumulator, FxAccumulator, CommodityAccumulator only)
  - *TargetFull, TargetExact, TargetTruncated* (applies to trade types EquityTaRF, FxTaRF, CommodityTaRF only)

  - *BestOfAssetOrCash, WorstOfAssetOrCash, MaxRainbow, MinRainbow* (applies to trade types EquityRainbowOption, FxRainbowOption, CommodityRainbowOption only)
  - *Vanilla, Asian, AverageStrike, LookbackCall, LookbackPut* (applies to trade types EquityBasketOption, FxBasketOption, CommodityBasketOption only)
  - *Asian* (applies to trade types EquityAsianOption, FxAsianOption only)

- Style: The exercise style of the option.

  Allowable values: *European* or *American* or *Bermudan*.

  Note that trade types IR Swaption and CallableSwap can have style *European* or *Bermudan*, but not *American*.

  FX, Equity and Commodity vanilla options can have styles *European* or *American*, but not *Bermudan*.

  Exotic FX, Equity and Commodity options can generally only have style *European*, see each trade type for details.

  Commodity Swaption and Commodity Average Price Options must have style *European*.

  Index CDS Options must have style *European*.

- NoticePeriod [Optional]: The notice period defining the date (relative to the exercise date) on which the exercise decision has to be taken. If not given the notice period defaults to 0D, i.e. the notice date is identical to the exercise date. Only supported for Swaptions and Callable Swaps currently.

- NoticeCalendar [Optional]: The calendar used to compute the notice date from the exercise date. If not given defaults to the null calendar (no holidays, weekends are no holidays either).

- NoticeConvention [Optional]: The convention used to compute the notice date from the exercise date. Defaults to Unadjusted if not given.

- Settlement: Delivery type. Note that Settlement is not required for Asian options.

  Allowable values: *Cash* or *Physical*

- SettlementMethod [Optional]: Specifies the method to calculate the settlement amount for Swaptions and CallableSwaps.

  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 [9] [10]

- 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 *true* if left blank or omitted.

  Allowable values: *true*, *false* .

  Note that for `IndexCreditDefaultSwapOption` PayOffAtExpiry must be set to *false* as only payoff at exercise is supported.

- ExerciseFees [Optional]: This node contains child elements of type ExerciseFee. Similar to a list of notionals (see 8.3.3) the fees can be given either

  - as a list where each entry corresponds to an exercise date and the last entry is used for all remaining exercise dates if there are more exercise dates than exercise fee entries, or
  - using the `startDate` attribute to specify a change in a fee from a certain day on (w.r.t. the exercise date schedule)

  Fees can either be given as an absolute amount or relative to the current notional of the period immediately following the exercise date using the `type` attribute together with specifiers `Absolute` resp. `Percentage`. If not given, the type defaults to `Absolute`.

  If a fee is given as a positive number the option holder has to pay a corresponding amount if they exercise the option. If the fee is negative on the other hand, the option holder receives an amount on the option exercise.

  Only supported for Swaptions and Callable Swaps currently.

- ExerciseFeeSettlementPeriod [Optional]: The settlement lag for exercise fee payments. Defaults to *0D* if not given. This lag is relative to the exercise date (as opposed to the notice date).

  Allowable values: A number followed by *D, W, M, or Y*

- ExerciseFeeSettlementCalendar [Optional]: The calendar used to compute the exercise fee settlement date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either).

  Allowable values: See Table 16 Calendar.

- ExerciseFeeSettlementConvention [Optional]: The convention used to compute the exercise fee settlement date from the exercise date. Defaults to *Unadjusted* if not given.

  Allowable values: See `Roll Convention` in Table 14.

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

- 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.

- Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 8.3.2

- AutomaticExercise [Optional]: Used if the option expiry date is on the current date or in the past, and the payment date is in the future - so that there still is an outstanding cashflow if the option was in the money on the expiry date. In this case, if AutomaticExercise is applied, the FX / Commodity / Equity fixing on the expiry date is used to automatically determine the payoff and thus whether the option was exercised or not.

  Currently, this field is only used for vanilla European cash settled FX, equity and commodity options. It is a boolean flag indicating if Automatic Exercise is applicable for the option trade. A value of *true* indicates that Automatic Exercise is applicable and a value of *false* indicates that it is not.

  Allowable values: A boolean value given in Table 28. If not provided, the default value is *false*.

- ExerciseData [Optional]: Currently, this node is only used for vanilla European cash settled FX, equity and commodity options where *Automatic Exercise* is not applicable. It has the structure shown in Listing 160 i.e. a child `Date` and `Price` node. It is used to supply the price at which an option was exercised and the date of exercise. For a European option, the supplied date clearly has to match the single option `ExerciseDate`. It is needed where the cash settlement date is after the `ExerciseDate`. If this node is not supplied, and the `ExerciseDate` is in the past relative to the valuation date, the option is assumed to have expired unexercised.

  Allowable values: The `Date` node should be a valid date as outlined in Table 14 and the `Price` node should be a valid price as a real number.

- PaymentData [Optional]: Currently, this node is only used for vanilla European cash settled FX, equity and commodity options. It is used to supply the date on which the option is cash settled if it is exercised. There are two methods in which this data may be supplied:

  1. The first method is an explicit list of dates as shown in Listing 161. The `Date` node should be a valid date as outlined in Table 14. Obviously, for European options, there should be exactly one date supplied.

  2. The second method is a set of rules that are used to generate the settlement date relative to either the exercise date of the option or the expiry date of the option. The structure of the `PaymentData` node in this case is given in Listing 162. The optional `RelativeTo` node must be either `Expiry` or `Exercise`. If it is `Expiry`, the expiry date is taken as the base date from which the rules are applied. If it is `Exercise`, the exercise date is taken as the base date from which the rules are applied. These two dates are the

same in the case of a European option. If not provided, Expiry is assumed. The Lag node is a non-negative integer giving the number of days from the base date to the cash settlement date. The Calendar gives the business day calendar for the cash settlement date and should be a valid calendar code as outlined in Table 16. The Convention gives the business day convention for the cash settlement date and should be a valid business day convention as outlined in Table 14.

Listing 161: Dates based PaymentData

```
<PaymentData>
  <Dates>
    <Date>...</Date>
  </Dates>
</PaymentData>
```

Listing 162: Rules based PaymentData

```
<PaymentData>
  <Rules>
    <Lag>...</Lag>
    <Calendar>...</Calendar>
    <Convention>...</Convention>
    <RelativeTo>...</RelativeTo>
  </Rules>
</PaymentData>
```

8.3.2 Premiums

The Premiums node holds data of one or more premiums to be paid. It is used in different trade types, notably in caps / floors (see section 8.2.3) and more generally in the option data component (see section 8.3.1). Listing 163 shows an example for a Premiums data block representing two premiums.

Listing 163: Premiums Node

```
<Premiums>
  <Premium>
    <Amount>1000</Amount>
    <Currency>EUR</Currency>
    <PayDate>2021-01-27</PayDate>
  </Premium>
  <Premium>
    <Amount>5000</Amount>
    <Currency>USD</Currency>
    <PayDate>2023-01-27</PayDate>
  </Premium>
</Premiums>
```

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

- Amount: The amount to be paid
  Allowable values: arbitrary number
- Currency: Currency of the premium to be paid
  Allowable values: See `Currency` in Table 14.
- PayDate: Date of the premium payment.
  Allowable values: See `Date` in Table 14.

We support a deprecated schema to represent a single premium as shown in listing 164 for backwards compatibility. The 3 nodes PremiumAmount, PremiumCurrency, PremiumPayDate can be used on the same level as the new Premiums node to represent a single premium payment. The deprecated and new schema may not be mixed.

Listing 164: Deprecated Single Premium Representation

```
<PremiumAmount>1000</PremiumAmount>
<PremiumCurrency>EUR</PremiumCurrency>
<PremiumPayDate>2021-01-27</PremiumPayDate>
```

8.3.3 Leg Data and Notionals

The `LegData` trade component node is used within the `CapFloorData`, `SwapData` , `SwaptionData` and `EquitySwapData` 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 165.

Listing 165: Leg data

```
            <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, DigitalCMS, DurationAdjustedCMS, CMSSpread, DigitalCMSSpread, Equity, CPI, YY, ZeroCouponFixed*

- 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 14. When `LegType` is *Equity*, `Minor Currencies` in Table 14 are also allowable.

- PaymentConvention: The payment convention of the leg coupons. Note that the calendar used in conjuction with the PaymentConvention and the PaymentLag to determine the payment date is the calendar in the `ScheduleData` node, except for compounding OIS indices where the index calendar is used. If no calendar is set in the `ScheduleData` node (for dates-based schedules the calendar field is optional), the *NullCalendar* is used.

  Allowable values: See `Roll Convention` in Table 14.

- PaymentLag [optional]: The payment lag given as as number of business days, applies to Fixed legs, Equity legs, and Floating legs with Ibor and OIS indices (but not to BMA/SIFMA indices).
  PaymentLag is also not supported for CapFloor Floating legs that have Ibor coupons with sub periods (HasSubPeriods = *true*), nor for CapFloor Floating legs with averaged ON coupons (IsAveraged = *true*).

  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 17.

- 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 166.

Listing 166: 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 167.

Listing 167: 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.

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 168. The `Exchanges` element is typically used in cross-currency swaps and inflation swaps, but can also be used in other trade and leg types. Note that for cross-currency swaps, the `NotionalInitialExchange` must be set to the same value on both legs. The `NotionalFinalExchange` must also be set to the same value on both legs, i.e. *true* on both, or *false* on both.

Allowable values for `NotionalInitialExchange`, `NotionalFinalExchange` and `NotionalAmortizingExchange`: *true, false*. Defaults to *false* if omitted, or if the entire `Exchanges` block is omitted.

Listing 168: 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 169 for an example.

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

  Allowable values: See `Currency` in Table 14.
- 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: A non-negative integer.
- FixingCalendar[Optional]: The calendar associated with the FX Index.

  Allowable values: See Table 16 Calendar. Defaults to the null calendar if left blank or omitted.

Listing 169: 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 8.3.4 Schedule Data and Dates.
- `PaymentDates` [Optional]: This node allows for the specification of a list of explicit payment dates that overrides the payment dates generated by the payment conventions. This is an optional node and is currently only used in commodity trades. See Listing 170 for an example.

Listing 170: Payment dates

```
                <PaymentDates>
                      <PaymentDate>2020-01-15</PaymentDate>
                      <PaymentDate>2021-01-15</PaymentDate>
                      <PaymentDate>2022-01-17</PaymentDate>
                </PaymentDates>
```

- FixedLegData: This trade component sub-node is required if `LegType` is set to *Fixed* It is outlined in section 8.3.5.
- FloatingLegData: This trade component sub-node is required if `LegType` is set to *Floating* It is outlined in section 8.3.6 Floating Leg Data and Spreads.
- CashflowLegData: This trade component sub-node is required if `LegType` is set to *Cashflow*. It is outlined in section 8.3.9.
- CMSLegData: This trade component sub-node is required if `LegType` is set to *CMS*. It is outlined in section 8.3.10.
- DigitalCMSLegData: This trade component sub-node is required if `LegType` is set to *DigitalCMS*. It is outlined in section 8.3.11.
- DurationAdjustedCMSLegData: This trade component sub-node is required if `LegType` is set to *DurationAdjustedCMS*. It is outlined in section 8.3.12.
- CMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *CMSSpread*. It is outlined in section 8.3.13.
- DigitalCMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *DigitalCMSSpread*. It is outlined in section 8.3.14.
- EquityLegData: This trade component sub-node is required if `LegType` is set to *Equity*. It is outlined in section 8.3.15.
- CPILegData: This trade component sub-node is required if `LegType` is set to *CPI*. It is outlined in section 8.3.16.
- YYLegData: This trade component sub-node is required if `LegType` is set to *YY*. It is outlined in section 8.3.17.
- ZeroCouponFixedLegData: This trade component sub-node is required if `LegType` is set to *ZeroCouponFixed*. It is outlined in section 8.3.18.

8.3.4 Schedule Data (Rules and Dates)

The `ScheduleData` trade component node is used within the `LegData` trade component. The Schedule can either be rules based (at least one sub-node Rules exists) or 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 171 and Listing 172, respectively.

Listing 171: Schedule data, rules based

```
<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 172: Schedule data, date based

```
<ScheduleData>
  <Dates>
    <Calendar>NYB</Calendar>
    <Convention>Following</Convention>
    <Tenor>3M</Tenor>
    <EndOfMonth>false</EndOfMonth>
    <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 14.
- EndDate: The schedule end date. This can be omitted to indicate a perpetual schedule. Notice that perpetual schedule are only supported by specific trade types (e.g. Bond).

  Allowable values: See `Date` in Table 14.

- 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. Also used to determine payment dates (except for compounding OIS index legs, which use the index calendar to determine payment dates).

  Allowable values: See Table 16 Calendar.

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

  Allowable values: See `Roll Convention` in Table 14.

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

  Allowable values: See `Roll Convention` in Table 14.

- 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 15 Rule. Defaults to *Forward* if omitted. Cannot be left blank.

- EndOfMonth [Optional]: Specifies whether the date generation rule is different for end of month, so that the last date of each month is generated, regardless of number of days in the month.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 28. 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. For date generation rules *CDS* and *CDS2015*, if given, this overwrites the first date of the schedule that is otherwise built from IMM dates.

  Allowable values: See `Date` in Table 14.

- LastDate [Optional]: Date for final stub period. For date generation rules *CDS* and *CDS2015*, if given, this overwrites the last date of the schedule that is otherwise built from IMM dates.

  Allowable values: See `Date` in Table 14.

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 the accrual schedule dates. Also used to determine payment dates (except for compounding OIS index legs, which use the index calendar), 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 16 Calendar. Defaults to *NullCalendar* if omitted, i.e. no holidays at all, not even on weekends.

- Convention [Optional]: Roll Convention to determine the accrual schedule dates, and also 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 14. 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.

- EndOfMonth [Optional]: Specifies whether the end of month convention is applied when calculating reference periods for irregular periods when the day count convention is ActActICMA and the schedule is dates based.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 28. Defaults to *false* if left blank or 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. Dates must be provided in chronological order. Note that if no calendar and roll convention is given, the specified dates must be given as adjusted dates.

  Allowable values: Each `Date` child element can take the allowable values listed in `Date` in Table 14.

8.3.5 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 173.

Listing 173: Fixed leg data

```
            <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 174.

Listing 174: Fixed leg data with 'dated' rates

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

8.3.6 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.

An example of a `FloatingLegData` node is shown in Listing 175.

Listing 175: Floating leg data

```
                <FloatingLegData>
                    <Index>USD-LIBOR-3M</Index>
                    <IsInArrears>false</IsInArrears>
                    <IsAveraged>false</IsAveraged>
                    <HasSubPeriods>false</HasSubPeriods>
                    <IncludeSpread>false</IncludeSpread>
                    <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>false</NakedOption>
                    <LocalCapFloor>false</LocalCapFloor>
                </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 18. TERM is not required for Overnight indices.

- 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. *false* indicates that the initial index period fixing determines the payment rate for the full tenor, i.e. no further fixings, no averaging and no compounding. IsAveraged is ignored for Ibor legs when HasSubPeriods is set to *false*. HasSubPeriods does not apply to Overnight indices.

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

- IncludeSpread [Optional]: Only applies to Sub Periods and (compounded) OIS 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.

  A Zero Coupon Floating leg with compounding that includes spread can be set up using a rules-based schedule as shown in Listing 176. Note that the `Tenor` in the rules-based schedule is not used when `Rule` is set to *Zero*.

Listing 176: Zero Coupon Floating Leg - Rules-based

```
<LegData>
    <LegType>Floating</LegType>
    <Payer>false</Payer>
    <Currency>USD</Currency>
    <Notionals>
        <Notional>200000.0000</Notional>
    </Notionals>
    <DayCounter>A360</DayCounter>
    <PaymentConvention>MF</PaymentConvention>
    <ScheduleData>
        <Rules>
            <StartDate>2020-01-14</StartDate>
            <EndDate>2020-07-14</EndDate>
            <Tenor>3M</Tenor>
            <Calendar>USD</Calendar>
            <Convention>MF</Convention>
            <TermConvention>MF</TermConvention>
            <Rule>Zero</Rule>
        </Rules>
    </ScheduleData>
    <FloatingLegData>
        <Index>USD-LIBOR-3M</Index>
        <IsAveraged>false</IsAveraged>
        <HasSubPeriods>true</HasSubPeriods>
        <IncludeSpread>true</IncludeSpread>
        <Spreads>
            <Spread>0.006500</Spread>
        </Spreads>
        <IsInArrears>false</IsInArrears>
        <FixingDays>2</FixingDays>
    </FloatingLegData>
</LegData>
```

A Zero Coupon Floating leg with compounding that includes spread can also be set up using a dates-based schedule with two dates (start and end) as shown in Listing 177.

Listing 177: Zero Coupon Floating Leg - Dates-based

