ORE User Guide

30 January 2026

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
16 September 2022	Acadia	updates for release 7
6 December 2022	Acadia	updates for release 8
31 March 2023	Acadia	updates for release 9
16 June 2023	Acadia	updates for release 10
16 October 2023	Acadia	updates for release 11
24 May 2024	Acadia	updates for release 12
January 2025	Acadia	separate user guide, methods, products
May 2025	LSEG	reorganise examples, updates for release 13
October 2025	LSEG	updates for release 14
January 2026	LSEG	updates for release 15

Contents

1 Introduction

The *Open Source Risk Project* [3] 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 [4].

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. And in the meantime, as ORE is used in risk services at industrial scale since 2018 with over 150 users, ORE is aimed at firms that consider the replacement of third party licensed software.

Contributions

Quaternion Risk Management [5] 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 [6] has been committed to continue the sponsorship. The Open Source Risk project work continues with former Quaternion operating as Acadia's Quantitative Services unit. And with Acadia's acquisiton by London Stock Exchange Group (LSEG) [7] in 2023, the journey continues under the roof of LSEG Post Trade Solutions.

The community is invited to contribute to ORE, through feedback, discussions and suggested enhancements. Our forum for that has moved to github, `https://github.com/OpenSourceRisk/Engine/discussions`. Issues can be reported at `https://github.com/OpenSourceRisk/Engine/Issues`. And contributions to the source code can be submitted via pull requests at https://github.com/OpenSourceRisk/Engine/pulls. See also the FAQ section on the ORE site [3] on how to get involved.

1.1 Scope

ORE currently provides portfolio pricing, cash flow generation, market risk analysis and a range of contemporary derivative portfolio analytics. The latter are based on a Monte Carlo simulation framework which yields the evolution of various 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 (xVA)

- 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 market risk framework provides

- sensitivity analysis, also in the "par" domain
- stress testing, also in the "par" domain
- several parametric VaR versions (Delta VaR, Delta-Gamma Normal VaR, Delta-Gamma VaR with Cornish-Fisher expansion and Saddlepoint method)
- historical simulation VaR
- P&L and P&L explain

across all asset classes and products.

Thanks to Acadia's open-source strategy, ORE's financial instrument scope was extended beyond the initial vanilla scope with a series of quarterly releases since version 7 to cover

- "First Generation" Equity and FX Exotics, released September with ORE v7
- Commodity products (Swaps, Basis Swaps, Average Price Options, Swaptions), released December 22 with ORE v8
- Credit products (Index CDS and Index CDS Options, Credit-Linked Swaps, Synthetic CDOs), released March 23 with ORE v9

- Bond products and Hybrids (Bond Options, Bond Repos, Bond TRS, Composite Trades, Convertible Bonds, Generic TRS with mixed basket underlyings, CFDs), released in June 23 with ORE v10
- a Scripted Trade framework in October 23 with ORE v11: This allows the modelling of complex hybrid payoffs such as Accumulators, TARFs, PRDCs, Basket Options, etc, across IR, FX, INF, EQ, COM classes. Scripted Trades are fully integrated into the market risk and exposure simulation frameworks, supported by American Monte Carlo methods for pricing and exposure simulation. The user can now extend the instrument scope conveniently by adding payoff scripts (embedded into the trade XML or in separate script "library" XML) and without recompiling the code base.
- Formula-based legs, Callable Swaps, Flexi Swaps, Balance Guaranteed Swaps and American Swaptions in May 24 with ORE v12

These contributions were accompanied by analytics extensions to enhance ORE usability

- Exposure simulation for xVA and PFE, adding Commodity to the asset class coverage, and adding American Monte Carlo for Exotics, released in December 22 with ORE v8
- Market Risk including multi-threaded sensitivity analysis, par sensitivity conversion, parametric delta/gamma VaR with Cornish-Fisher expansion and Saddlepoint method, released in March 23 with ORE v9
- Portfolio Credit Model, released in June 23 with ORE v10
- ISDA's Standard Initial Margin Model (SIMM), released in June 23 with ORE v10
- Historical Simulation VaR, P&L and P&L Explain, released in May 24 with ORE v12

Recent analytics additions are

- XVA Risk with sensitivities, P&L and P&L explain
- Regulatory Capital for CCR and CVA Risk (SA-CCR, BA-CVA, SA-CVA)
- Dynamic SIMM based on path-wise sensitivities computed with Algorithmic Differentiation (AAD)

The product coverage of the latest release of ORE is sketched in Table 1.