```
<LegData>
    <LegType>Floating</LegType>
    <Payer>false</Payer>
    <Currency>USD</Currency>
    <Notionals>
        <Notional>200000.0000</Notional>
    </Notionals>
    <DayCounter>A360</DayCounter>
    <PaymentConvention>MF</PaymentConvention>
    <ScheduleData>
        <Dates>
            <Calendar>USD</Calendar>
            <Convention>MF</Convention>
            <Dates>
                <Date>2020-01-14</Date>
                <Date>2020-07-14</Date>
            </Dates>
        </Dates>
    </ScheduleData>
    <FloatingLegData>
        <Index>USD-LIBOR-3M</Index>
        <IsAveraged>false</IsAveraged>
        <HasSubPeriods>true</HasSubPeriods>
        <IncludeSpread>true</IncludeSpread>
        <Spreads>
            <Spread>0.006500</Spread>
        </Spreads>
        <IsInArrears>false</IsInArrears>
        <FixingDays>2</FixingDays>
    </FloatingLegData>
</LegData>
```

- IsInArrears [Optional]: *true* indicates that fixing is in arrears, *false* indicates that fixing is in advance.
  - For Ibor coupons, "in arrears" means that the fixing gap is calculated in relation to the current period end date, while "in advance" means that the fixing gap is calculated in relation to the period start date.
  - For OIS coupons, "in arrears" means that the compounding (or averaging) of ON rates is done over the current period, while "in advance" means that the compounding (averaging) is done over the previous period. For the first period, a virtual previous period will be constructed based on the schedule construction rules. In the context of RFRs there are two common "in advance" variants:
    * "Last Recent" which means the length of the period used for compounding / averaging is independent of the original period. This former period is specified in the LastRecentPeriod field.
    * "Last Reset" which means the original period will be used for compounding / averaging. This variant is indicated by omitting the LastRecentPeriod field.

Notice that the use of the LastRecentPeriod field is not restricted to "in advance" OIS coupons, i.e. it can also be used in combination with "in arrears".

Allowable values: *true, false*. Defaults to *false* for Ibor and to *true* for OIS coupons, if left blank or omitted.

- LastRecentPeriod [Optional]: Only applies to OIS coupons. If given, the compounding / averaging of ON rates will not be done over the usual reference period derived from the accrual period and the Lookback, FixingDays and IsInArrears parameters, but instead over a period determined by the end date of this usual period and the LastRecentPeriod parameter as [EndDate - LastRecentPeriod, EndDate]. The calendar used to compute EndDate - LastRecentPeriod is the schedule calendar unless a specific LastRecentPeriodCalendar is specified. To represent SOFR 30D, 90D, 180D average indices, the LastRecentPeriodCalendar should be set to NullCalendar, since these averages refer to rolling averages over 30, 90, 180 calendar days.
  Allowable values: any valid period, e.g. 30D, 90D, 180D, 1M, 2M, 6M

- LastRecentPeriodCalendar [Optional]: The calendar used to compute the LastRecentPeriod, see this field for more details. If not given, defaults to the schedule calendar.
  Allowable values: See Table 16 Calendar.

- FixingDays [Optional]: The fixing gap. For Ibor coupons this is the number of business days before the accrual period's *reference* date to observe the index fixing. Here, the accrual period reference date is the accrual start date for an in advanced fixed coupon and the accrual end date for in arrears fixed coupon. For overnight coupons this is the number of business days by which the value dates are shifted into the past to get the fixing observation dates. In the context of RFRs the FixingDays parameter is sometimes also called "obervation lag".

  Allowable values: A non-negative whole number. Defaults to the index's fixing days if blank or omitted. See defaults per index in Table 19.

- Lookback [Optional]: Only applicable to OIS legs. A period by which the value dates schedule of (averaged, compounded) OIS legs is shifted into the past. On top of this the gap defined by the FixingDays is applied to get the final fixing date for an original date in the OIS value dates schedule. In the context of RFRs the Lookback parameter is sometimes also called "shift". With this terminology, first the shift and then the observation lag is applied to get the fixing date for an original value date of an overnight coupon.

  Allowable values: any valid period, e.g. 2D, 3M, 1Y

- RateCutoff [Optional]: Only applicable to OIS legs. The number of fixing dates at the end of the fixing period for which the fixing value is held constant and set to the previous value. Defaults to 0.

  Allowable values: any non-negative whole number

- 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 start dates as shown in Listing 178.

Listing 178: 'Dated' spreads

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

If the entire `<Spreads>` section is omitted, it defaults to a spread of *0%*.

- 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.

  If the entire `<Gearings>` section is omitted, it defaults to a gearing of *1*.

- 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. Caps / Floors are supported for Ibor and compounded / averaged OIS coupons, but not for coupons with subperiods.

  For OIS coupons notice how the gearing g and spread s enter the calculation of the coupon amount A dependent on the IncludeSpread and LocalCapFloor flags and the cap rate C, floor rate F, daily rates f_i, daily accrual fractions τ_i and the coupon accrual fraction τ. Notice that the gearing must be 1 if include spread is set to true for capped / floored coupons. The cases for compounded coupons are:

  - IncludeSpread = false, LocalCapFloor = false:

$$A = \min\left(\max\left(g \cdot \frac{\prod(1+\tau_i f_i) - 1}{\tau} + s, F\right), C\right)$$

  - IncludeSpread = true, LocalCapFloor = false:

$$A = \min\left(\max\left(\cdot \frac{\prod(1+\tau_i(f_i + s)) - 1}{\tau}, F\right), C\right)$$

- IncludeSpread = false, LocalCapFloor = true:

$$A = g \cdot \frac{\prod(1 + \tau_i \min(\max(f_i, F), C)) - 1}{\tau} + s$$

- IncludeSpread = true, LocalCapFloor = true:

$$A = \cdot \frac{\prod(1 + \tau_i \min(\max(f_i + s, F), C)) - 1}{\tau}$$

The cases for Averaged coupons are:

- IncludeSpread = true or false, LocalCapFloor = false:

$$A = \min\left(\max\left(\cdot \frac{\sum(\tau_i f_i)}{\tau} + s, F\right), C\right)$$

- IncludeSpread = false, LocalCapFloor = true:

$$A = g \cdot \frac{\sum(\tau_i \min(\max(f_i, F), C))}{\tau} + s$$

- IncludeSpread = true, LocalCapFloor = true:

$$A = g \cdot \frac{\sum(\tau_i \min(\max(f_i + s, F), C))}{\tau}$$

- 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, if *true* the leg represents only the embedded floor, cap or collar. By convention the embedded floor (or cap) are considered long if the leg is a receiver leg, otherwise short. For a collar the floor is long and the cap is short if the leg is a receiver leg. Notice that this is opposite to the definition of a collar in 8.2.3.

  Allowable values: *true*, *false* . Defaults to *false* if left blank or omitted.
- LocalCapFloor [Optional]: Optional node, if *true* a cap (floor) will be applied to the daily rates of a compounded / averaged overnight coupon. If *false* the effective period rate will be capped (floored). The flag is ignored for coupons other than overnight coupons.

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

8.3.7 Leg Data with Amortisation Structures

Amortisation structures can (optionally) be added to a leg as indicated in the following listing 179, within a block of information enclosed by `<Amortizations>` and `</Amortizations>` tags. Note that `<Amortizations>` structures are not supported for trade type CapFloor.

Listing 179: 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`.

8.3.8 Indexings

This trade component can be used as an optional node within the `LegData` component to scale the notional of the coupons of a leg by one or several index prices. This feature is typically used within equity swaps with notional reset to align the notional of the funding leg with the one from the equity leg for all return periods. See 8.2.12 for the specific usage in equity swaps. Notice that typically it is sufficient to set the `FromAssetLeg` flag to *true* in the `Indexings` node definition, i.e.

```
<LegData>
    <LegType>Floating</LegType>
    <!-- no notionals node required -->
    <ScheduleData> ... </ScheduleData>
    <Indexings>
        <FromAssetLeg>true</FromAssetLeg>
    </Indexings>
    <FloatingLegData> ... </FloatingLegData>
</LegData>
```

which will cause the trade builder to pull all the indexing details from the asset leg (the equity leg in an equity swap) and populate the indexing data on the funding leg accordingly. Notice that no definition of a `Notionals` node is required in this case, this will also generated automatically.

In what follows we will describe the full syntax of the Indexings node below for reference. The Indexing component can be used in combination with the following leg types:

- Fixed
- Floating
- CMS
- DigitalCMS
- CMSSpread
- DigitalCMSSpread

If specified the notional of the single coupons in the leg is scaled by one or several index prices and a quantity. The indices can be equity or FX indices. Notice that if notional exchanges are enabled on a leg with `Indexings` defined, the notional exchanges are *not* influenced by the indexing definitions. In general we assume that notional exchanges are not enabled in combination with indexings, but it is not forbidden technically. Listing 180 shows an example of an Floating leg indexed by both an equity price and a FX rate.

Listing 180: Indexings node

```
<LegData>
    <LegType>Floating</LegType>
    <Notionals> ... </Notionals>
    <ScheduleData> ... </ScheduleData>
    <Indexings>
        <FromAssetLeg>false</FromAssetLeg>
        <Indexing>
            <Quantity>1000</Quantity>
            <Index>EQ-RIC:.STOXX50E</Index>
            <InitialFixing>2937.36</InitialFixing>
            <ValuationSchedule>
              <Dates>...</Dates>
              <Rules>...</Rules>
            </ValuationSchedule>
            <FixingDays>0</FixingDays>
            <FixingCalendar/>
            <FixingConvention>U</FixingConvention>
            <IsInArrears>false</IsInArrears>
        </Indexing>
        <Indexing>
            <Index>FX-ECB-EUR-USD</Index>
            <IndexFixingDays>2</IndexFixingDays>
            <IndexFixingCalendar>EUR,USD</IndexFixingCalendar>
            <InitialFixing>1.1469</InitialFixing>
            <ValuationSchedule> ... </ValuationSchedule>
            <FixingDays>0</FixingDays>
            <FixingCalendar/>
            <FixingConvention>U</FixingConvention>
            <IsInArrears>false</IsInArrears>
        </Indexing>
    </Indexings>
    <FloatingLegData> ... </FloatingLegData>
</LegData>
```

The Indexings node contains the following elements:

- `FromAssetLeg` [Optional]: If *true*, and the trade type supports this, the notionals on the funding leg (i.e. the leg with the `FromAssetLeg` field) will be derived from the respective asset leg. Internally, the trade builder will add `Indexing` blocks automatically reflecting the necessary indexings (equity price and FX in the case of an equity swap) from the notional reset feature of the asset leg. Also, the Notionals node of the funding leg will internally be set to a single notional 1. The actual Notionals node in the XML on the funding leg is not required and can be omitted.

  `FromAssetLeg` is supported for the following trade types:

  – `EquitySwap`: Setting `FromAssetLeg` to *true*, aligns the notionals for all return periods on the non-equity funding leg, to the equity leg by deriving equity price, quantity and FX from the equity leg.
  Note that `FromAssetLeg` is only supported if `NotionalReset` is *true* on the equity leg - `FromAssetLeg` is ignored otherwise.

  – `BondTRS`: Setting `FromAssetLeg` to *true*, aligns the notionals for all return periods on the funding leg (in the `FundingData` block), to the total return leg (in the `TotalReturnData` block) by deriving bond price, bond notional and FX from the total return leg, bond data and the reference bond.

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

- `Indexing` [Optional, an arbitrary number can be given]: Each Indexing node describes one indexing as follows:
  - Quantity [Optional]: The quantity that applies. For equity that should be the number of shares, for FX it should be 1, i.e. for FX this field can be omitted. The notional of each coupon is in general determined as Original Coupon Notional x Quantity x Equity Price x FX Rate depending on which indexing types are given. Typically, the original coupon notional will be set to 1.

    Allowable values: Any number. Defaults to *1* if left blank or omitted.
  - Index: The relevant index. This is either an equity or FX index. For an FX index, one of the currencies of the index must match the leg currency. It is then ensured that the FX conversion is applied using the correct direction, i.e. if the foreign currency of the index matches the leg currency, the reciprocals of the index fixings are used as a multiplier.

    Allowable values: This is “FX-SOURCE-CCY1-CCY” for FX, see 8.3.20 and 20 for details, or “EQ-NAME” for Equity with “Name” being the general string representation for equity underlyings IdentifierType:Name:Currency:Exchange, see 8.3.20.
  - IndexFixingDays [Optional]: fixing days of the index, this is only used for FX indices

    Allowable values: Non-negative whole numbers. Defaults to *0* if left blank or omitted.
  - IndexFixingCalendar [Optional]: Fixing calendar of the index, this is only used for FX indices and Bond indices

    Allowable values: See Table 16 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.
  - Dirty [Optional]: Only used for bond indices. Indicates whether to use dirty (*true*) or clean (*false*) prices.

    Allowable values: *true*, *false*. Defaults to *true* if left blank or omitted.
  - Relative [Optional]: Only used for bond indices. Indicates whether to use relative (*true*) or absolute prices (*false*). The absolute price is the dirty or clean npv as of the settlement date of the bond in absolute “dollar” terms using the bond details (in particular the notional) from the reference data. The relative price is the absolute price divided by the current notional as of the settlement date.

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

– ConditionalOnSurvival [Optional]: Only used for bond indices. Indicates whether to forecast bond prices conditional on survival (*true*) or including the default probability from today until the fixing date (*false*).

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

– InitialFixing [Optional]: If given the index fixing value to apply on the fixing date of the first coupon. If not given the value is read from the relevant fixing history.

  Allowable values: any number

– ValuationSchedule [Optional]: If given the schedule from which the fixing dates are deduced. If not given, it defaults to the original leg's schedule.

  If the valuation schedule has the same size as the original leg's schedule, it is assumed that the periods correspond one to one, i.e. the ith fixing date is derived from the ith (inArrears = false) or $i + 1$th (inArrears = true) date in the valuation schedule using the FixingDays, FixingCalendar and FixingConvention.

  If the valuation schedule has a different size than the original leg's schedule, the relevant valuation date for the ith original leg's coupon is determined as the latest valuation date that is less or equal to accrual start date (inArrears = false) resp. accrual end date (inArrears = true) of that coupon. The fixing date is derived from the relevant valuation date as above, i.e. using the FixingDays, FixingCalendar and FixingConvention.

  Allowable values: a valid schedule definition, see 8.3.4

– FixingDays [Optional]: If given defines the number of fixing days to apply when deriving the fixing dates from the valuation schedule (see above).

  Allowable values: Any non-negative whole number. Defaults to *0* if left blank or omitted.

– FixingCalendar [Optional, defaults to NullCalendar (no holidays): If given defines the fixing calendar to use when deriving the fixing dates from the valuation schedule (see above).

  Allowable values: Allowable values: See Table 16 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

– FixingConvention [Optional]: If given defines the business day convention to use when deriving the fixing dates from the valuation schedule (see above). Defaults to *Preceding* if left blank or omitted.

  Allowable values: Any valid business day convention, e.g. (*F, MF, P, MP, U*). See `Roll Convention` in Table 14.

– IsInArrears [Optional]: If *true*, the fixing dates are derived from the period end dates, otherwise from the period start dates as described for `ValuationSchedule` above.

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

8.3.9 Cashflow Leg Data

A Cashflow leg is used to represent one or more custom cashflows, with specified dates and amounts. Listing 181 shows an example for a leg of type Cashflow.

Listing 181: Cashflow leg data

```
<LegData>
  <Payer>false</Payer>
  <LegType>Cashflow</LegType>
  <Currency>EUR</Currency>
  <PaymentConvention>ModifiedFollowing</PaymentConvention>
  <CashflowData>
    <Cashflow>
      <Amount date="2024-12-15">105000</Amount>
    </Cashflow>
  </CashflowData>
</LegData>
```

The CashflowData block contains the following elements:

- Cashflow: This node contains child elements of type `Amount`, each representing a cashflow. Each child element should include the date of the cashflow using the form:

  ```
  <Amount date="YYYY-MM-DD">[amount]</Amount>
  ```

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

8.3.10 CMS Leg Data

Listing 182 shows an example for a leg of type CMS.

Listing 182: CMS leg data

```
<LegData>
  <LegType>CMS</LegType>
  <Payer>false</Payer>
  <Currency>EUR</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>
    <NakedOption>false</NakedOption>
  </CMSLegData>
</LegData>
```

The CMSLegData block contains the following elements:

- Index: The underlying CMS index.

  Allowable values: See 18, a string on the form CCY-CMS-TENOR, where the CMS part stays constant and TENOR is an integer followed by Y.

- Spreads [Optional]: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.

  Allowable values: Each child spread 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.

- 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. Defaults to the index's fixing gap.

  Allowable values: A non-negative whole number. Defaults to the fixing days of the Ibor index the swap references if blank or omitted. See defaults per index in

Table 19.

- 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.

  If the entire `<Gearings>` section is omitted, it defaults to a gearing of *1*.

- 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]: If *true* the leg represents only the embedded floor, cap or collar. By convention the embedded floor (or cap) are considered long if the leg is a receiver leg, otherwise short. For a collar the floor is long and the cap is short if the leg is a receiver leg.

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

8.3.11 Digital CMS Leg Data

Listing 183 shows an example for a leg of type *DigitalCMS*.

Listing 183: Digital CMS leg data

```
<LegData>
  <LegType>DigitalCMS</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <DigitalCMSLegData>
    <CMSLegData>
      <Index>EUR-CMS-10Y</Index>
      <FixingDays>2</FixingDays>
      <Gearings>
        <Gearing>3.0</Gearing>
      </Gearings>
      <Spreads>
        <Spread>0.0010</Spread>
      </Spreads>
      <NakedOption>false</NakedOption>
    </CMSLegData>
    <CallPosition>Long</CallPosition>
    <IsCallATMIncluded>false</IsCallATMIncluded>
    <CallStrikes>
      <Strike>0.003</Strike>
    </CallStrikes>
    <CallPayoffs>
      <Payoff>0.003</Payoff>
    </CallPayoffs>
    <PutPosition>Short</PutPosition>
    <IsPutATMIncluded>false</IsPutATMIncluded>
    <PutStrikes>
      <Strike>0.05</Strike>
    </PutStrikes>
    <PutPayoffs>
      <Payoff>0.05</Payoff>
    </PutPayoffs>
  </DigitalCMSLegData>
</LegData>
```

The `DigitalCMSLegData` block contains the following elements:

- CMSLegData: a `CMSLegData` block describing the underlying Digital CMS leg (see 8.3.10). Caps and floors in the underlying CMS leg are not supported for Digital CMS Options. The `NakedOption` flag in the `CMSLegData` block is supported and can be used to separate the digital option payoff from the underlying CMS coupon.
- 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

8.3.12 Duration Adjusted CMS Leg Data

Listing 184 shows an example for a leg of type DurationAdjustedCMS.

Listing 184: Duration Adjusted CMS leg data

```
<LegData>
  <LegType>DurationAdjustedCMS</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>21000000</Notional>
  </Notionals>
  <DayCounter>Simple</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <PaymentLag>0</PaymentLag>
  <PaymentCalendar>TARGET</PaymentCalendar>
  <DurationAdjustedCMSLegData>
    <Index>EUR-CMS-20Y</Index>
    <Duration>10</Duration>
    <Spreads>
      <Spread>0</Spread>
    </Spreads>
    <Gearings>
      <Gearing>1</Gearing>
    </Gearings>
    <IsInArrears>false</IsInArrears>
    <FixingDays>2</FixingDays>
    <Caps>
      <Cap>0.015</Cap>
    </Caps>
    <NakedOption>true</NakedOption>
  </DurationAdjustedCMSLegData>
  <ScheduleData>
    <Rules>
      <StartDate>2019-12-31</StartDate>
      <EndDate>2023-12-31</EndDate>
      <Tenor>3M</Tenor>
      <Calendar>EUR</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Backward</Rule>
    </Rules>
  </ScheduleData>
</LegData>
```

The DurationAdjustedCMSLegData is identical to the CMSLegData block (see 8.3.10). In addition to this it contains a field defining the duration adjustment:

- Duration [Optional]: A non-negative whole number n that defines the duration adjustment for the coupons. If γ is the underlying CMS index fixing for a coupon the duration adjustment δ is defined as

$$\delta = \sum_{i=1}^{n} \frac{1}{(1+\gamma)^i}$$

  If n is zero or the duration is not given, δ is defined as 1, i.e. no adjustment is applied in this case and the coupon will be an ordinary CMS coupon. The coupon amount A for a duration adjusted coupon with a spread s, a gearing g, a cap c, a floor f and with nominal N and accrual fraction τ is given by:

$$A = \delta \cdot N \cdot \tau \cdot \max(\min(g \cdot \gamma + s, c), f)$$

Allowable values: A non-negative whole number.

8.3.13 CMS Spread Leg Data

Listing 185 shows an example for a leg of type CMSSpread.

Listing 185: 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>
    <NakedOption>false</NakedOption>
  </CMSSpreadLegData>
</LegData>
```

The elements of the CMSSpreadLegData block are identical to those of the CMSLegData (see 8.3.10), except for the index which is defined by two CMS indices as the difference between `Index1` and `Index2`.

8.3.14 Digital CMS Spread Leg Data

Listing 186 shows an example for a leg of type *DigitalCMSSpread*.

Listing 186: Digital CMS Spread leg data

```
      <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>false</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 `DigitalCMSSpreadLegData` block contains the following elements:

- CMSSpreadLegData: a `CMSSpreadLegData` block describing the underlying Digital CMS Spread leg (see 8.3.13). Caps and floors in the underlying CMS Spread leg are not supported for Digital CMS Spread Options. The `NakedOption` flag in the `CMSSpreadLegData` block is supported and can be used to separate the digital option payoff from the underlying CMS Spread coupon.
- 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

8.3.15 Equity Leg Data

Listing 187 shows an example of a leg of type Equity. Note that a resetting Equity Leg (NotionalReset set to *true*) must have either:
a) a Quantity, or
b) an InitialPrice and a Notional in the leg

The EquityLegData block contains the following elements:

- Quantity[Optional]: The number of shares. Either a notional or the quantity must be given for the leg, but not both at the same time.

  Allowable values: Any positive real number

- ReturnType: *Price* indicates that the coupons on the equity leg are determined by the price movement of the underlying equity, i.e.:
  $Notional \cdot \frac{FinalPrice - InitialPrice}{InitialPrice}$,
  whereas *Total* indicates that coupons are determined by the total return of the underlying equity including dividends, i.e.:
  $Notional \cdot \frac{(FinalPrice + dividends * DividendFactor) - InitialPrice}{InitialPrice}$

  Allowable values: *Price*, *Total*

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

  Allowable values: See `Name` for equity trades in Table 23.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 8.3.20.

- InitialPrice [Optional]: Initial Price of the equity, if not present, the first valuation date is used to determine the initial price. If InitialPrice is zero then each coupon's price is just the discounted fixing from the coupon's FixingEndDate. For any divisions we assume the value is one, i.e. when NotionalReset = true we have instead Quantity = Notional. The Initial price can be either given in the currency of the equity or in the leg currency, see InitialPriceCurrency.

  Allowable values: Any positive real number

- InitialPriceCurrency [Optional]: If an initial price is given, it can be either given in the original equity ccy or the leg currency (if these are different). This field determines in which currency the initial price is given. If omitted, it is assumed that the initial price is given in equity currency.

  Allowable values: A valid currency code, See `Currency` and `Minor Currencies` in Table 14.

- NotionalReset [Optional]: Defaults to *true*. Notional resets only affect the equity leg. If NotionalReset is set to *true* the quantity or number of shares of the underlying equity is fixed for all the coupons on the equity leg and the Notional for a period is computed as

  Notional = Quantity x (share price at valuation date for period) x (FX conversion rate at valuation date for period)

  Notice that either a) the Quantity or b) a Notional and an explicit InitialPrice must be given in the leg data for a resettable leg. In the latter case the Quantity is computed as

  Quantity = Notional / InitialPrice

  No FX conversion is allowed if the Quantity has to be derived from the Notional and the InitialPrice.

  If NotionalReset is set to *false* the quantity of the underlying equity varies per period, as per:

  Quantity = Notional / (Equity Price at valuation date for the period)

  For the first period, the InitialPrice is the Equity Price at valuation date. Here, the Notional is taken to be the Notional specified in the leg or - if the Quantity is given - to be

  Notional = Quantity x InitialPrice

  where again the InitialPrice must be explicitly given in the leg data and no FX conversion is allowed in this case.

  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: $0 <$ 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 8.3.4). Note that the number of dates (and periods) in the `ValuationSchedule` must be the same as in the `ScheduleData`. 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*.

Allowable values: Any non-negative integer.

- FXTerms [Mandatory when leg and equity currencies differ]: For the case when the currency the underlying equity is quoted in, is different from the leg currency. The `FXTerm` node contains the following elements:
  - EquityCurrency [Mandatory within `FXTerms`]: Currency underlying equity is quoted in. Required if FXTerms is present.

    Allowable values: See `Currency` and `Minor Currencies` in Table 14.
  - FXIndex [Mandatory within `FXTerms`]: Name of the index for FX fixings for the leg vs equity currency pair, e.g. FX-TR20H-EUR-USD for Thomson Reuters 20:00 EURUSD FX fixing. Required if FXTerms present.

    Allowable values: See Table 20
  - FXIndexFixingDays [Optional]: Number of fixing days for FX Index, defaults to *0*.

    Allowable values: Any non-negative integer.
  - FXIndexCalendar [Optional]: The fixing calendar for the FX Index. Typically the union of the equity leg currency calendar and the underlying equity currency calendar.

    Allowable values: See Table 16 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

Listing 187: Equity leg data

```
<LegData>
  <LegType>Equity</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <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>
    <Quantity>1000.0</Quantity>
    <ReturnType>Price</ReturnType>
    <Underlying>
      <Type>Equity</Type>
      <Name>.SPX</Name>
      <IdentifierType>RIC</IdentifierType>
    </Underlying>
    <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>
     <FXTerms>
       <EquityCurrency>USD</EquityCurrency>
       <FXIndex>FX-TR20H-EUR-USD</FXIndex>
       <FXIndexFixingDays>2</FXIndexFixingDays>
       <FXIndexCalendar>TARGET,USD</FXIndexCalendar>
     <FXTerms>
  </EquityLegData>
</LegData>
```

8.3.16 CPI Leg Data

A CPI leg contains a series of CPI-linked coupon payments $(I(t)/I_0)\,N\,c\,\delta$ and, if `NotionalFinalExchange` is set to *true*, a final inflation-linked redemption $(I(t)/I_0)\,N$. Each coupon and the final redemption can be subtracting the (un-inflated) notional N, i.e. $(I(t)/I_0 - 1)\,N$, see below.

Note that CPI legs with just a final redemption and no coupons, can be set up with a dates-based Schedule containing just a single date - representing the date of the final redemption flow. In this case `NotionalFinalExchange` must be set to *true*, otherwise the whole leg is empty, and the Rate is not used and can be set to any value.

Listing 188 shows an example for a leg of type CPI with annual coupons, and 189 shows an example for a leg of type CPI with just the final redemption.

The `CPILegData` block contains the following elements:

- Index: The underlying zero inflation index.

  Allowable values: See `Inflation CPI Index` in Table 21.

- 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 value I_0 used to determine the lifting factor for the fixed coupons.

  Allowable values: Any positive real number.

- StartDate [Optional]: The start date needs to be provided in case the schedule comprises only a single date. If the schedule has at least two dates and a start date is given at the same time, the first schedule date is taken as the start date and the supplied `StartDate` is ignored.

  Allowable values: See `Date` in Table 14.

- ObservationLag: The observation lag to be applied. It's the amount of time from the fixing at the start or end of the period, moving backward in time, to the inflation index observation date (the inflation fixing).

  Allowable values: An integer followed by *D*, *W*, *M* or *Y*. Interpolation lags are typically expressed in a positive number of *M*, months. Note that negative values are allowed, but mean that the inflation is observed forward in time from the period start/end date, which is unusual.

- Interpolation: The type of interpolation that is applied to inflation fixings. *Linear* interpolation means that the inflation fixing for a given date is interpolated linearly between the surrounding - usually monthly - actual fixings, whereas with *Flat* interpolation the inflation fixings are constant for each day at the value of the previous/latest actual fixing (flat forward interpolation).

  Allowable values: *Linear, Flat*

- 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*: $N\,(I(T)/I_0 - 1)$.
  Final notional payment if *false*: $N\,I(T)/I_0$

- SubtractInflationNotionalAllCoupons [Optional]: A flag indicating whether the non-inflation adjusted notional amount should be subtracted from all coupons. Note that the final redemption payment is not affected by this flag.
  Coupon payment if *true*: $N\,(I(T)/I_0 - 1)$.
  Coupon payment if *false*: $N\,I(T)/I_0$

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 28.
  Defaults to *false* if left blank or omitted.

- Caps [Optional]: This node contains child elements of type `Cap` indicating that the inflation indexed payment is capped; the cap is applied to the inflation index and expressed as an inflation rate, see CPI Cap/Floor in the Product Description. If the cap is constant over the life of the cpi leg, only one cap value should be entered. If two or more coupons have different caps, multiple cap values are required, each represented by a `Cap` child element. The first cap value corresponds to the first coupon, the second cap value corresponds to the second coupon, etc. If the number of coupons exceeds the number of cap values, the cap will be kept at at the value of last entered spread for the remaining coupons. The number of entered cap values cannot exceed the number of coupons. Notice that the caps defined under this node only apply to the cpi coupons, but not a final notional flow (if present). A cap for the final notional flow can be defined under the FinalFlowCap node.

  Allowable values: Each child element can take any real number. The cap is expressed in decimal form, e.g. 0.03 is a cap of 3%.

- Floors [Optional]: This node contains child elements of type `Floor` indicating that the inflation indexed payment is floored; the floor is applied to the inflation index and expressed as an inflation rate. The mode of specification is analogous to caps, see above. Notice that the floors defined under this node only apply to the cpi coupons, but not a final notional flow (if present). A floor for the final notional flow can be defined under the FinalFlowFloor node.

  Allowable values: Each child element can take any real number. The floor is expressed in decimal form, e.g. 0.01 is a cap of 1%.

- FinalFlowCap [Optional]: The cap to be applied to the final notional flow of the cpi leg. If not given, no cap is applied.

  Allowable values: A real number.

- FinalFlowFloor [Optional]: The floor to be applied to the final notional flow of the cpi leg. If not given, no floor is applied.

  Allowable values: A real number.

- NakedOption [Optional]: Optional node, if *true* 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.

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

Whether the leg cotains a final redemption flow at all or not depends on the notional exchange setting, see section 8.3.3 and listing 168.

Listing 188: CPI leg data with capped annual coupons

```
<LegData>
  <LegType>CPI</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
    <Exchanges>
      <NotionalInitialExchange>false</NotionalInitialExchange>
      <NotionalFinalExchange>true</NotionalFinalExchange>
    </Exchanges>
  </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>
    <StartDate>2016-07-18</StartDate>
    <ObservationLag>2M</ObservationLag>
    <Interpolation>Linear</Interpolation>
    <Caps>
       <Cap>0.03</Cap>
    </Caps>
    <Floors>
      <Floor>0.0</Floor>
    <Floors>
    <FinalFlowCap>0.03</FinalFlowCap>
    <FinalFlowFloor>0.0</FinalFlowFloor>
    <NakedOption>false</NakedOption>
    <SubtractInflationNotionalAllCoupons>false</SubtractInflationNotionalAllCoupons>
  </CPILegData>
</LegData>
```

Listing 189: CPI leg data with just the final redemption

```
      <LegData>
        <Payer>false</Payer>
        <LegType>CPI</LegType>
        <Currency>GBP</Currency>
        <PaymentConvention>ModifiedFollowing</PaymentConvention>
        <DayCounter>ActActISDA</DayCounter>
        <Notionals>
          <Notional>25000000.0</Notional>
          <Exchanges>
            <NotionalInitialExchange>false</NotionalInitialExchange>
            <NotionalFinalExchange>true</NotionalFinalExchange>
          </Exchanges>
        </Notionals>
        <ScheduleData>
          <Dates>
            <Calendar>GBP</Calendar>
            <Dates>
              <Date>2020-08-17</Date>
            </Dates>
          </Dates>
        </ScheduleData>
        <CPILegData>
          <Index>UKRPI</Index>
          <Rates>
            <Rate>1.0</Rate>
          </Rates>
          <BaseCPI>280.64</BaseCPI>
          <StartDate>2018-08-19</StartDate>
          <ObservationLag>2M</ObservationLag>
          <Interpolated>false</Interpolated>
          <SubtractInflationNotional>true</SubtractInflationNotional>
          <SubtractInflationNotionalAllCoupons>false</SubtractInflationNotionalAllCoupons>
        </CPILegData>
      </LegData>
```

8.3.17 YY Leg Data

Listing 190 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.

- Spreads [Optional]: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.

- 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).
- 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).
- 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.
- AddInflationNotional [Optional]: If true, the payoff will include the notional of the coupon $N\,\tau\,\frac{I_t}{I_{t-1Y}}$.
- IrregularYoY [Optional]: If true, instead of using a YoY inflation rate the coupon is based on the inflation rate during the actual coupon period, e.g. for a 6M coupon the inflation rate will be computed as $\frac{I_t}{I_{t-6m}} - 1$.

Listing 190: YY leg data

```
<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>
    <Spreads>
      <Spread>0.0010</Spread>
    </Spreads>
    <Gearings>
      <Gearing>2.0</Gearing>
    </Gearings>
    <Caps>
      <Cap>0.05</Cap>
    </Caps>
    <Floors>
      <Floor>0.01</Floor>
    </Floors>
    <NakedOption>N</NakedOption>
    <AddInflationNotional>false</AddInflationNotional>
                              <IrregularYoY>false</IrregularYoY>
  </YYLegData>
</LegData>
```

8.3.18 ZeroCouponFixed Leg Data

A Zero Coupon Fixed leg contains a series of Zero Coupon payments, i.e $(1+r)^t\,N$ for a compounded coupon. The uninflated notional N can be subtracted from the payment , i.e $\left((1+r)^t - 1\right)N$, see `SubtractNotional` below.

Listing 191 shows an example for a leg of type Zero Coupon Fixed.

To create a leg with only one payment, the schedule must only contain the start and end date. Not that this can be achieved by setting the *Tenor* to *0D* in a rule based Schedule.

- 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.

  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. $(1 + r)^t$

- SubtractNotional [Optional]: Decides whether the notional is subtracted from the compounding factor, i.e. $(1 + r * t) - 1$ respectively $(1 + r)^t - 1$, or not, i.e. $(1 + r * t)$ respectively $(1 + r)^t$

  Allowable values: *true*, *Y* or *false*, *N*, defaults to *true* if left blank or omitted.

Listing 191: 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>0D</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>
            <SubtractNotional>false</SubtractNotional>
        </ZeroCouponFixedLegData>
      </LegData>
```

8.3.19 CDS Reference Information

This trade component can be used to define the reference entity, tier, currency and documentation clause in credit derivative trades. For example, it can be used in the

`CreditDefaultSwapData` section in a CDS trade and in the `BasketData` section in credit derivatives involving more than one underlying reference entity. The value for each of these fields is generally agreed and specified in the credit derivative contract and they determine the credit curve that is used in pricing the trade.