The simulation models applied in ORE's risk factor evolution implement the models discussed in detail in *Modern Derivatives Pricing and Credit Exposure Analysis* [17]: 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 (or Jarrow-Yildirim) models for inflation. The model components are calibrated to cross currency discounting and forward curves, Swaptions, FX Options, EQ Options and CPI caps/floors. With the 8th release, Commodity simulation has been added, as well as the foundation for a multi-factor Hull-White based IR/FX/COM simulation model.

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, Callable Swaps	Y	Y	Y	Y
Constant Maturity Swaps, CMS Caps/Floors	Y	Y	Y	Y
FX Forwards and Average Forwards	Y	Y	Y	Y
Cross Currency Swaps	Y	Y	Y	Y
FX European and Asian Options	Y	Y	Y	Y
FX Exotic Options (see below)	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 Exotic Options (see below)	Y	Y	Y	Y
Equity Future Options	Y	Y	Y	Y
Commodity Forwards and Swaps	Y	Y	Y	Y
Commodity European and Asian Options	Y	Y	Y	Y
Commodity Digital Options	Y	Y	Y	Y
Commodity Swaptions	Y	Y	Y	Y
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, Options	Y	Y	N	Y
Index Credit Default Swaps, Options	Y	Y	N	Y
Credit Linked Swaps	Y	Y	N	Y
Index Tranches, Synthetic CDOs	Y	Y	N	Y
Composite Trades	Y	Y	Y	Y
Total Return Swaps and Contracts for Difference	Y	Y	Y	Y
Convertible Bonds	Y	Y	Y	N
ASCOTs	Y	Y	Y	Y
Scripted Trades	Y	Y	Y	Y
Flexi Swaps and Balance Guaranteed Swaps	Y	Y	Y	Y

Table 1: ORE product coverage. FX/Equity Exotics include Barrier, Digital, Digital Barrier (FX only), Double Barrier, European Barrier, KIKO Barrier (FX only), Touch and Double Touch Options, Outperformance options and Pairwise Variance Swaps. Scripted Trades cover single and multi-asset products across all asset classes except Credit (so far), see Example_52 and the separate documentation in Docs/ScriptedTrade.

1.2 ORE in Python or Java

ORE is written in C++ and comes with a command line executable `ore.exe` that supports batch processes. But since early versions of ORE we also provide language bindings following QuantLib's example using SWIG, in ORE's case with focus on Python and Java modules. The ORE SWIG module extends (contains) the QuantLib SWIG module and offers moreover access to a part of ORE's functionality. Since ORE v9, Python *wheels* are provided for each release, so that users can install the most recent ORE Python module by calling

```
pip install open-source-risk-engine
```

See section 4.15 on how to use ORE-Python.

Note that, technically, the ORE SWIG module source code has moved into the ORE Engine repository with release 13, see directory ORE-SWIG there. And we have removed the separate repository formerly located at https://github.com/OpenSourceRisk/ORE-SWIG.

1.3 Roadmap

ORE grows with community contributions and the demand of clients who utilise ORE to replace existing risk applications with the help of the Quant Services team at Acadia/LSEG.

Moreover, it is generally planned that subsequent ORE releases will extend the scope of the **Regulatory Capital** analytics in ORE, with

- broader product scope of SA-CCR and BA-CVA
- addition of Market Risk capital (FRTB-SA)

Performance: ORE v12 contains applications of **AAD** for sensitivity analysis, CVA sensitivity (proof-of concept stage), as well as applications of **GPUs** to parallelize computations (see legacy Examples 56 and 61), with significant speed-ups. Both areas are active work in progress, and further enhancements and tests should be released with the next versions.

ORE Python: Moreover, there is demand among ORE users for extended coverage of the ORE-Python version, so that we also expect steady growth of the Python wrapper around ORE.