Listing 192: CDS reference information node

```
<ReferenceInformation>
  <ReferenceEntityId>...</ReferenceEntityId>
  <Tier>...</Tier>
  <Currency>...</Currency>
  <DocClause>...</DocClause>
</ReferenceInformation>
```

The meanings and allowable values of the elements in the `ReferenceInformation` node are as follows:

- `ReferenceEntityId`: This is typically a six digit Markit RED code specifying the underlying reference entity with the prefix `RED:` e.g. `RED:008CA0`.
- `Tier`: The debt tier that is applicable for the specified reference entity in the credit derivative. Table 25 provides the allowable values.
- `Currency`: The currency that is applicable for the specified reference entity in the credit derivative. The `Currency` section in Table 14 provides the allowable values.
- `DocClause`: The documentation clause that is applicable for the specified reference entity in the credit derivative. This defines what constitutes a credit event for the contract as well as any limitations on the deliverable debt in the event of a credit event. Table 26 provides the allowable values.

8.3.20 Underlying

This trade component can be used to define the underlying entity for an Equity, Commodity or FX trade, but it can also define an underlying interest rate, inflation index or credit name. It can be used for a single underlying, or within a basket with associated weight. For an equity underlying a string representation is used to match `Underlying` node to requied configuration and reference data. The string representation is of the form IdentifierType:Name:Currency:Exchange, with all entries optional except for Name.

Listing 193: Underlying node

```
<Underlying>
  <Type>...</Type>
  <Name>...</Name>
  <Weight>...</Weight>
  <Currency>...</Currency>
  <IdentifierType>...</IdentifierType>
  <Exchange>...</Exchange>
  <PriceType>...</PriceType>
  <FutureMonthOffset>...</FutureMonthOffset>
  <DeliveryRollDays>...</DeliveryRollDays>
  <DeliveryRollCalendar>...</DeliveryRollCalendar>
</Underlying>
```

Example structures of the `Underlying` trade component node are shown in Listings 194 and 195 for an equity underlying, in Listing 198 for an fx underlying, in Listing 199 for a commodity underlying, in Listing 200 for an underlying interest rate index, in Listing 201 for an underlying inflation index, in Listing 202 for an underlying credit name.

Listing 194: Equity Underlying - RIC

```
        <Underlying>
            <Type>Equity</Type>
            <Name>.SPX</Name>
            <Weight>1.0</Weight>
            <IdentifierType>RIC</IdentifierType>
        </Underlying>
```

Listing 195: Equity Underlying - ISIN

```
        <Underlying>
            <Type>Equity</Type>
            <Name>NL0000852580</Name>
            <Weight>1.0</Weight>
            <IdentifierType>ISIN</IdentifierType>
            <Currency>EUR</Currency>
            <Exchange>XAMS</Exchange>
        </Underlying>
```

Listing 196: Equity Underlying - FIGI

```
        <Underlying>
            <Type>Equity</Type>
            <Name>BBG000BLNNV0</Name>
            <IdentifierType>FIGI</IdentifierType>
        </Underlying>
```

Listing 197: Equity Underlying - Bloomberg Identifier (Parsekey)

```
        <Underlying>
            <Type>Equity</Type>
            <Name>BARC LN Equity</Name>
            <IdentifierType>BBG</IdentifierType>
        </Underlying>
```

Listing 198: FX Underlying

```
        <Underlying>
          <Type>FX</Type>
          <Name>ECB-EUR-USD</Name>
          <Weight>1.0</Weight>
        </Underlying>
```

Listing 199: Commodity Underlying

```
        <Underlying>
          <Type>Commodity</Type>
          <Name>NYMEX:CL</Name>
          <Weight>1.0</Weight>
          <PriceType>FutureSettlement</PriceType>
          <FutureMonthOffset>0</FutureMonthOffset>
          <DeliveryRollDays>0</DeliveryRollDays>
          <DeliveryRollCalendar>TARGET</DeliveryRollCalendar>
        </Underlying>
```

Listing 200: InterestRate Underlying

```
        <Underlying>
          <Type>InterestRate</Type>
          <Name>USD-CMS-10Y</Name>
          <Weight>1.0</Weight>
        </Underlying>
```

Listing 201: Inflation Index Underlying

```
        <Underlying>
          <Type>Inflation</Type>
          <Name>USCPI</Name>
          <Weight>1.0</Weight>
          <!-- optional -->
          <Interpolation>Linear</Interpolation>
</Underlying>
```

Listing 202: Credit Underlying

```
        <Underlying>
          <Type>Credit</Type>
          <Name>ISSUER_A</Name>
          <Weight>1.0</Weight>
        </Underlying>
```

The meanings and allowable values of the elements in the `Underlying` node are as follows:

- `Type`: The type of the Underlying asset.

  Allowable values: *Equity*, *FX*, *Commodity*, *InterestRate*, *Inflation*, *Credit*.

- `Name`: The name of the Underlying asset.

  Allowable values:

  Equity: See `Name` for equity trades in Table 23

  FX: A string on the form SOURCE-CCY1-CCY2, where SOURCE is the FX fixing source, and the fixing is expressed as amount in CCY2 per one unit of CCY1. See Table 20, and note that the FX- prefix is not included in `Name` as it is already included in `Type`.

  InterestRate: Any valid interest rate index name, see Table 18

  Inflation: Any valid zero coupon inflation index (CPI) name, See Table 21

  Credit: Any valid credit name with a configured default curve, see Table 22

  Commodity: An identifier specifying the commodity being referenced in the leg. Table 24 lists the allowable values for `Name` and gives a description.

- `Weight` [Optional]: The relative weight of the underlying if part of a basket. For a single underlying this can be omitted or set to 1.

  Allowable values: A number between 0 and 1. Defaults to 1 if left blank or omitted

- `IdentifierType` [Optional]: Only valid when `Type` is *Equity*. The type of the identifier being used.

  Allowable values: *RIC*, *ISIN*, *FIGI*, *BBG*. Defaults to *RIC*, if left blank or omitted, and `Type`: is *Equity*.

- `Currency` [Mandatory when `IdentifierType` is *ISIN*]: Only valid when `Type` is *Equity*. The currency the underlying equity is quoted in. Used when `IdentifierType` is *ISIN*, to - together with the `Exchange` convert a given ISIN to a RIC code.

  Allowable values: See the `Currency` section in Table 14. Mandatory when `IdentifierType` is *ISIN*, should not be used when `IdentifierType` is *RIC*, and ignored otherwise. When `Type` is *Equity*, `Minor Currencies` in Table 14 are also allowable.

- `Exchange` [Mandatory when `IdentifierType` is *ISIN*]: Only valid when `Type` is *Equity*. A string code representing the exchange the equity is traded on. Used when `IdentifierType` is *ISIN*, to - together with the `Currency` convert a given ISIN to a RIC code.

  Allowable values: The MIC code of the exchange, see Table 27. Mandatory when `IdentifierType` is *ISIN*, should not be used when `IdentifierType` is *RIC*, and ignored otherwise.

- `PriceType` [Optional]: Only valid when `Type` is *Commodity*. Whether the Spot or Future price is referenced.

  Allowable values: *Spot*, *FutureSettlemment*. Mandatory when `Type` is *Commodity* .

- `FutureMonthOffset` [Optional]: Only valid when `Type` is *Commodity*. Only relevant for the *FutureSettlemment* price type, in which case the the $N + 1$th future with expiry greater than ObservationDate for the given commodity underlying will be referenced.

  Allowable values: An integer. Mandatory for when `Type` is *Commodity* and `PriceType` is *FutureSettlemment*.

- `DeliveryRollDays` [Optional]: Only valid when `Type` is *Commodity*. The number of days the observation date is rolled forward before the next future expiry is looked up.

  Allowable values: An integer. Defaults to 0 if left blank or omitted, and `Type`: is *Commodity*.

- `DeliveryRollCalendar` [Optional]: Only valid when `Type` is *Commodity*. The calendar used to roll forward the observation date.

  Allowable values: See Table 16. Defaults to the null calendar if left blank or omitted, and `Type`: is *Commodity*.

- `Interpolation` [Optional]: Only valid when `Type` is *Inflation*. The index observation interpolation between fixings.

  Allowable values: Flat, Linear

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`	*AED,AFN,ALL,AMD,ANG,AOA,ARS,AUD,AWG,AZN, BAM,BBD,BDT,BGN,BHD,BIF,BMD,BND,BOB,BOV, BRL,BSD,BTN,BWP,BYN,BZD,CAD,CDF,CHE,CHF, CHW,CLF,CLP,CNH,CNT,CNY,COP,COU,CRC,CUC, CUP,CVE,CZK,DJF,DKK,DOP,DZD,EGP,ERN,ETB, EUR,FJD,FKP,GBP,GEL,GGP,GHS,GIP,GMD,GNF, GTQ,GYD,HKD,HNL,HRK,HTG,HUF,IDR,ILS,IMP, INR,IQD,IRR,ISK,JEP,JMD,JOD,JPY,KES,KGS, KHR,KID,KMF,KPW,KRW,KWD,KYD,KZT,LAK,LBP, LKR,LRD,LSL,LYD,MAD,MDL,MGA,MKD,MMK,MNT, MOP,MRU,MUR,MVR,MWK,MXN,MXV,MYR,MZN,NAD, NGN,NIO,NOK,NPR,NZD,OMR,PAB,PEN,PGK,PHP, PKR,PLN,PYG,QAR,RON,RSD,RUB,RWF,SAR,SBD, SCR,SDG,SEK,SGD,SHP,SLL,SOS,SRD,SSP,STN, SVC,SYP,SZL,THB,TJS,TMT,TND,TOP,TRY,TTD, TWD,TZS,UAH,UGX,USD,USN,UYI,UYU,UYW,UZS, VES,VND,VUV,WST,XAF,XAG,XAU,XCD,XOF,XPD, XPF,XPT,XSU,XUA,YER,ZAR,ZMW,ZWL.* This full list of currencies is available via loading the provided `currencies.xml` at start-up. Note: Currency codes must also match available currencies in the `simulation.xml` file.
`Minor Currencies`	*GBp, GBX, ILa, ILX, ZAc, ZAC, ZAX*, Note: Minor Currency codes only supported for Equity products.
`Roll Convention`	*F, Following, FOLLOWING* *MF, ModifiedFollowing, Modified Following, MODIFIEDF* *P, Preceding, PRECEDING* *MP, ModifiedPreceding, Modified Preceding, MODIFIEDP* *U, Unadjusted, INDIFF*

Table 14: Allowable values for standard trade data.

`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.)
LastWednesday	All dates but effective date and termination date are taken to be on the last Wednesday of their month (with forward calculation.)
ThirdThursday	All dates but effective date and termination date are taken to be on the third Thursday of their month (with forward calculation.)
ThirdFriday	All dates but effective date and termination date are taken to be on the third Friday of their month (with forward calculation.)
MondayAfterThird-Friday	All dates but effective date and termination date are taken to be on the Monday following the third Friday of their month (with forward calculation.)
TuesdayAfterThird-Friday	All dates but effective date and termination date are taken to be on the Tuesday following the third Friday 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. For quarterly periods (`Tenor` set to *3M*): (Assuming no `FirstDate`/`LastDate`) Dates fall on 20th of March, June, September, December. A *Following* roll convention will be applied if the 20th falls on a non-business day. If the `EndDate` in the schedule is set to a date beyond the rolled quarterly CDS date, the actual trade termination date will be on the following quarterly CDS date. The first coupon will be paid on the quarterly CDS date following the `StartDate`, and be for the period since the previous quarterly CDS date. For monthly periods (`Tenor` set to *1M*): (Assuming no `FirstDate`/`LastDate`) Dates fall on 20th of each month, but the termination is still adjusted to be in line with quarterly periods. If the `EndDate` in the schedule is set to a date beyond the rolled quarterly CDS date (i.e. the 20th+roll Mar, Jun, Sep, Dec), the actual termination date will be on the following quarterly CDS date, causing a long final stub. The first coupon will be paid on the next 20th monthly following the `StartDate`, and be for the period since the previous month's 20th.
CDS2015	Credit derivatives standard rule updated in 2015. Same as *CDS* but with termination dates adjusted to 20th June and 20th December. For schedule `EndDates` from the 20th of March to the 19th September, both included, the termination date will fall on the 20th June (with *Following* roll). For schedule `EndDates` from the 20th September to the 19th March, both included, the termination date will fall on the 20th December (with *Following* roll).

Table 15: Allowable Values for Rule

`Calendar`	
Allowable Values	**Resulting Calendar**
TARGET, TGT, EUR	Target Calendar
CA, CAN, CAD, TRB	Canada Calendar
JP, JPN, JPY, TKB	Japan Calendar
CH, CHE, CHF, ZUB	Switzerland Calendar
GB, GBR, GBP, LNB, UK	UK Calendar
US, USA, USD, NYB	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
AR, ARG, ARS	Argentina Calendar
AU, AUD, AUS	Australia Calendar
AT, AUT, ATS	Austria Calendar
BE, BEL, BEF	Belgium Calendar
BW, BWA, BWP	Botswana Calendar
BR, BRA, BRL	Brazil Calendar
CL, CHL, CLP	Chile Calendar
CN, CHN, CNH, CNY	China Calendar
CO, COL, COP	Colombia Calendar
CZ, CZE, CZK	Czech Republic Calendar
DK, DNK, DKK, DEN	Denmark Calendar
FI, FIN	Finland Calendar
FR, FRF	France Calendar
DE, DEU	Germany Calendar
HK, HKG, HKD	Hong Kong Calendar
HU, HUN, HUF	Hungary Calendar
IS, ISL, ISK	Iceland Calendar
IN, IND, INR	India Calendar
ID, IDN, IDR	Indonesia Calendar
IL, ISR, ILS	Israel Calendar
Telbor	Tel Aviv Inter-Bank Offered Rate Calendar
IT, ITA, ITL	Italy Calendar
LU, LUX, LUF	Luxembourg Calendar
MX, MEX, MXN	Mexico Calendar
MY, MYS, MYR	Malaysia Calendar
NL, NLD, NZD	New Zealand Calendar
NO, NOR, NOK	Norway Calendar
PE, PER, PEN	Peru Calendar
PH, PHL, PHP	Philippines Calendar
PO, POL, PLN	Poland Calendar
RO, ROU, RON	Romania Calendar
RU, RUS, RUB	Russia Calendar
SAU, SAR	Saudi Arabia
SG, SGP, SGD	Singapore Calendar
ZA, ZAF, ZAR, SA	South Africa Calendar
KR, KOR, KRW	South Korea Calendar
ES, ESP	Spain Calendar
SE, SWE, SEK, SS	Sweden Calendar
TW, TWN, TWD	Taiwan Calendar
TH, THA, THB	Thailand Calendar
TR, TUR, TRY	Turkey Calendar
UA, UKR, UAH	Ukraine Calendar

BVMF	Brazil Bovespa Calendar
XTSE	Canada Toronto Stock Exchange Calendar
XSHG	China Shanghai Stock Exchange Calendar
XFRA	Germany Frankfurt Stock Exchange
XETR	Germany XETRA Calendar
ECAG	Germany EUREX Calendar
EUWA	Germany EUWAX Calendar
XJKT	Indonesia Jakarta Stock Exchange (now IDX) Calendar
XIDX	Indonesia Indonesia Stock Exchange Calendar
XTAE	Israel Tel Aviv Stock Exchange Calendar
XMIL	Italy Italian Stock Exchange Calendar
MISX	Russia Moscow Exchange Calendar
XKRX	Korea Exchange Calendar
XSWX	Switzerland SIX Swiss Exchange Calendar
XLON	UK London Stock Exchange
XLME	UK London Metal Exchange
XNYS	US New York Stock Exchange Calendar
WMR	Thomson Reuters QM/Reuters Spot
WeekendsOnly	Weekends Only Calendar
ICE_FuturesUS	ICE Futures U.S. Currency, Stock and Credit Index, Metal, Nat Gas, Power, Oil and Environmental
ICE_FuturesUS_1	ICE Futures U.S. Sugar, Cocoa, Coffee, Cotton and FCOJ
ICE_FuturesUS_2	ICE Futures U.S. Canola
ICE_FuturesEU	ICE Futures Europe
ICE_FuturesEU_1	ICE Futures Europe for contracts where 26 Dec is a holiday
ICE_EndexEnergy	ICE Endex European power and natural gas products
ICE_EndexEquities	ICE Endex European equities
ICE_SwapTradeUS	ICE Swap Trade U.S.
ICE_SwapTradeUK	ICE Swap Trade U.K.
ICE_FuturesSingapore	ICE futures Singapore
CME	CME group exchange calendar

Table 16: Allowable Values for Calendar. Combinations of calendars can be provided using comma separated calendar names.

`DayCount Convention`	
Allowable Values	**Resulting DayCount Convention**
A360, Actual/360, ACT/360, Act/360	Actual 360
A365, A365F, Actual/365 (Fixed), Actual/365 (fixed), ACT/365.FIXED, ACT/365, ACT/365L, Act/365, Act/365L	Actual 365 Fixed
A364, Actual/364, Act/364, ACT/364	Actual 364
Actual/365 (No Leap), Act/365 (NL), NL/365, Actual/365 (JGB)	Actual 365 Fixed (No Leap Year)
T360 ,30/360, 30/360 (Bond Basis), ACT/nACT	Thirty 360 (US)
30E/360 (Eurobond Basis), 30E/360, 30E/360.ISDA	Thirty 360 (European)
30/360 (Italian)	Thirty 360 (Italian)
ActActISDA, ACT/ACT.ISDA, Actual/Actual (ISDA), ActualActual (ISDA), ACT/ACT, ACT	Actual Actual (ISDA)
ActActISMA, Actual/Actual (ISMA), ActualActual (ISMA), ACT/ACT.ISMA	Actual Actual (ISMA)
ActActICMA, Actual/Actual (ICMA), ActualActual (ICMA), ACT/ACT.ICMA	Actual Actual (ICMA)
ActActAFB, ACT/ACT.AFB, Actual/Actual (AFB)	Actual Actual (AFB)
BUS/252, Business/252	Brazilian Bus/252
1/1	1/1

Table 17: 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-ESTER, EUR-ESTR, EUR-STR* *EUR-EURIBOR* *EUR-LIBOR* *EUR-CMS* *USD-FedFunds* *USD-SOFR* *USD-Prime* *USD-LIBOR* *USD-SIFMA* *USD-CMS* *GBP-SONIA* *GBP-LIBOR* *GBP-CMS* *GBP-BoEBase* *JPY-LIBOR* *JPY-TIBOR* *JPY-EYTIBOR* *JPY-TONAR* *JPY-CMS* *CHF-LIBOR* *CHF-SARON* *AUD-LIBOR* *AUD-BBSW* *CAD-CDOR* *CAD-BA* *SEK-STIBOR* *SEK-LIBOR* *SEK-STINA* *DKK-LIBOR* *DKK-CIBOR* *DKK-CITA* *SGD-SIBOR* *SGD-SOR* *HKD-HIBOR* *HKD-HONIA* *NOK-NIBOR* *HUF-BUBOR* *IDR-IDRFIX* *INR-MIFOR* *MXN-TIIE* *PLN-WIBOR* *SKK-BRIBOR* *THB-THBFIX* *THB-BIBOR* *NZD-BKBM*
TENOR	An integer followed by *D, W, M or Y*

Table 18: Allowable values for Index.

Defaults for `FixingDays`	
Index	**Default value**
Ibor indices	2, except for the Ibor indices below:
USD-SIFMA	1
GBP-LIBOR	0
AUD-BBSW	0
CAD-CDOR	0
CNY-SHIBOR	1
HKD-HIBOR	0
MXN-TIIE	1
MYR-KLIBOR	0
TRY-TRLIBOR	0
ZAR-JIBAR	0
Overnight indices	0, except for the Overnight indices below:
CHF-TOIS	1
CLP-CAMARA	2
PLN-POLONIA	1
DKK-DKKOIS	1
SEK-SIOR	1

Table 19: Defaults for FixingDays

`FX Index`	
Index Format	**Allowable Values**
FX-SOURCE-CCY1-CCY2	The FX- part of the string stays constant for all currency pairs. SOURCE is the market data fixing source defined in the market configuration. CCY1 and CCY2 are the ISO currency codes of the fx pair. Fixings are expressed as amount in CCY2 for one unit of CCY1.

Table 20: Allowable values for FX index fixings.

`Inflation CPI Index`	
Trade Data	**Allowable Values**
`Index` for CPI leg	Any string (provided it is the ID of an inflation index in the market configuration)

Table 21: Allowable values for CPI index.

Credit `CreditCurveId`	
Trade Data	**Allowable Values**
`CreditCurveId` for credit trades - single name and index	Any string (provided it is the ID of a single name or index reference entity in the market conguration). Typically a RED-code with the *RED:* prefix Examples: *RED:2I65BRHH6* (CDX N.A. High Yield, Series 13, Version 1) *RED:008CA0|SNRFOR|USD|MR14* (Agilent Tech Senior USD)

Table 22: Allowable values for credit `CreditCurveId`

Equity `Name`	
Trade Data	**Allowable Values**
`Name` for equity trades	Any string (provided it is the ID of an equity in the market conguration). Typically a RIC-code with the *RIC:* prefix Examples: *RIC:.SPX* (S&P 500 Index) *RIC:EEM.N* (iShares MSCI Emerging Markets ETF)

Table 23: Allowable values for equity `Name`.

Commodity Curve Name	
Trade Data	**Allowable Values**
`Name` for commodity trades	Any string (provided it is the ID of an commodity in the market configuration)

Table 24: Allowable values for commodity data.

`Tier`	
Value	**Description**
`SNRFOR`	Senior unsecured for corporates or foreign debt for sovereigns
`SUBLT2`	Subordinated or lower Tier 2 debt for banks
`SNRLAC`	Senior loss absorbing capacity
`SECDOM`	Secured for corporates or domestic debt for sovereigns
`JRSUBUT2`	Junior subordinated or upper Tier 2 debt for banks
`PREFT1`	Preference shares or Tier 1 capital for banks

Table 25: Allowable values for `Tier`

DocClause	
Value	**Description**
`CR`	Full or old restructuring referencing the 2003 ISDA Definitions
`MM`	Modified modified restructuring referencing the 2003 ISDA Definitions
`MR`	Modified restructuring referencing the 2003 ISDA Definitions
`XR`	No restructuring referencing the 2003 ISDA Definitions
`CR14`	Full or old restructuring referencing the 2014 ISDA Definitions
`MM14`	Modified modified restructuring referencing the 2014 ISDA Definitions
`MR14`	Modified restructuring referencing the 2014 ISDA Definitions
`XR14`	No restructuring referencing the 2014 ISDA Definitions

Table 26: Allowable values for `DocClause`

Exchange	
Trade Data	**Allowable Values**
`Exchange`	Any string, typically a MIC code (provided it is the ID of an exchange in the market configuration)

Table 27: Allowable Values for Exchange

Boolean nodes	
Node Value	**Evaluates To**
`Y`, `YES`, `TRUE`, `true`, `1`	`true`
`N`, `NO`, `FALSE`, `false`, `0`	`false`

Table 28: Allowable values for boolean node

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 203: 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 204: 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>
        <ApplyInitialMargin>Y</ApplyInitialMargin>
        <InitialMarginType>Bilateral</InitialMarginType>
    </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.

ApplyInitialMargin Apply (dynamic) initial Margin in addition to variation margin

InitialMarginType There are three possible values here:

- Bilateral: Both parties to the CSA are legally entitled to request collateral to cover their MPOR 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.

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 14.

- **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 29.

- **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, COMMODITY, COMMODITY_FWD, COMMODITY_OPTION*
Quote Type	*BASIS_SPREAD, CREDIT_SPREAD, YIELD_SPREAD, HAZARD_RATE, RATE, RATIO, PRICE, RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, BASE_CORRELATION, SHIFT*

Table 29: Allowable values for Instrument and Quote type market data.

An excerpt from a typical `market.txt` file is shown in Listing 205.

Listing 205: 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 30.

Property	**Allowable values**	**Description**
Instrument Type	*ZERO*	
Quote Type	*RATE, YIELD_SPREAD*	
Currency	See `Currency` in Table 14	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 17	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 14	Either a Tenor for tenor based zero quotes, or an explicit maturity date (ZeroDate)

Table 30: 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 31.

Property	**Allowable values**	**Description**
Instrument Type	*DISCOUNT*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 14	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 14	Either a Term is used to determine the maturity date, or an explicit maturity date (Discount Date) is given.

Table 31: 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 14	Unit/Source currency
Target currency	See `Currency` in Table 14	Target currency

Table 32: 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 14	Unit/Source currency
Target currency	See `Currency` in Table 14	Target currency
Term	An integer followed by D, W, M or Y.	Period from today to maturity

Table 33: 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 14	Currency of the Deposit rate
IndexName	Optional, any string	Generally used to differentiate money market rates referencing different interest rate indices with the same tenor
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 34: 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 7.11, 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 14	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 35: 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 14	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 36: 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 14	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 37: Money Market Futures Price

Expiry month is quoted here as YYYY-MM. The exact expiry date follows from a date rule defined in the future conventions, see 7.11.3.

Example:

- MM_FUTURE/PRICE/EUR/2018-06/LIF3ME/3M

10.8 Overnight Index Futures Price

Property	**Allowable values**	**Description**
Instrument Type	*OI_FUTURE*	
Quote Type	*PRICE*	
Currency	See `Currency` in Table 14	Currency of the Overnight Index Future price
Expiry	Alphanumeric string of the form YYYY-MM	Expiry month and year
Contract	String	Contract name
Term	An integer followed by M or Y	Underlying Term in months or years

Table 38: Overnight Index Futures Price

Expiry month is quoted here as YYYY-MM. The exact expiry date follows from a date rule defined in the future conventions, see 7.11.3.

Example: Three Months SOFR Futures (DEC 2019):

- OI_FUTURE/PRICE/USD/2019-12/CME:SR3Z2019/3M

10.9 Swap Rate

Property	Allowable values	Description
Instrument Type	*IR_SWAP*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 14	Currency of the Swap rate
IndexName	Optional, any string	Generally used to differentiate swaps referencing different interest rate indices with the same tenor
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 39: Swap Rate

Forward start is usually not quoted, but needs to be derived from conventions.

Example:

- IR_SWAP/RATE/EUR/2D/6M/10Y

10.10 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 14	Currency of the basis swap spread
Optional Identifier	String	Basis swap name
Term	An integer followed by D, W, M or Y	Swap length from start to maturity

Table 40: Basis Swap Spread

Examples:

- BASIS_SWAP/BASIS_SPREAD/6M/3M/CHF/10Y
- BASIS_SWAP/BASIS_SPREAD/3M/1D/USD/2Y

- BASIS_SWAP/BASIS_SPREAD/3M/1D/USD/LIBOR_PRIME/2Y
- BASIS_SWAP/BASIS_SPREAD/3M/1D/USD/LIBOR_FEDFUNDS/2Y

10.11 Cross Currency Basis Swap Spread

Property	**Allowable values**	**Description**
Instrument Type	*CC_BASIS_SWAP*	
Quote Type	*BASIS_SPREAD*	
Flat currency	See `Currency` in Table 14	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 14	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 41: Cross Currency Basis Swap Spread

Example:

- CC_BASIS_SWAP/BASIS_SPREAD/USD/3M/JPY/6M/10Y

10.12 CDS Spread

Property	**Allowable values**	**Description**
Instrument Type	`CDS`	
Quote Type	`CREDIT_SPREAD` or `CONV_CREDIT_SPREAD`	
Entity	String	The CDS reference entity name
Tier	String	The CDS tier
DocClause	String	Optional, the CDS doc clause
Currency	See `Currency` in Table 14	The CDS currency
Term	A valid tenor string	The CDS tenor
RunningSpread	A number	The CDS running coupon in bps e.g. 100 for 0.01

Table 42: CDS spread quote

There are two possible quote types to allow for the presence of two CDS spread types in the market data. In particular, there is typically a conventional CDS spread and a par CDS spread quoted in the market. The quote type distinction here would allow the conventional spreads to be stored with quote type set to `CONV_CREDIT_SPREAD` and the par spreads to be stored with quote type set to `CREDIT_SPREAD`. As noted in the table above, the CDS documentation clause and CDS running spread is optional. The following list shows valid CDS spread quote examples.

- `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/5Y`
- `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/5Y/100`
- `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/XR14/5Y`
- `CDS/CREDIT_SPREAD/JPM/SNRFOR/USD/XR14/5Y/100`
- `CDS/CREDIT_SPREAD/RBS/SUBLT2/EUR/MR14/10Y`
- `CDS/CREDIT_SPREAD/RBS/SUBLT2/EUR/MR14/10Y/500`
- `CDS/CREDIT_SPREAD/RBS/SUBLT2/EUR/1Y`
- `CDS/CREDIT_SPREAD/RBS/SUBLT2/EUR/1Y/500`

10.13 CDS Upfront Price

Property	Allowable values	Description
Instrument Type	`CDS`	
Quote Type	`PRICE`	
Entity	String	The CDS reference entity name
Tier	String	The CDS tier
Currency	See `Currency` in Table 14	The CDS currency
DocClause	String	Optional, the CDS doc clause
Term	A valid tenor string	The CDS tenor
RunningSpread	A number	The CDS running coupon in bps e.g. 100 for 0.01

Table 43: CDS upfront price quote

As noted in the table above, the CDS documentation clause and CDS running spread is optional. Note that if the running spread is omitted from the CDS upfront price quote string, it should be included in any default curve configuration that uses those quotes. In other words, to bootstrap a default curve from CDS price quotes, the contractual running spread needs to be provided in either the quote string or in the default curve configuration. If both are provided, the running spread in the quote string takes precedence. The following list shows valid CDS upfront price quote examples.

- `CDS/PRICE/JPM/SNRFOR/USD/5Y`
- `CDS/PRICE/JPM/SNRFOR/USD/5Y/100`
- `CDS/PRICE/JPM/SNRFOR/USD/XR14/5Y`
- `CDS/PRICE/JPM/SNRFOR/USD/XR14/5Y/100`

- CDS/PRICE/RBS/SUBLT2/EUR/MR14/10Y
- CDS/PRICE/RBS/SUBLT2/EUR/MR14/10Y/500
- CDS/PRICE/RBS/SUBLT2/EUR/1Y
- CDS/PRICE/RBS/SUBLT2/EUR/1Y/500

10.14 CDS Recovery Rate

Property	**Allowable values**	**Description**
Instrument Type	RECOVERY_RATE	
Quote Type	RATE	
Entity	String	The CDS reference entity name
Tier	String	The CDS tier
Currency	See Currency in Table 14	The CDS currency
DocClause	String	Optional, the CDS doc clause

Table 44: CDS Recovery Rate

As noted in the table above, the CDS documentation clause is optional. The following list shows valid recovery rate quote examples.

- RECOVERY_RATE/RATE/JPM/SNRFOR/USD
- RECOVERY_RATE/RATE/JPM/SNRFOR/USD/XR14
- RECOVERY_RATE/RATE/RBS/SUBLT2/EUR/MR14
- RECOVERY_RATE/RATE/RBS/SUBLT2/EUR

10.15 CDS Option Implied Volatility

A CDS option implied volatility quote can take any one of the following four forms:

1. INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[EXPIRY]
2. INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[EXPIRY]/[STRIKE]
3. INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[TERM]/[EXPIRY]
4. INDEX_CDS_OPTION/RATE_LNVOL/[NAME]/[TERM]/[EXPIRY]/[STRIKE]

The terms in the quote string have the following interpretations:

- The [NAME] is the name of the CDS reference entity or index CDS.
- The [EXPIRY] is the expiry of the CDS option and may be a tenor or an explicit date.
- The [TERM] is optional and gives the term of the underlying CDS or index CDS. This should be a tenor e.g. 3Y, 5Y, etc.
- The [STRIKE] is optional and gives the strike of the CDS or index CDS option.

10.16 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 45: Security Recovery Rate

Example:

- RECOVERY_RATE/RATE/SECURITY_1

10.17 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 14	Hazard rate currency
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 46: Hazard Rate

Example:

- HAZARD_RATE/RATE/CPTY_A/SR/USD/30Y
- HAZARD_RATE/RATE/CPTY_C/SR/EUR/0Y

10.18 FX Option Implied Volatility

Property	**Allowable values**	**Description**
Instrument Type	*FX_OPTION*	
Quote Type	*RATE_LNVOL*	
Unit currency	See `Currency` in Table 14	Unit/Source currency
Target currency	See `Currency` in Table 14	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 47: 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

10.19 Cap Floor Implied Volatility

Property	**Allowable values**	**Description**
Instrument Type	*CAPFLOOR*	
Quote Type	*RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, SHIFT*	Lognormal quoted volatility, normal quoted volatility, shifted lognormal quoted volatility, shift quote.
Currency	See `Currency` in Table 14	Currency of the cap floor volatility.
Term	A valid tenor string	Period from start to cap or floor maturity.
IndexTenor	A valid tenor string	Underlying index tenor e.g. `3M` for `EUR-EURIBOR-3M`.
ATM	`1` or `0`	True, i.e. `1`, for an ATM quote and false, i.e. `0`, for a strike quote.
Relative	`1` or `0`	Should be set to `1` for a quote relative to ATM and to `0` for an absolute strike quote.
Strike	Real number	Strike of cap or floor. Should be set to `0` for an ATM quote.

Table 48: Cap floor implied volatility quote

If a cap floor shift quote needs to be provided, i.e. in the case of a shifted lognormal surface, the quote is of the form `CAPFLOOR/SHIFT/Currency/IndexTenor` where the meaning of `Currency` and `IndexTenor` are given in Table 48.