ORE Service: ORE v12 contains an implementation of a restful API around ORE, see folder ore/Api and the example therein. This is written in Python, uses the ORE-Python module and the Flask web framework. This implementation is proof of concept, for demonstration purposes and to encourage the community to extend and contribute alternatives.

1.4 Further Resources

- Open Source Risk Project site: `http://www.opensourcerisk.org`
- Source code and releases: `https://github.com/opensourcerisk/engine`
- Frequently Asked Questions: `http://www.opensourcerisk.org/faqs`
- Follow ORE on Twitter `@OpenSourceRisk` for updates on releases and events
- ORE Product Catalogue [1]
- ORE Methodology [2]
- ORE Academy on youtube: `https://www.youtube.com/channel/UCrCpkb1-s3pxKd7U-YgJulA`

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 2 and installation instructions in section 3 we start in section 4 with a series of examples that illustrate how to launch ORE using its command line application, and we discuss typical results and reports.

The description of products, pricing and trade representration in ORE XML has been carved out and moved to [1].

And finally, a summary of methodologies applied in ORE can be found in [2].

2 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).

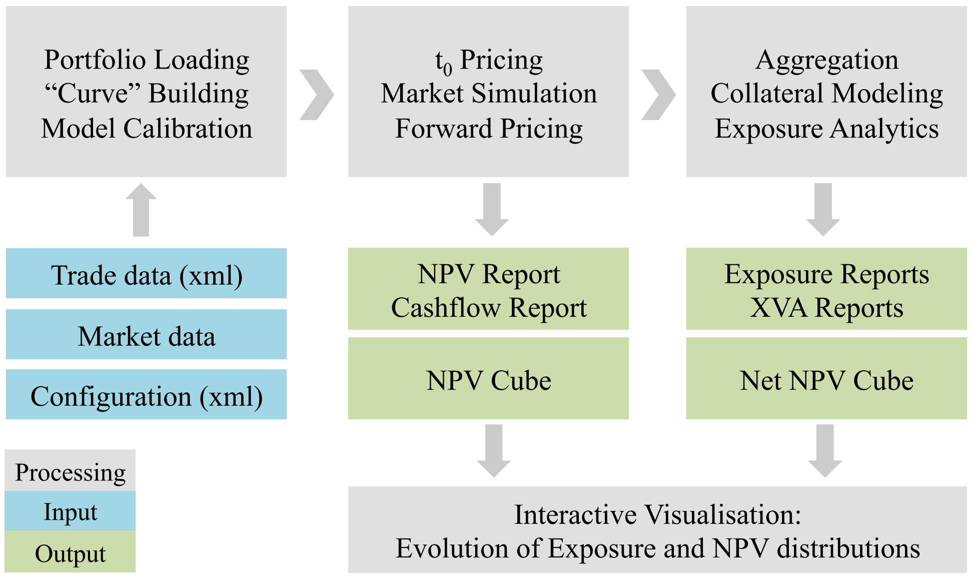

Figure 1: Sketch of the ORE process, inputs and outputs.

The overall ORE process needs to be parametrised using a set of configuration XML files which is the subject of section 5. The portfolio is provided in XML format which is explained in detail in [1] and 8. 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 9.
The first processing step (upper left box) then comprises

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

The second processing step (upper middle box) is then

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

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

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

The outputs of this process step are XVA reports and the 'net' NPV cube, i.e. after aggregation, netting and collateral.
The example section 4 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'.

3 Getting and Building ORE

ORE's source code is hosted at `https://github.com/opensourcerisk/engine`.

You can get ORE sources in two ways, either by downloading a release bundle as described in section 3.1 or by checking out the source code from the github repository as described in section 3.2. In both cases, the user usually needs to build (compile) from sources in a platform dependent way. To avoid that build process, one can either get the pre-built ORE Python module as mentioned in section 1.2, or try the executables built on github automatically for a few platforms for testing purposes, see the "Actions" menu.

3.1 ORE Releases

ORE releases are snapshots of the ORE codebase, regularly provided in the form of source code archives. These archives are provided at `https://github.com/opensourcerisk/engine/releases` under the "Assets" heading for each release, accessible via the download links

- **Source code** (zip) and
- **Source code** (tar.gz)

Additional "assets" include pdf documentation for convenience (built from tex sources in the release) such as user guide, product and methodology catalogue, or the scripted trade guide.