We have the following examples of cap floor implied volatility quotes:

- `CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/1/1/0`: 10Y ATM cap floor implied lognormal volatility quote where the index tenor is 6M.
- `CAPFLOOR/RATE_LNVOL/EUR/10Y/6M/0/0/0.035`: 10Y 3.5% strike cap floor implied lognormal volatility quote where the index tenor is 6M.

10.20 Swaption Implied Volatility

Property	**Allowable values**	**Description**
Instrument Type	*SWAPTION*	
Quote Type	*RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, SHIFT*	Lognormal quoted volatility, Normal quoted volatility, shifted lognormal quoted volatility, shift quote.
Currency	See `Currency` in Table 14	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 49: 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).
If a swaption shift quote needs to be provided, i.e. in the case of a shifted lognormal surface, the quote is of the form `SWAPTION/SHIFT/Currency/Term` where the meaning of `Currency` and `Term` are given in Table 49.

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)

10.21 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 14	Currency of the equity

Table 50: Equity Spot Price

10.22 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 14	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 51: Equity Forward Price

Examples:

- EQUITY_FWD/PRICE/SP5/USD/2016-06-16
- EQUITY_FWD/PRICE/SP5/USD/2Y

10.23 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 14	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 52: Equity Dividend Yield Rate

Examples:

- EQUITY_DIVIDEND/RATE/SP5/USD/2016-06-16
- EQUITY_DIVIDEND/RATE/SP5/USD/2Y

10.24 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 14	Currency of the equity
Expiry	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote
Strike	`ATM/AtmSpot` (= `ATM`), `ATM/AtmFwd` (=`ATMF`), `MNY/[Spot|Fwd]/1.2` where 1.2 is the moneyness level, or a `Real` for an absolute strike	strike
CallPut	`C` for Call, `P` for Put	Optional Call/Put flag (defaults to Call)

Table 53: Equity Option Implied Volatility

Volatilities are quoted as a function of strike price - either at-the-money spot or forward, a moneyness level or else a specified real number, corresponding to the absolute strike value. Only log-normal implied volatilities (`RATE_LNVOL`) are supported.

If K is the absolute strike, S the spot, F the forward and m the moneyness level, we have $K = Sm$ if spot moneyness and $K = Fm$ if forward moneyness is specified.

Example:

- EQUITY_OPTION/RATE_LNVOL/SP5/USD/6M/ATMF
- EQUITY_OPTION/RATE_LNVOL/SP5/USD/2018-06-30/ATMF
- EQUITY_OPTION/RATE_LNVOL/SP5/USD/6M/MNY/Fwd/1.2

10.25 Equity Option Premium

Property	**Allowable values**	**Description**
Instrument Type	*EQUITY_OPTION*	
Quote Type	*PRICE*	
Name	String	Identifying name of the equity
Currency	See `Currency` in Table 14	Currency of the equity
Expiry	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote
Strike	`ATM/AtmSpot` (= `ATM`), `ATM/AtmFwd` (=`ATMF`) or a `Real` for an absolute strike	strike
CallPut	`C` for Call, `P` for Put	Optional Call/Put flag (defaults to Call)

Table 54: Equity Option Premium

Premiums are quoted as a function of strike price - either at-the-money spot or forward or else a specified real number, corresponding to the absolute strike value.

Example:

- EQUITY_OPTION/PRICE/SP5/USD/6M/ATMF
- EQUITY_OPTION/PRICE/SP5/USD/2018-06-30/2000
- EQUITY_OPTION/PRICE/SP5/USD/2018-06-30/2000/C

10.26 Commodity Spot Price

Property	**Allowable values**	**Description**
Instrument Type	*COMMODITY*	
Quote Type	*PRICE*	
Name	String	Identifying name of the commodity
Currency	See `Currency` in Table 14	Currency of the commodity

Table 55: Commodity Spot Price

Examples:

- COMMODITY/PRICE/PM:XAUUSD/USD

10.27 Commodity Forward Price

Property	Allowable values	Description
Instrument Type	*COMMODITY_FWD*	
Quote Type	*PRICE*	
Name	String	Identifying name of the commodity
Currency	See `Currency` in Table 14	Currency of the commodity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 56: Commodity Forward Price

Examples:

- COMMODITY_FWD/PRICE/NYMEX:CL/USD/2030-11-20

10.28 Commodity Option Implied Volatility

Property	Allowable values	Description
Instrument Type	*COMMODITY_OPTION*	
Quote Type	*RATE_LNVOL*	
Name	String	Identifying name of the commodity
Currency	See `Currency` in Table 14	Currency of the commodity
Expiry	Date string, or integer followed by D, W, M or Y or continuation notation c1, c2, etc	Expiry of the volatility quote, for continuation notation c1 indicates the next expiry, c2 the one after that, etc.
Vol Quote Type, or Absolute Strike	`DEL`, `ATM` or `MNY` or a `Real` for absolute strikes	`DEL` is for delta quotes, i.e. the volatility is for a call or put option with a given delta. `ATM` is for At-The-Money volatility quotes. `MNY` is for volatility smile quotes for given relative moneyness levels. Each Vol Quote Type is described further in sub-tables below. Note that instead of a Vol Quote Type, an absolute strike level can be entered

Table 57: Commodity Option Implied Volatility - Root table

Property	**Allowable values**	**Description**
Vol Quote Type	`DEL`, `ATM` or `MNY`	In this table it is assumed `DEL` is chosen.
Delta Convention	`Fwd` or `Spot`	Delta forward or spot quote
Option Type	`Call` or `Put`	Option type for the delta quote
Delta	a positive `Real` number, typically between 0.1 and 0.45	Delta, e.g. a delta of 0.40 for a Call means that if the underlying commodity price increases by 1 unit, the call option price increases by 0.40 units

Table 58: Commodity Option Implied Volatility - Delta Quotes Table

Property	**Allowable values**	**Description**
Vol Quote Type	`DEL`, `ATM` or `MNY`	In this table it is assumed `ATM` is chosen.
Atm Convention	`AtmFwd`, `AtmSpot`, or `AtmDeltaNeutral`	Atm Forward or Atm Spot quote, to be used standalone, or when the smile is given by moneyness (`MNY`) quotes. The Delta Neutral Atm to be used as standalone, or when the smile is given by delta (`DEL`) quotes. Note that when `AtmFwd` or `AtmSpot` are used, the string stops here, no further entries "tokens" are required
Atm Quote Type	`DEL`	When `AtmDeltaNeutral` is used, the quote type must be set to Delta (`DEL`)
Atm Delta Convention	`Fwd` or `Spot`	When `AtmDeltaNeutral` is used, the Atm delta quote can be a Spot or Forward quote.

Table 59: Commodity Option Implied Volatility - ATM Quotes Table

Property	**Allowable values**	**Description**
Vol Quote Type	`DEL`, `ATM` or `MNY`	In this table it is assumed `MNY` is chosen.
Moneyness Type	`Fwd` or `Spot`	Moneyness Forward or Spot quote
Moneyness	a positive `Real` number	The relative moneyness expressed in decimal form, relative to the AtmSpot or the AtmFwd strikes

Table 60: Commodity Option Implied Volatility - Moneyness Quotes Table

Volatilities are quoted:

- as a function of the delta - either the delta neutral at-the-money spot or forward, or for a call or put option with a given delta, or
- as a function of strike price - either at-the-money spot or forward, or a relative strike moneyness level, or else
- as a specified real number, corresponding to the absolute strike value.

Only log-normal implied commodity volatilities (`RATE_LNVOL`) are supported.

For strike quoted volatilities, If K is the absolute strike, S the spot, F the forward and m the moneyness level, we have $K = Sm$ if spot moneyness and $K = Fm$ if forward moneyness is specified.

Example of delta forward quotes:

COMMODITY_OPTION/RATE_LNVOL/ICE:B/USD/c9/DEL/Fwd/Put/0.40
COMMODITY_OPTION/RATE_LNVOL/ICE:B/USD/c9/DEL/Fwd/Put/0.45
COMMODITY_OPTION/RATE_LNVOL/ICE:B/USD/c9/ATM/AtmDeltaNeutral/DEL/Fwd
COMMODITY_OPTION/RATE_LNVOL/ICE:B/USD/c9/DEL/Fwd/Call/0.45
COMMODITY_OPTION/RATE_LNVOL/ICE:B/USD/c9/DEL/Fwd/Call/0.40

Example of delta spot quotes:

COMMODITY_OPTION/RATE_LNVOL/PM:XAGEUR/EUR/1Y/DEL/Spot/Put/0.35
COMMODITY_OPTION/RATE_LNVOL/PM:XAGEUR/EUR/1Y/DEL/Spot/Put/0.45
COMMODITY_OPTION/RATE_LNVOL/PM:XAGEUR/EUR/1Y/ATM/AtmDeltaNeutral/DEL
/Spot
COMMODITY_OPTION/RATE_LNVOL/PM:XAGEUR/EUR/1Y/DEL/Spot/Call/0.25
COMMODITY_OPTION/RATE_LNVOL/PM:XAGEUR/EUR/1Y/DEL/Spot/Call/0.15

Example of forward strike quotes with relative moneyness:

COMMODITY_OPTION/RATE_LNVOL/NYMEX:AA5/USD/c11/MNY/Fwd/1.40
COMMODITY_OPTION/RATE_LNVOL/NYMEX:AA5/USD/c11/MNY/Fwd/1.20
COMMODITY_OPTION/RATE_LNVOL/NYMEX:AA5/USD/c11/ATM/AtmFwd
COMMODITY_OPTION/RATE_LNVOL/NYMEX:AA5/USD/c11/MNY/Fwd/0.80
COMMODITY_OPTION/RATE_LNVOL/NYMEX:AA5/USD/c11/MNY/Fwd/0.60

Example of absolute moneyness quotes:

COMMODITY_OPTION/RATE_LNVOL/PM:XAUUSD/USD/c12/1600
COMMODITY_OPTION/RATE_LNVOL/PM:XAUUSD//USD/c12/1700
COMMODITY_OPTION/RATE_LNVOL/PM:XAUUSD//USD/c12/1800
COMMODITY_OPTION/RATE_LNVOL/PM:XAUUSD//USD/c12/1900
COMMODITY_OPTION/RATE_LNVOL/PM:XAUUSD//USD/c12/2000

10.29 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 61: 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, FRCPI, UKRPI, USCPI, ZACPI, BEHICP, AUCPI.

10.30 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 62: 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, FRCPI, UKRPI, USCPI, ZACPI, BEHICP.

10.31 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 63: 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, FRCPI, UKRPI, USCPI, ZACPI, BEHICP.

10.32 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 64: 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, FRCPI, UKRPI, USCPI, ZACPI, BEHICP.

10.33 Bond Yield Spreads

Property	**Allowable values**	**Description**
Instrument Type	*BOND*	
Quote Type	*YIELD_SPREAD*	
Name	String	Identifying name of the bond

Table 65: Bond Yield Spreads

This quote provides the spread for a specified bond over the benchmark rate.

Examples:

- BOND/YIELD_SPREAD/SECURITY_1

10.34 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 66: 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

10.35 Conditional Prepayment Rates

Property	**Allowable values**	**Description**
Instrument Type	*CPR*	
Quote Type	*RATE*	
Name	String	Identifying name of the bond

Table 67: 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 14.

- Index Name: The name of the Index.

  Allowable values are given in Table 68.

- 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 206. Note that alternative index name formats are used (Table 68).

Listing 206: 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**
	EUR-EONIA
	EUR-EURIBOR
	EUR-LIBOR
	USD-FedFunds
	USD-Prime
	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-STINA
	SEK-LIBOR
	DKK-LIBOR
	DKK-CIBOR
CCY-INDEX	*DKK-CITA*
	SGD-SIBOR
	HKD-HIBOR
	CZK-PRIBOR
	HUF-BUBOR
	IDR-IDRFIX
	INR-MIFOR
	JPY-TIBOR
	JPY-EYTIBOR
	KRW-KORIBOR
	MXN-TIIE
	MYR-KLIBOR
	NOK-NIBOR
	NZD-BKBM
	PLN-WIBOR
	SEK-STIBOR
	SGD-SOR
	SKK-BRIBOR
	TWD-TAIBOR
	THB-THBFIX
	THB-BIBOR
	ZAR-JIBAR
	DEM-LIBOR
TENOR	An integer followed by *D, W, M or Y*

Table 68: 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 or CCY-CMS-TAG-TENOR:	
Index Component	**Allowable Values**
CCY	Any supported currency code
CMS	Must be "CMS" (to denote a swap index)
TAG	An optional tag that allows to define several CMS indices per currency
TENOR	An integer followed by *D, W, M or Y*

Table 69: 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 70: Allowable values for FX fixings.

Zero Inflation Indices of form NAME:	
Index Component	**Allowable Values**
NAME	*CACPI* *DKCPI* *EUHICP* *EUHICPXT* *FRHICP* *FRCPI* *SECPI* *UKRPI* *USCPI* *ZACPI* *BEHICP*

Table 71: Allowable values for zero inflation indices.

Generic Indices of form GENERIC-NAME:	
Index Component	**Allowable Values**
GENERIC	Must be "GENERIC" (to denote a generic fixing)
NAME	Any string

Table 72: Allowable values for generic indices.

12 Dividends History

Historical dividend payments data in the `dividends.txt` file is given in three columns; Equity Name, Ex-Dividend Date and Dividend Amount in the Currency of the Equity Curve. Columns are separated by semicolons ";" or blanks. Dividends are used in some trades with path dependent features.

- Ex-Dividend Date: The day the stock starts trading without the value of the dividend payment

  Allowable values: See `Date` in Table 14.

- Equity Name: The name of the dividend paying equity.

  Allowable values are the names of the Equity Curves defined in `curveconfig.xml`.

- Dividend Amount: The amount of the dividend payment date.

  Allowable values: Any real number (not expressed as a percentage).

An excerpt of a fixings file is shown in Listing 207.

Listing 207: Excerpt of a dividends file

```
20130411 DAI:GR 2.2
20140410 DAI:GR 2.25
20150402 DAI:GR 2.45
20160407 DAI:GR 3.25
20170330 DAI:GR 3.25
20180406 DAI:GR 3.65
20190523 DAI:GR 3.25
20120815 HSBA:LN 5.5538
20121024 HSBA:LN 5.604
20130320 HSBA:LN 11.585
20130522 HSBA:LN 6.58
20130821 HSBA:LN 6.2033
20131023 HSBA:LN 6.102
20140312 HSBA:LN 11.2919
20140521 HSBA:LN 5.8768
20140820 HSBA:LN 6.1622
20141023 HSBA:LN 6.3633
20150305 HSBA:LN 13.4
20150521 HSBA:LN 6.3709
20150813 HSBA:LN 6.4436
20151022 HSBA:LN 6.6015
20160303 HSBA:LN 14.7908
20160519 HSBA:LN 7.5421
20160811 HSBA:LN 7.6633
20161020 HSBA:LN 8.0417
20170223 HSBA:LN 16.6757
20170518 HSBA:LN 7.8636
20170803 HSBA:LN 7.577
20171012 HSBA:LN 7.6405
20180222 HSBA:LN 14.762
20180517 HSBA:LN 7.5502
20180816 HSBA:LN 7.632
20181011 HSBA:LN 7.78
20190221 HSBA:LN 15.9271
20190516 HSBA:LN 7.8368
```

A Methodology Summary

A.1 Risk Factor Evolution Model

ORE applies the cross asset model described in detail in [21] 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. Extensions to full simulation of Commodity is planned.
The Cross Asset Model is based on the Linear Gauss Markov model (LGM) for interest rates, lognormal FX and equity processes, Dodgson-Kainth model for inflation, LGM or Extended Cox-Ingersoll-Ross model (CIR++) for credit. 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 [21], 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 $k = 1, \ldots, K$ and the credit processes in the domestic LGM measure with state variables C, k and $y_{C,k}$ 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

$$\begin{aligned}
dz_0 &= \alpha_0 \, dW_0^z \\
dz_i &= \gamma_i \, dt + \alpha_i \, dW_i^z, \qquad i > 0 \\
\frac{dx_i}{x_i} &= \mu_i \, dt + \sigma_i \, dW_i^x, \qquad i > 0 \\
\frac{ds_j}{s_j} &= \mu_j^S \, dt + \sigma_j^S \, dW_j^S \\
dz_{I,k} &= \alpha_{I,k}(t) dW_k^I \\
dy_{I,k} &= \alpha_{I,k}(t) H_{I,k}(t) dW_k^I \\
dz_{C,k} &= \alpha_{C,k}(t) dW_k^C \\
dy_{C,k} &= H_{C,k}(t) \alpha_{C,k}(t) dW_k^C
\end{aligned}$$

$$\begin{aligned}
\gamma_i &= -\alpha_i^2 \, H_i - \rho_{ii}^{zx} \, \sigma_i \, \alpha_i + \rho_{i0}^{zz} \, \alpha_i \, \alpha_0 \, H_0 \\
\mu_i &= r_0 - r_i + \rho_{0i}^{zx} \, \alpha_0 \, H_0 \, \sigma_i \\
\mu_j^S &= (r_{\phi(j)}(t) - q_j(t) + \rho_{0j}^{zs} \alpha_0 H_0 \sigma_j^S - \epsilon_{\phi(j)} \rho_{j\phi(j)}^{sx} \sigma_j^S \sigma_{\phi(j)}) \\
r_i &= f_i(0,t) + z_i(t) \, H_i'(t) + \zeta_i(t) \, H_i(t) \, H_i'(t), \quad \zeta_i(t) = \int_0^t \alpha_i^2(s) \, ds \\
dW_a^\alpha \, dW_b^\beta &= \rho_{ij}^{\alpha\beta} \, dt, \qquad \alpha, \beta \in \{z, x, I, C\}, \qquad a, b \text{ suitable indices}
\end{aligned}$$

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:

$$N(t) = \frac{1}{P(0,t)} \exp\left\{ H_t \, z_t + \frac{1}{2} H_t^2 \, \zeta_t \right\} \tag{6}$$

$$P(t,T,z_t) = \frac{P(0,T)}{P(0,t)} \, \exp\left\{ -(H_T - H_t) \, z_t - \frac{1}{2} \left(H_T^2 - H_t^2 \right) \, \zeta_t \right\}. \tag{7}$$

Note that the LGM model is closely related to the Hull-White model in T-forward measure [21].