Unpacking the downloaded release archive creates a directory `Engine-<VERSION>` (because the repository is called Engine - feel free to rename this to "ore" after unpacking) with the following files resp. subdirectories

1. `App/`

2. `cmake/`
3. `CMakeLists.txt`
4. `CMakePresets.json`
5. `cmake/`
6. `Docker/`
7. `Docs/`
8. `Examples/`
9. `FrontEnd/`
10. `OREAnalytics/`
11. `OREData/`
12. `ORETest/`
13. `ORE-SWIG/`
14. `QuantExt/`
15. `QuantLib/`
16. `ThirdPartyLibs/`
17. `Tools/`
18. `Tutorials/`
19. `xsd/`

Note:

- Each release also contains the QuantLib source version that ORE depends on; this is usually the latest QuantLib release that precedes the ORE release including a small number of patches.
- Since release 13 ORE-SWIG sources are included in the Engine repository, and ORE-SWIG contains QuantLib-SWIG

3.2 Access via Git

To access the evolving code base on GitHub or specific versions conveniently, one should clone ORE's git repository.

1. Install Git on your machine following instructions at [9]
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.

The above fetches ORE from the master branch, if you want to fetch a specific release then right after `% cd ore` do `% git checkout tags/release_name`, e.g. `% git checkout tags/v1.8.12.0`.

3.3 Prerequisites for Building ORE

To build ORE from sources one needs to install the following collection of tools

- C++ compiler
- Boost libraries
- CMake
- Ninja (optional)
- zlib (optional)
- Eigen
- Swig
- Python

where the latter two are required to build the ORE Python module.

C++ Compiler

QuantLib and ORE are written in C++, so we need a C++ compiler – this comes on Windows with Visual Studio, it is gcc on Linux, clang on macOS.

On Windows, download and install Visual Studio Community Edition (Version 2019 or later, 2022 is recommended), and during the installation, make sure to

- install the Visual C++ support under the Programming Languages features (disabled by default)
- install the feature 'C++ CMake Tools for Windows'

CMake

The ORE build requires CMake (https://cmake.org), to be installed on Linux and macOS using the usual package managers (apt, homebrew). Visual Studio has CMake support in VS2019 or later, and you can install the feature 'C++ CMake Tools for Windows' with VS instead of installing CMake as standalone program on Windows.

Boost

QuantLib and ORE depend on the Boost C++ libraries, to be installed before kicking off the build process. At least Boost version 1.75.0 is required.

On Linux and macOS, one can install boost conveniently using apt or homebrew

```
sudo apt install libboost-all-dev
brew install boost
```

Otherwise, when building boost following instructions at [11] and when installing boost in a non-standard path, take note of the boost include and library directories on your system. These paths need to be passed to the CMake configure step, see below.

On Windows:

1. Download the pre-compiled binaries for your MSVC version (e.g. MSVC-14.3 for MSVC2022) from [10]
   - 32-bit: [10]\VERSION\boost_VERSION-msvc-14.3-32.exe\download
   - 64-bit: [10]\VERSION\boost_VERSION-msvc-14.3-64.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: VS2022 x86 Native Tools Command Prompt
   - 64-bit: VS2022 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-14.3 \
     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-14.3 \
     address-model=64 --with-test --with-system --with-filesystem \
     --with-serialization --with-regex --with-date_time stage
     ```

Configure boost paths, setting environment variables, e.g.:

- `%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\lib32msvc14.3`

- `%BOOST_LIB64%` pointing to your x64 lib directory, e.g, `C:\boost_1_72_0\lib64msvc14.3`

Compiler / Boost Versions

Ensure consistency of compiler and boost version: The following table 2 reflects the compiler / boost version combinations that the users/developers at Quaternion/Acadia/LSEG can confirm as working combinations with the latest ORE v13.

Compiler	Boost	ORE
AppleClang version 15.0.0	1.83.0	13
AppleClang version 15.0.0	1.86.0	13
AppleClang version 15.0.0	1.88.0	13
clang 19.1.7	1.83.0	13
VS2022	1.86.0	13

Table 2: Supported compiler and boost versions for ORE v13, every boost version between 1.72. and 1.86 should work on Windows.

Ninja

The installation of Ninja is optional.

When running the build process in the command line on a Linux or macOS system, the default "generator" is `GNU make` which is expected to be available on the system. As an alternative and for build speed we recommend `Ninja` (`https://ninja-build.org`). Moreover, Ninja is also available on Windows, so that we can run command line builds with same commands on Windows, Linux and macOS.

However, Windows users can rely on Visual Studio's collaboration with CMake (see below) without additional generator.

zlib

ORE can use zlib for writing compressed data, to be installed on Linux or macOS using the usual package manager. It can be installed on Windows with the open source c++ library manager VCPKG. To get VCPG, see `https://vcpkg.io/en/getting-started.html`. And to install ZLIB with VCPKG:

```
vcpkg install --triplet x64-windows zlib
```

To make VCPKG visible to CMake, create an environment variable `VCPKG_ROOT` pointing to the root of the vcpkg directory and configure ORE with the flag

`-DCMAKE_TOOLCHAIN_FILE=%VCPKG_ROOT%/scripts/buildsystems/vcpkg.cmake`.

To use VCPKG with Visual Studio add the toolChainFile to the configurePresets in the CMakePresets.json:

```
"toolchainFile":  "$env{VCPKG_ROOT}/scripts/buildsystems/vcpkg.cmake",
```

To enable ZLIB support in ORE, configure CMake with the flag `-DORE_USE_ZLIB=ON`.

Eigen

ORE requires the Eigen libraries (https://gitlab.com/libeigen/eigen) for some Credit Risk analytics. On Windows, one can conveniently install using VCPKG with:

```
vcpkg install --triplet x64-windows eigen3
```

Swig and Python

Finally, to also build the ORE Python module from sources, install `swig` (https://swig.org) and Python (version 3.5 or higher). The latter is also required to run the examples in section 4 or to use the pre-built ORE Python module in section 4.15.

3.4 Building ORE

ORE uses the CMake build system with the following essential parameters (see ore/CMakeLists.txt, ore/QuantExt/CMakeLists.txt, ore/QuantLib/CMakeLists.txt) which affect the build:

Option	Description	Default
ORE_BUILD_DOC	Build PDF documentation	ON
ORE_BUILD_EXAMPLES	Build examples	ON
ORE_BUILD_TESTS	Build test suites	ON
ORE_BUILD_APP	Build the ORE commandline application `ore[.exe]`	ON
ORE_BUILD_SWIG	Build the ORE Python module	ON
ORE_ENABLE_OPENCL	Build the compute environment using OpenCL, to utilise a GPU	OFF
ORE_ENABLE_CUDA	Build the compute environment using CUDA, to utilise a GPU	OFF
ORE_PYTHON_INTEGRATION	Build ORE with Python Integration, allows calling from C++ into Python	OFF
ORE_USE_ZLIB	Use compression for boost::iostreams, e.g. to write compressed cube files	OFF
ORE_MULTITHREADING _CPU_AFFINITY	Set cpu affinitity in multithreaded calculations	OFF
ORE_ENABLE_PARALLEL _UNIT_TEST_RUNNER	Enable the parallel unit test runner	OFF
MSVC_LINK_DYNAMIC_RUNTIME	Link against dynamic runtime	ON
MSVC_PARALLELBUILD	Use flag /MP	ON
QL_USE_PCH	Use precompiled headers	OFF
QL_ENABLE_SESSIONS	Singletons return different instances for different sessions; turn ON for ORE's multi-threading	OFF
QL_BUILD_EXAMPLES	Build QuantLib examples	ON
QL_BUILD_TEST_SUITE	Build QuantLib test suite	ON
BOOST_INCLUDEDIR, BOOST_ROOT	directory containing the "boost" folder, root of all boost includes	user-provided if not found by CMake
BOOST_LIBRARYDIR	folder containing compiled boost libraries	user-provided if not found by CMake
Python_ROOT	directory containing Python.h	user-provided if not found by CMake
Python_LIBRARY	Python shared library name including full path	user-provided if not found by CMake

The latter BOOST and Python variables have been added to the list, because CMake may use them to locate the BOOST and Python development files, if CMake does not find them automatically.