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 [21], chapter 16. The expectation of the interest rate process z_i conditional on $\mathcal{F}_{t_0}$ at $t_0+\Delta t$ is

$$\begin{aligned}
\mathbb{E}_{t_0}[z_i(t_0+\Delta t)] &= z_i(t_0)+\mathbb{E}_{t_0}[\Delta z_i], \qquad \text{with} \quad \Delta z_i = z_i(t_0+\Delta t)-z_i(t_0) \\
&= z_i(t_0) - \int_{t_0}^{t_0+\Delta t} H_i^z\,(\alpha_i^z)^2\,du + \rho^{zz}_{0i}\int_{t_0}^{t_0+\Delta t} H_0^z\,\alpha_0^z\,\alpha_i^z\,du \\
&\quad -\epsilon_i\rho^{zx}_{ii}\int_{t_0}^{t_0+\Delta t}\sigma_i^x\,\alpha_i^z\,du
\end{aligned}$$

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

$$\begin{aligned}
\mathbb{E}_{t_0}[\ln x_i(t_0+\Delta t)] &= \ln x_i(t_0)+\mathbb{E}_{t_0}[\Delta \ln x_i], \qquad \text{with} \quad \Delta\ln x_i = \ln x_i(t_0+\Delta t)-\ln x_i(t_0) \\
&= \ln x_i(t_0) + (H_0^z(t)-H_0^z(s))\,z_0(s) - (H_i^z(t)-H_i^z(s))\,z_i(s) \\
&\quad +\ln\left(\frac{P_0^n(0,s)}{P_0^n(0,t)}\frac{P_i^n(0,t)}{P_i^n(0,s)}\right) \\
&\quad -\frac{1}{2}\int_s^t (\sigma_i^x)^2\,du \\
&\quad +\frac{1}{2}\left((H_0^z(t))^2\zeta_0^z(t)-(H_0^z(s))^2\zeta_0^z(s)-\int_s^t (H_0^z)^2(\alpha_0^z)^2\,du\right) \\
&\quad -\frac{1}{2}\left((H_i^z(t))^2\zeta_i^z(t)-(H_i^z(s))^2\zeta_i^z(s)-\int_s^t (H_i^z)^2(\alpha_i^z)^2\,du\right) \\
&\quad +\rho^{zx}_{0i}\int_s^t H_0^z\,\alpha_0^z\,\sigma_i^x\,du \\
&\quad -\int_s^t (H_i^z(t)-H_i^z)\,\gamma_i\,du, \qquad \text{with} \quad s=t_0, \quad t=t_0+\Delta t
\end{aligned}$$

with

$$\gamma_i = -H^z_i\,(\alpha^z_i)^2 + H^z_0\,\alpha^z_0\,\alpha^z_i\,\rho^{zz}_{0i} - \sigma^x_i\,\alpha^z_i\,\rho^{zx}_{ii}$$

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

$$\begin{aligned}
\mathbb{E}_{t_0}[\ln s_j(t_0+\Delta t)] &= \ln s_j(t_0) + \mathbb{E}_{t_0}[\Delta \ln s_j], \qquad \text{with} \quad \Delta \ln s_j = \ln s_j(t_0+\Delta t) - \ln s_j(t_0) \\
&= \ln s_j(t_0) + \ln\left[\frac{P_{\phi(j)}(0,s)}{P_{\phi(j)}(0,t)}\right] - \int_s^t q_j(u)du - \frac{1}{2}\int_s^t \sigma^S_j(u)\sigma^S_j(u)du \\
&\quad + \rho^{zs}_{0j}\int_s^t \alpha_0(u)H_0(u)\sigma^S_j(u)du - \epsilon_{\phi(j)}\rho^{sx}_{j\phi(j)}\int_s^t \sigma^S_j(u)\sigma_{\phi(j)}(u)du \\
&\quad + \frac{1}{2}\left(H^2_{\phi(j)}(t)\zeta_{\phi(j)}(t) - H^2_{\phi(j)}(s)\zeta_{\phi(j)}(s) - \int_s^t H^2_{\phi(j)}(u)\alpha^2_{\phi(j)}(u)du\right) \\
&\quad + (H_{\phi(j)}(t) - H_{\phi(j)}(s))z_{\phi(j)}(s) + \epsilon_{\phi(j)}\int_s^t \gamma_{\phi(j)}(u)(H_{\phi(j)}(t) - H_{\phi(j)}(u))du
\end{aligned}$$

The IR-IR covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$\begin{aligned}
\mathrm{Cov}[\Delta z_a, \Delta \ln x_b] &= \rho^{zz}_{0a}\int_s^t (H^z_0(t) - H^z_0)\,\alpha^z_0\,\alpha^z_a\,du \\
&\quad -\rho^{zz}_{ab}\int_s^t \alpha^z_a\,(H^z_b(t) - H^z_b)\,\alpha^z_b\,du \\
&\quad +\rho^{zx}_{ab}\int_s^t \alpha^z_a\,\sigma^x_b\,du.
\end{aligned}$$

The IR-FX covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$\begin{aligned}
\mathrm{Cov}[\Delta z_a, \Delta \ln x_b] &= \rho^{zz}_{0a}\int_s^t (H^z_0(t) - H^z_0)\,\alpha^z_0\,\alpha^z_a\,du \\
&\quad -\rho^{zz}_{ab}\int_s^t \alpha^z_a\,(H^z_b(t) - H^z_b)\,\alpha^z_b\,du \\
&\quad +\rho^{zx}_{ab}\int_s^t \alpha^z_a\,\sigma^x_b\,du.
\end{aligned}$$

The FX-FX covariance over the interval $[s,t] := [t_0, t_0+\Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$\begin{aligned}
\mathrm{Cov}[\Delta \ln x_a, \Delta \ln x_b] &= \int_s^t (H_0^z(t) - H_0^z)^2\, (\alpha_0^z)^2\, du \\
&\quad -\rho_{0a}^{zz} \int_s^t (H_a^z(t) - H_a^z)\, \alpha_a^z\, (H_0^z(t) - H_0^z)\, \alpha_0^z\, du \\
&\quad -\rho_{0b}^{zz} \int_s^t (H_0^z(t) - H_0^z)\, \alpha_0^z\, (H_b^z(t) - H_b^z)\, \alpha_b^z\, du \\
&\quad +\rho_{0b}^{zx} \int_s^t (H_0^z(t) - H_0^z)\, \alpha_0^z \sigma_b^x\, du \\
&\quad +\rho_{0a}^{zx} \int_s^t (H_0^z(t) - H_0^z)\, \alpha_0^z\, \sigma_a^x\, du \\
&\quad -\rho_{ab}^{zx} \int_s^t (H_a^z(t) - H_a^z)\, \alpha_a^z \sigma_b^x, du \\
&\quad -\rho_{ba}^{zx} \int_s^t (H_b^z(t) - H_b^z)\, \alpha_b^z\, \sigma_a^x\, du \\
&\quad +\rho_{ab}^{zz} \int_s^t (H_a^z(t) - H_a^z)\, \alpha_a^z\, (H_b^z(t) - H_b^z)\, \alpha_b^z\, du \\
&\quad +\rho_{ab}^{xx} \int_s^t \sigma_a^x\, \sigma_b^x\, du
\end{aligned}$$

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

$$\begin{aligned}
\mathrm{Cov}[\Delta x_a, \Delta z_{I,b}] &= \rho_{0b}^{zI} \int_s^t \alpha_0(s)(H_0(t) - H_0(s)) \alpha_{I,b}(s) ds \\
&\quad -\rho_{ab}^{zI} \int_s^t \alpha_a(s)(H_a(t) - H_a(s)) \alpha_{I,b}(s) ds \\
&\quad +\rho_{ab}^{xI} \int_s^t \sigma_a(s) \alpha_{I,b}(s) ds \\
\mathrm{Cov}[\Delta x_a, \Delta y_{I,b}] &= \rho_{0b}^{zI} \int_s^t \alpha_0(s)(H_0(t) - H_0(s)) H_{I,b}(s) \alpha_{I,b}(s) ds \\
&\quad -\rho_{ab}^{zI} \int_s^t \alpha_a(s)(H_a(t) - H_a(s)) H_{I,b}(s) \alpha_{I,b}(s) ds \\
&\quad +\rho_{ab}^{xI} \int_s^t \sigma_a(s) H_{I,b}(s) \alpha_{I,b}(s) ds
\end{aligned}$$

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

$$
\begin{aligned}
Cov\,[\Delta ln[s_i], \Delta ln[s_j]] &= \rho^{zz}_{\phi(i)\phi(j)} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_{\phi(j)}(t) \\
&\quad -H_{\phi(j)}(u))\alpha_{\phi(i)}(u)\alpha_{\phi(j)}(u)du \\
&\quad +\rho^{zs}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\sigma^S_j(u)du \\
&\quad +\rho^{zs}_{\phi(j)i} \int_s^t (H_{\phi(j)}(t) - H_{\phi(j)}(u))\alpha_{\phi(j)}(u)\sigma^S_i(u)du \\
&\quad +\rho^{ss}_{ij} \int_s^t \sigma^S_i(u)\sigma^S_j(u)du
\end{aligned}
$$

The equity/FX covariance over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
Cov\,[\Delta ln[s_i], \Delta ln[x_j]] &= \rho^{zz}_{\phi(i)0} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_0(t) - H_0(u))\alpha_{\phi(i)}(u)\alpha_0(u)du \\
&\quad -\rho^{zz}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))(H_j(t) - H_j(u))\alpha_{\phi(i)}(u)\alpha_j(u)du \\
&\quad +\rho^{zx}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\sigma_j(u)du \\
&\quad +\rho^{sz}_{i0} \int_s^t (H_0(t) - H_0(u))\alpha_0(u)\sigma^S_i(u)du \\
&\quad -\rho^{sz}_{ij} \int_s^t (H_j(t) - H_j(u))\alpha_j(u)\sigma^S_i(u)du \\
&\quad +\rho^{sx}_{ij} \int_s^t \sigma^S_i(u)\sigma_j(u)du
\end{aligned}
$$

The equity/IR covariance over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) is

$$
\begin{aligned}
Cov\,[\Delta ln[s_i], \Delta z_j] &= \rho^{zz}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\alpha_j(u)du \\
&\quad +\rho^{sz}_{ij} \int_s^t \sigma^S_i(u)\alpha_j(u)du
\end{aligned}
$$

The equity/inflation covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$
\begin{aligned}
Cov\left[\Delta ln[s_i], \Delta z_{I,j}\right] &= \rho^{zI}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\alpha_{I,j}(u)du \\
&\quad + \rho^{sI}_{ij} \int_s^t \sigma^S_i(u)\alpha_{I,j}(u)du \\
Cov\left[\Delta ln[s_i], \Delta y_{I,j}\right] &= \rho^{zI}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)H_{I,j}(u)\alpha_{I,j}(u)du \\
&\quad + \rho^{sI}_{ij} \int_s^t \sigma^S_i(u)H_{I,j}(u)\alpha_{I,j}(u)du
\end{aligned}
$$

The expectation of the Credit processes $z_{C,k}, y_{C,k}$ conditional on $\mathcal{F}_{t_0}$ at any time $t > t_0$ is equal to $z_{C,k}(t_0)$ resp. $y_{C,k}(t_0)$ since both processes are drift free.

The credit/credit covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$
\begin{aligned}
Cov\left[\Delta z_{C,a}, \Delta z_{C,b}\right] &= \rho^{CC}_{ab} \int_s^t \alpha_{C,a}(u)\alpha_{C,b}(u)du \\
Cov\left[\Delta z_{C,a}, \Delta y_{C,b}\right] &= \rho^{CC}_{ab} \int_s^t \alpha_{C,a}(u)H_{C,b}(u)\alpha_{C,b}(u)du \\
Cov\left[\Delta y_{C,a}, \Delta y_{C,b}\right] &= \rho^{CC}_{ab} \int_s^t \alpha_{C,a}(u)H_{C,a}(u)\alpha_{C,b}(u)H_{C,b}(u)du
\end{aligned}
$$

The IR/credit covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$
\begin{aligned}
Cov\left[\Delta z_a, \Delta z_{C,b}\right] &= \rho^{zC}_{ab} \int_s^t \alpha_a(u)\alpha_{C,b}(u)du \\
Cov\left[\Delta z_a, \Delta y_{C,b}\right] &= \rho^{zC}_{ab} \int_s^t \alpha_a(u)H_{C,b}(u)\alpha_{C,b}(u)du
\end{aligned}
$$

The FX/credit covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$
\begin{aligned}
\mathrm{Cov}[\Delta x_a, \Delta z_{C,b}] &= \rho^{zC}_{0b} \int_s^t \alpha_0(s)(H_0(t) - H_0(s))\alpha_{C,b}(s)ds \\
&\quad - \rho^{zC}_{ab} \int_s^t \alpha_a(s)(H_a(t) - H_a(s))\alpha_{C,b}(s)ds \\
&\quad + \rho^{xC}_{ab} \int_s^t \sigma_a(s)\alpha_{C,b}(s)ds \\
\mathrm{Cov}[\Delta x_a, \Delta y_{C,b}] &= \rho^{zC}_{0b} \int_s^t \alpha_0(s)(H_0(t) - H_0(s))H_{C,b}(s)\alpha_{C,b}(s)ds \\
&\quad - \rho^{zC}_{ab} \int_s^t \alpha_a(s)(H_a(t) - H_a(s))H_{C,b}(s)\alpha_{C,b}(s)ds \\
&\quad + \rho^{xC}_{ab} \int_s^t \sigma_a(s)H_{C,b}(s)\alpha_{C,b}(s)ds
\end{aligned}
$$

The inflation/credit covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$\begin{aligned}
\mathrm{Cov}[\Delta z_{I,a}, \Delta z_{C,b}] &= \rho^{IC}_{ab} \int_s^t \alpha_{I,a}\alpha_{C,b}(u) du \\
\mathrm{Cov}[\Delta z_{I,a}, \Delta y_{C,b}] &= \rho^{IC}_{ab} \int_s^t \alpha_{I,a} H_{C,b}(u)\alpha_{C,b}(u) du \\
\mathrm{Cov}[\Delta y_{I,a}, \Delta z_{C,b}] &= \rho^{IC}_{ab} \int_s^t \alpha_{I,a} H_{I,a}(u)\alpha_{C,b}(u) du \\
\mathrm{Cov}[\Delta y_{I,a}, \Delta y_{C,b}] &= \rho^{IC}_{ab} \int_s^t \alpha_{I,a} H_{I,a}(u)\alpha_{C,b}(u) H_{C,b}(u) du
\end{aligned}$$

The equity/credit covariances over the interval $[s,t] := [t_0, t_0 + \Delta t]$ (conditional on $\mathcal{F}_{t_0}$) are as follows:

$$\begin{aligned}
Cov\left[\Delta ln[s_i], \Delta z_{C,j}\right] &= \rho^{zC}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u)\alpha_{C,j}(u) du \\
&\quad + \rho^{sC}_{ij} \int_s^t \sigma^S_i(u)\alpha_{C,j}(u) du \\
Cov\left[\Delta ln[s_i], \Delta y_{C,j}\right] &= \rho^{zC}_{\phi(i)j} \int_s^t (H_{\phi(i)}(t) - H_{\phi(i)}(u))\alpha_{\phi(i)}(u) H_{C,j}(u)\alpha_{C,j}(u) du \\
&\quad + \rho^{sC}_{ij} \int_s^t \sigma^S_i(u) H_{C,j}(u)\alpha_{C,j}(u) du
\end{aligned}$$

A.3 Change of Measure

We can change measure from LGM to the T-Forward measure by applying a shift transformation to the H parameter of the domestic LGM process, as explained in [21] and shown in Example 12, section 5.12. This does not involve amending the system of SDEs above.

In the following we show how to move from the LGM to the Bank Account measure when we start with the Cross Asset Model in the LGM measure. This description and the implementation in ORE is limited so far to the cross currency case.