Linux and macOS

To run CMake we first create a directory `build` in the ORE root directory, and change to `build`.

The CMake build system is then configured by calling `cmake` in the command line with a series of command line parameters `-D` which allow overriding/setting the essential parameters listed above:

```
cmake ..  \
     -D QL_ENABLE_SESSIONS=ON \
     -D QL_BUILD_EXAMPLES=OFF \
     -D ORE_USE_ZLIB=ON \
     [ -D BOOST_INCLUDEDIR=<path/to/boost/includes> \ ]
     [ -D BOOST_LIBRARYDIR=<path/to/boost/libraries> \ ]
     [ -D Python_ROOT=<directory containing Python.h> \ ]
     [ -D Python_LIBRARY=<path/to/PythonLibrary> \ ]
     [ -G Ninja ]
```

Then build ORE with calling `make` or `ninja` depending on the configuration above.

The default CMake build includes the ORE Python module already (since `ORE_BUILD_SWIG = ON` by default) which can be used to run ORE Python locally as shown in section 4.15. There are alternative ways of building ORE Python, in particular building Python wheels for distribution. Section 3.5 points to related tutorials.

Windows

Visual Studio 2019 and later supports CMake Projects.

1. Start Visual Studio 2019 or later.
2. Select "Open a local folder" from the start page or menu.
3. In the dialog window, select the ORE root directory.
4. Visual Studio will read the cmake presets from CMakePresets.json and the project file CMakeList.txt and configure the project.
5. Once the configuration is finished and one can build the project.
6. The executables are built in the subfolder `/build/TARGET/CONFIGURATION/EXECUTABLE`, e.g. `/build/App/Release/ore.exe`.

ORE is shipped with configuration and build presets using Visual Studio 2022 and the Ninja build system. Those presets are configured in the CMakePreset.json which is read by Visual Studio by default when opening the CMake project. If you want to use Visual Studio 2019 instead, you would have to change the Generator in the CMakePreset.json from "Visual Studio 17 2022" to "Visual Studio 16 2019".

You can switch in the solution explorer from the file view to the projects view, where the CMake Targets View can be selected. In this view, the various target projects can be seen below "ORE Project" and be used in a similar manner as the usual VS projects.

Alternatively, Visual Studio project files can be auto-generated from the CMake project files or ORE can be built with the CMake command line tool, similar to UNIX / Mac systems.

1. Generate MSVC project files from CMake files:
   - Open a Visual Studio Tools Command Prompt
     - 32-bit: VS2022/x86 Native Tools Command Prompt for VS 2022
     - 64-bit: VS2022/x64 Native Tools Command Prompt for VS 2022
   - Navigate to the ORE root directory
   - Run CMake command:
     - 64-bit:
       `cmake -G "Visual Studio 17 2022" -A x64 -DBOOST_INCLUDEDIR=%BOOST% -DBOOST_LIBRARYDIR=%BOOST_LIB64% -DQL_ENABLE_SESSIONS=ON -DMSVC_LINK_DYNAMIC_RUNTIME=true -B build`
     - 32-bit:
       `cmake -G "Visual Studio 17 2022" -A x32 -DBOOST_INCLUDEDIR=%BOOST% -DBOOST_LIBRARYDIR=%BOOST_LIB32% -DQL_ENABLE_SESSIONS=ON -DMSVC_LINK_DYNAMIC_RUNTIME=true -B build`

   Replace the generator "Visual Studio 17 2022" with the actual installed version. The solution and project files will be generated in the ⟨`ORE_ROOT`⟩`\build` subdirectory.

2. build the cmake project with the command `cmake --build build -v --config Release`,

3. or open the MSVC solution file `build\ORE.sln` and build the entire solution with Visual Studio (again, make sure to select the correct platform in the configuration manager first).

3.5 Building ORE Python Wheels

The ORE Python module is part of the CMake build process by default, see above.

However, to build Python wheels (modules for distribution) we use Python's setup.py approach. See the tutorials at
`https://github.com/OpenSourceRisk/Engine/blob/master/tutorials_index.md`.

The wheels for various platforms and Python versions that get distributed via `https://pypi.org` (so that users can do a "pip install open-source-risk-engine") are built on github using github workflows, see ore/.github/workflows and related actions).

Typical usage of the Python module is shown in ORE's `Examples/ORE-Python` directory and described in user guide section 4.15.

3.6 ORE with Python Integration

This section describes a new (still experimental) feature of ORE that supports calls from ORE into Python in order to utilise some functionality that is readily available in Python so that we can avoid its implementation in C++ in "core ORE", e.g. for testing purposes. An example is the use of regression methods provided in the `scikit-learn` and `py-earth2` Python modules, as demonstrated in Example/InitialMargin where we have provided an option to utilise such methods embedded into American Monte Carlo as alternatives for the usual polynomial regression provided by QuantLib.

Prerequisites for using this functionality are

- building ORE with the `ORE_PYTHON_INTEGRATION` switch set to `ON`, see section 3.4;
- specifying the `Python_ROOT` and `Python_LIBRARY` variables when calling `cmake` so that cmake finds the Python installation on your system, as in the case of building Python wheels 3.5;
- extending the `PYTHONPATH` environment variable to contain directory `ore/PythonIntegration` where ORE will be looking for the script `ore_python_integration.py` that contains the Python functions that can be called from the C++ side of ORE.

See the Jupyter notebook Examples/InitialMargin/ore_dynamicsimm.ipynb for a demonstration of this new feature.

4 Examples

Over 80 Examples of ORE usage have been compiled since the first release of ORE, in order to help users getting started and to serve as plausibility checks for the results generated with ORE.

The examples a grouped by topic as shown in table 3 below. The structure of the `ore/Examples` folder follows the same structure as this User Guide section.

Change to each subsection below and the associated Examples subdirectory to learn more about the list of cases.

Example topic	Description	Contained Legacy Examples
Academy	Basic ORE Academy examples, with minimal configuration	
Minimal Setup	Example showing the minimal configuration required for pricing (market data, todays market, curve config, conventions)	14
Products	Demonstrate ORE's product coverage across asset classes, 150+ products, vanilla and complex	18-21, 27, 45-48, 51, 65-66, 71, 74
Curve Building	Consistency check that market instruments are repriced correctly after bootstrap, demo several curve building features	26, 28-30, 49, 53, 59
Market Risk	Sensitivity, Stress Testing, Par Conversion, Parametric and HistSim VaR, P&L and P&L Explain, Market Risk Capital (SMRC)	15, 22, 40, 50, 57-58, 62-63, 68-69, 77
Initial Margin	ISDA SIMM and IM Schedule, Dynamic Initial Margin via Regression and Dynamic Delta VaR	13, 44
Exposure	Uncollateralised exposure simulation, product by product, mainly vanilla, demo various simulation features	1-9, 11-12, 16-17, 23-25, 32-38, 64
Exposure with Collateral	Impact of Variation Margin and CSA details, Impact of Initial Margin	10, 31, 72
Scripted Trade	Introducing the Scripted Trade framework	52
American Monte Carlo	Fast and accurate Exposures using AMC, selected vanilla and complex products	39, 54, 55, 60, 73, 75
XVA Risk	Sensitivity, Stress Testing, P&L Explain, CVA Capital (SA-CVA and BA-CVA)	67, 68, 70
Credit Risk	Credit Portfolio Model, Derivatives Credit Risk Capital (SA-CCR)	43, 68
Performance	NPV and CVA Sensitivities using AAD and using bump & reval with GPU parallelization	41, 56, 61
ORE-Python	Python Wrapper covering ORE libraries and QuantLib	42, and the Python examples previously in the ORE-SWIG repository
ORE-API	ORE Web Service prototype using ORE-Python	0

Table 3: ORE example topics.

All example results can be produced with the Python scripts `run.py` in the ORE Example subdirectories 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 5.

ORE is driven by a number of input files, listed in table 4 and explained in detail in sections 5 to 10. In all examples, these input files are either located in the example's sub directory `Examples/Example_#/Input` or the main input directory `Examples/Input` if used across several examples. The particular selection of input files is determined by the 'master' input file `ore.xml`.

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 4: ORE input files

The typical list of output files and reports is shown in table 5. 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`.