First note that the stochastic Bank Account (BA) can be written

$$B(t) = \frac{1}{P(0,t)} \exp\left(\int_0^t (H_t - H_s)\,\alpha_s\, dW^B_s + \frac{1}{2}\int_0^t (H_t - H_s)^2\,\alpha^2_s\, ds\right)$$

with Wiener processes in the BA measure. We can express this in terms of the domestic LGM's state variable $z(t)$ and an auxiliary random variable $y(t)$

$$B(t) = \frac{1}{P(0,t)} \exp\left(H(t)\,z(t) - y(t) + \frac{1}{2}\left(H^2(t)\,\zeta_0(t) + \zeta_2(t)\right)\right)$$

with

$$
\begin{aligned}
dz(t) &= \alpha(t)\,dW^B(t) - H(t)\,\alpha^2(t)\,dt \\
dy(t) &= H(t)\,\alpha(t)\,dW^B(t) \\
\zeta_n(t) &= \int_0^t \alpha^2(s)\,H^n(s)\,ds
\end{aligned}
$$

Note the drift of LGM state variable $z(t)$ in the BA measure and the auxiliary state variable $y(t)$ which is driven by the same Wiener process as $z(t)$. The instantaneous correlation of dz and dy is one, but the terminal correlation of $z(t)$ and $y(t)$ is less than one because of their different volatility functions. This is all we need to switch measure to BA in a pure domestic currency case.

To change measure in the cross currency case we need to make changes to the SDE beyond adding an auxiliary state variable y and adding a drift to the domestic LGM state. Let us write down the SDEs in the LGM and BA measure with respective drift terms that ensure martingale properties.

SDE in the LGM measure

$$
\begin{aligned}
dz_0 &= \alpha_0\,dW_0^z \\
dz_i &= \left(-\alpha_i^2\,H_i - \rho_{ii}^{zx}\,\sigma_i\,\alpha_i + \rho_{i0}^{zz}\,\alpha_i\,\alpha_0\,H_0\right)\,dt + \alpha_i\,dW_i^z \\
d\ln x_i &= \left(r_0 - r_i - \frac{1}{2}\sigma_i^2 + \rho_{0i}^{zx}\,\alpha_0\,H_0\,\sigma_i\right)\,dt + \sigma_i\,dW_i^x
\end{aligned}
$$

SDE in the BA measure

$$
\begin{aligned}
dy_0 &= \alpha_0\,H_0\,d\widetilde{W}_0^z \\
dz_0 &= -\alpha_0^2\,H_0\,dt + \alpha_0\,d\widetilde{W}_0^z \\
dz_i &= \left(-\alpha_i^2\,H_i - \rho_{ii}^{zx}\,\sigma_i\,\alpha_i\right)\,dt + \alpha_i\,d\widetilde{W}_i^z \\
d\ln x_i &= \left(r_0 - r_i - \frac{1}{2}\sigma_i^2\right)\,dt + \sigma_i\,d\widetilde{W}_i^x, \qquad r_i = f_i(0,t) + z_i(t)\,H_i'(t) + \zeta_i(t)\,H_i(t)\,H_i'(t)
\end{aligned}
$$

Blue terms are added, red terms are removed when moving from LGM to BA.

These drift term changes lead to the following changes in conditional expectations

$$
\begin{aligned}
\mathbb{E}[\Delta y_0] =& 0 \\
\mathbb{E}[\Delta z_0] =& -\int_s^t H_0\,\alpha_0^2\,du \\
\mathbb{E}[\Delta z_i] =& -\int_s^t H_i\,\alpha_i^2\,du - \rho_{ii}^{zx}\int_s^t \sigma_i^x\,\alpha_i\,du + \rho_{0i}^{zz}\int_s^t H_0\,\alpha_0\,\alpha_i\,du \\
\mathbb{E}[\Delta \ln x] =& (H_0(t)-H_0(s))\,z_0(s) - (H_i(t)-H_i(s))\;z_i(s) \\
&+ \ln\left(\frac{P_0^n(0,s)}{P_0^n(0,t)}\frac{P_i^n(0,t)}{P_i^n(0,s)}\right) \\
&- \frac{1}{2}\int_s^t (\sigma_i^x)^2\,du \\
&+ \frac{1}{2}\left(H_0^2(t)\,\zeta_0(t) - H_0^2(s)\,\zeta_0(s) - \int_s^t H_0^2\alpha_0^2\,du\right) \\
&- \frac{1}{2}\left(H_i^2(t)\,\zeta_i(t) - H_i^2(s)\,\zeta_i(s) - \int_s^t H_i^2\alpha_i^2\,du\right) \\
&+ \rho_{0i}^{zx}\int_s^t H_0\,\alpha_0\,\sigma_i^x\,du \\
&- \int_s^t (H_i(t)-H_i)\,\gamma_i\,du \qquad \text{with} \qquad \gamma_i = -\alpha_i^2\,H_i - \rho_{ii}^{zx}\,\sigma_i\,\alpha_i + \rho_{i0}^{zz}\,\alpha_i\,\alpha_0\,H_0 \\
&+ \int_s^t (H_0(t)-H_0)\;\gamma_0\,du \qquad \text{with} \qquad \gamma_0 = -H_0\,\alpha_0^2
\end{aligned}
$$

and the following additional variances and covariances

$$
\begin{aligned}
\mathrm{Var}[\Delta y_0] =& \int_s^t \alpha_0^2\,H_0^2\,du \\
\mathrm{Cov}[\Delta y_0, \Delta z_i] =& \rho_{0i}^{zz}\int_s^t \alpha_0\,H_0\,\alpha_i\,du \\
\mathrm{Cov}[\Delta y_0, \Delta \ln x_i] =& \int_s^t (H_0(t)-H_0)\,\alpha_0^2\,H_0\,du \\
&- \rho_{0i}^{zz}\int_s^t \alpha_0\,H_0\,(H_i(t)-H_i)\;\alpha_i\,du \\
&+ \rho_{0i}^{zx}\int_s^t \alpha_0\,H_0\,\sigma_i^x\,du
\end{aligned}
$$

Example 36 in section 5.36 illustrates the effect of the choice of measure on exposure simulations.

A.4 Exposures

In ORE we use the following exposure definitions

$$
EE(t) = EPE(t) = \mathbb{E}^N\left[\frac{(NPV(t)-C(t))^+}{N(t)}\right] \tag{8}
$$

$$
ENE(t) = \mathbb{E}^N\left[\frac{(-NPV(t)+C(t))^+}{N(t)}\right] \tag{9}
$$

where $NPV(t)$ stands for the netting set NPV and $C(t)$ is the collateral balance[11] 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 (8) 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

$$EE_B(t) = \mathbb{E}[\max(NPV(t) - C(t), 0)] \tag{10}$$

Expected Positive Exposure

$$EPE_B(T) = \frac{1}{T}\sum_{t<T} EE_B(t)\cdot \Delta t \tag{11}$$

Effective Expected Exposure, recursively defined as running maximum

$$EEE_B(t) = \max(EEE_B(t-\Delta t), EE_B(t)) \tag{12}$$

Effective Expected Positive Exposure

$$EEPE_B(T) = \frac{1}{T}\sum_{t<T} EEE_B(t)\cdot \Delta t \tag{13}$$

The last definition, Effective EPE, is used in Basel documents since Basel II for Exposure At Default and capital calculation. Following [12, 13] 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 (8) 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:

$$PFE_\alpha(t) = \left(\inf\{x|F_t(x) \geq \alpha\}\right)^+ \tag{14}$$

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.

[11] $C(t) > 0$ means that we have *received* collateral from the counterparty

A.5 CVA and DVA

Using the expected exposures in A.4 unilateral discretised CVA and DVA are given by [21]

$$CVA = \sum_i PD(t_{i-1}, t_i) \times LGD \times EPE(t_i) \tag{15}$$

$$DVA = \sum_i PD_{Bank}(t_{i-1}, t_i) \times LGD_{Bank} \times ENE(t_i) \tag{16}$$

where

$EPE(t)$	expected exposure (8)
$ENE(t)$	expected negative exposure (9)
$PD(t_i, t_j)$	counterparty probability of default in $[t_i; t_j]$
$PD_{Bank}(t_i, t_j)$	our probability of default in $[t_i; t_j]$
LGD	counterparty loss given default
LGD_{Bank}	our loss given default

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 [16]. This choice can be easily changed in the ORE source code or made configurable.
Moreover, formulas (15, 16) 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 is generaised and is applicable to calculations with dynamic credit

$$CVA^{dyn} = \sum_i \mathbb{E}^N \left[\frac{PD^{dyn}(t_{i-1}, t_i) \times PE(t_i)}{N(t)} \right] \times LGD \tag{17}$$

$$DVA^{dyn} = \sum_i \mathbb{E}^N \left[\frac{PD^{dyn}_{Bank}(t_{i-1}, t_i) \times NE(t_i)}{N(t)} \right] \times LGD_{Bank} \tag{18}$$

where

$PE(t)$	random variables representing positive exposure at $t : (NPV(t) - C(t))^+$
$NE(t)$	random variables representing negative exposure at $t : (-NPV(t) + C(t))^+$
$PD^{dyn}(t_i, t_j)$	random variables representing counterparty probability of default in $[t_i; t_j]$
$PD^{dyn}_{Bank}(t_i, t_j)$	random variables representing our probability of default in $[t_i; t_j]$
LGD	counterparty loss given default
LGD_{Bank}	our loss given default

A.6 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 (15,16), namely as an expectation of exposures times funding spreads:

$$
\begin{aligned}
FVA = & \underbrace{\sum_{i=1}^{n} f_l(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left\{ S_C(t_{i-1})\, S_B(t_{i-1})\, [-NPV(t_i) + C(t_i)]^+ \, D(t_i) \right\}}_{\text{Funding Benefit Adjustment (FBA)}} \\
& - \underbrace{\sum_{i=1}^{n} f_b(t_{i-1}, t_i)\, \delta_i\, \mathbb{E}^N \left\{ S_C(t_{i-1})\, S_B(t_{i-1})\, [NPV(t_i) - C(t_i)]^+ \, D(t_i) \right\}}_{\text{Funding Cost Adjustment (FCA)}} \qquad (19)
\end{aligned}
$$

where

$D(t_i)$	stochastic discount factor, $1/N(t_i)$ in LGM
$NPV(t_i)$	portfolio value at time t_i
$C(t_i)$	Collateral account balance at time t_i
$S_C(t_j)$	survival probability of the counterparty
$S_B(t_j)$	survival probability of the bank
$f_b(t_j)$	borrowing spread for the bank relative to OIS flat
$f_l(t_j)$	lending spread for the bank relative to OIS flat

For details see e.g. Chapter 14 in Gregory [19] and the discussion in [21].

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 (19) 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.7 COLVA

When the CSA defines a collateral compounding rate that deviates from the overnight rate, this gives rise to another value adjustment labeled COLVA [21]. In the simplest case the deviation is just given by a constant spread Δ:

$$COLVA = \mathbb{E}^N\left[\sum_i -C(t_i)\cdot\Delta\cdot\delta_i\cdot D(t_{i+1})\right] \tag{20}$$

where $C(t)$ is the collateral balance[12] 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.8 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 [21] that the following formula is a good approximation to the collateral floor value

$$\Pi_{Floor} = \mathbb{E}^N\left[\sum_i -C(t_i)\cdot(-r(t_i))^+\cdot\delta_i\cdot D(t_{i+1})\right] \tag{21}$$

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

$$\Pi_{Floor,\Delta} = \mathbb{E}^N\left[\sum_i -C(t_i)\cdot((r(t_i)-\Delta)^+ - r(t_i))\cdot\delta_i\cdot D(t_{i+1})\right] \tag{22}$$

[12]see A.4, $C(t) > 0$ means that we have *received* collateral from the counterparty

A.9 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 [22, 24]. 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 [21] in discrete form

$$MVA = \sum_{i=1}^{n} (f_b - s_I)\, \delta_i\, S_C(t_i)\, S_B(t_i) \times \mathbb{E}^N \left[DIM(t_i)\, D(t_i) \right]. \tag{23}$$

with borrowing spread f_b as in the FVA section A.6 and spread s_I received on initial margin, both spreads relative to the cash collateral rate.

A.10 KVA (CCR)

The KVA is calculated for the Counterparty Credit Risk Capital charge (CCR) following the IRB method concisely described in [20], 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):

$$M = \min\left(5, 1 + \frac{\sum\limits_{t_k > 1yr} EE_B(t_k) \Delta t_k B(0, t_k)}{\sum\limits_{t_k \le 1yr} EEE_B(t_k) \Delta t_k B(0, t_k)} \right)$$

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.11 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 [13, 14, 15]. ORE implements the *stand-alone* capital charge $SCVA$ for a netting set and computes a KVA for it[13]. 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.10 (for a bank using IMM to calculate EAD), but without applying a cap of 5;

[13]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.

- 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 $EEPE$.

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.10 at 1 for netting sets that contribute to CVA capital, if using the IRB approach for CCR capital.

A.12 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

$$CSA(t_m) = \begin{cases} \max(0, V_{set}(t_m) - I_A - T_{hold}), & V_{set}(t_m) - I_A \geq 0 \\ \min(0, V_{set}(t_m) - I_A + T_{hold}), & V_{set}(t_m) - I_A < 0 \end{cases} \tag{24}$$

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:

$$D(t_m) = \begin{cases} \Delta(t), & |\Delta(t)| \geq MTA \\ 0, & |\Delta(t)| < MTA \end{cases} \tag{25}$$

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 [17]. A more detailed framework for collateralised exposure modelling is introduced in the 2016 article [23], indicating a potential route for extending ORE.

In case we use an auxiliary "close-out" grid in the simulation (see section 7.4), we do *not* settle the delivery amount with the delay mentioned above. With a close-out grid the portfolio value is rather evolved relative to the collateral balance, also leading to residual exposure. However, the close-out valuation approach allows more detailed modelling of what happens in the close-out period; currently ORE supports two options, with and without portfolio "ageing" over the close-out period. The option "without ageing" is more aggressive in that it avoids any exposure evolution spikes due to contractual cashflows that occur in the close-out period after default, the only exposure effect is due to market evolution over the period. The "with ageing" option is more conservative in that it includes the effect of all contractual cash flows in the close-out period, in particular outgoing cashflows at any time in the period which cause an exposure jump upwards.

A.13 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 [18]. Allocation is done pathwise which first leads to allocated expected exposures and then to allocated CVA/DVA by inserting these exposures into equations (15,16). 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.14 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 31 illustrates for the one-dimensional case how shifts are applied.

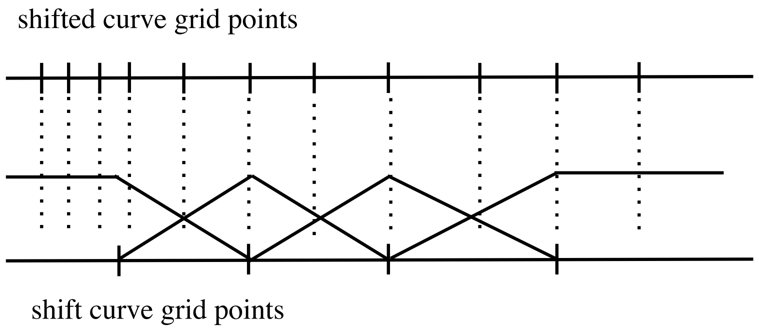


Figure 31: 1-d shift curve (bottom) applied to a more granular underlying curve (top).

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 73 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.

A.15 Value at Risk

For the computation of the parametric, or variance-covariance VaR, we rely on a second order sensitivity-based P&L approximation

$$\pi_S = \sum_{i=1}^{n} D^i_{T_i} V \cdot Y_i + \frac{1}{2} \sum_{i,j=1}^{n} D^{i,j}_{T_i,T_j} V \cdot Y_i \cdot Y_j \qquad (26)$$

with

- portfolio value V

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 73: 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

- 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 [25];
- 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 (26) ORE offers the techniques outlined below.

Delta Gamma Normal Approximation

The distribution of (26) 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 [26] 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 (26) 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 (26) 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://www.acadia.inc

[5] http://quantlib.org/install/vc10.shtml

[6] https://git-scm.com/downloads

[7] https://sourceforge.net/projects/boost/files/boost-binaries

[8] http://www.boost.org

[9] http://jupyter.org

[10] https://docs.continuum.io/anaconda

[11] http://www.libreoffice.org

[12] 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

[13] 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

[14] Basel Committee on Banking Supervision, *Review of the Credit Valuation Adjustment Risk Framework*, https://www.bis.org/bcbs/publ/d325.pdf, 2015

[15] Basel Committee on Banking Supervision, *Basel III: Finalising post-crisis reforms*, https://www.bis.org/bcbs/publ/d424.pdf, 2017

[16] Damiano Brigo and Fabio Mercurio, *Interest Rate Models: Theory and Practice, 2nd Edition*, Springer, 2006.

[17] Michael Pykhtin, *Collateralized Credit Exposure*, in Counterparty Credit Risk, (E. Canabarro, ed.), Risk Books, 2010

[18] 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

[19] Jon Gregory, *Counterparty Credit Risk and Credit Value Adjustment, 2nd Ed.*, Wiley Finance, 2013.

[20] Jon Gregory, *The xVA Challenge, 3rd Ed.*, Wiley Finance, 2015.

[21] 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.

[22] 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

[23] Leif B. G. Andersen, Michael Pykhtin, Alexander Sokol, *Rethinking Margin Period of Risk*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2719964, 2016

[24] 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

[25] 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

[26] Carol Alexander, Market Risk Analysis, Volume IV, Value at Risk Models, Wiley 2009