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 collateral, in readable text format
`*.csv.gz`	Intermediate storage of NPV cube and scenario data
`*.pdf`	Exposure graphics produced by the python script `run.py` after ORE completed

Table 5: ORE output files

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.

4.1 Testsuites

The build includes a set of testsuites for each of QuantLib, QuantExt, OREData, and OREAnalytics libraries. You can run all tests by navigating into the Build folder and running the following command: `ctest -C Release -j4 -timeout 3600`

Python

Running any of the examples described in the following sections needs a python3 installation.

4.2 Academy

This section contains a few examples with minimal configuration for demo cases shown in ORE Academy videos on youtube:
https://www.youtube.com/channel/UCrCpkb1-s3pxKd7U-YgJulA

4.3 Minimal Setup

The example in folder `MinimalSetup` demonstrates using a minimal market data setup in order to run the "opening" vanilla Swap exposure simulation in section 4.8. The minimal market data uses single points per curve where possible.

Run with: `python run.py`

4.4 Products

This example contains 150+ sample trades in ORE XML across six asset classes, vanilla and complex, see subfolder `Example_Trades`. The sample trades have also been concatenated into a single portfolio in `Input/portfolio.xml`.

The Input folder contains the necessary configuration and market data to support a valuation batch. Run it with: `python run.py`

See the **ORE Product Catalogue** in `Docs/UserGuide/products.tex|pdf` for payoff descriptions, input guide and pricing methods.

In addition to the `npv` and `cashflow` analytic we have selected the `portfolioDetails` analytic in `Products/Input/ore.xml`:

```xml
<Setup>
   ...
</Setup>
<Analytics>
  <Analytic type="npv">
    <Parameter name="active">Y</Parameter>
    <Parameter name="baseCurrency">USD</Parameter>
    <Parameter name="outputFileName">npv.csv</Parameter>
    <Parameter name="additionalResults">Y</Parameter>
    <Parameter name="additionalResultsReportPrecision">12</Parameter>
  </Analytic>
  <Analytic type="cashflow">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">flows.csv</Parameter>
  </Analytic>
  <Analytic type="portfolioDetails">
    <Parameter name="active">Y</Parameter>
    <Parameter name="riskFactorFileName">riskFactors.csv</Parameter>
    <Parameter name="marketObjectFileName">marketObjects.csv</Parameter>
    </Analytic>
</Analytics>
```

This leads to the reporting of portfolio composition in a series of extra files in folder `Products/Output`:

- counterparties.csv
- marketObjects.csv
- netting_sets.csv
- riskFactors.csv
- swap_indices.csv
- trade_types.csv
- underlying_indices.csv

4.5 Curve Building

This directory demonstrates several curve building cases

- Bootstrap Consistency: `python run_consistency.py`
- Discount Ratio Curves: `python run_discountratio.py`
- Fixed vs Float Cross Currency Helpers: `python run_fixedfloatccs.py`
- USD-Prime Curve Building via Prime-LIBOR Basis Swaps: `python run_prime.py`
- Bond Yield Shifted Curves: `python run_bondyieldshifted.py`
- Central Bank Meeting Dates: `python run_dentralbank.py`
- SABR Swaption and Cap/Floor Volatilities: `python run_sabr.py`

Run all with: `python run.py`.

4.5.1 Bootstrap Consistency

This example 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)

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

```
python TradeGenerator.py
```

at the `Examples/CurveBuilding` level.

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

```
python run_consistency.py
```

at the `Examples/CurveBuilding` level.

Results are in three files `*_npv.csv` in `Examples/CurveBuilding/Output/consistency` and should show zero NPVs for all benchmark instruments.

4.5.2 Discount Ratio Curves

This example 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 collateralization scenarios can be run calling `python run.py`.

4.5.3 Fixed vs Float Cross Currency Helpers

This example 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_fixedfloatccs.py` at the `Examples/CurveBuilding` level or "directly" calling `ore[.exe] ore.xml`.

4.5.4 USD-Prime Curve Building

This example 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/CurveBuilding` with `python run_prime.py` yields the USD-Prime curve in `Examples/CurveBuilding/Output/prime/curves.csv`.

4.5.5 Bond Yield Shifted Curves

This example shows how to use a yield curve built from a BondYieldShifted segment, as described in section 5.7.1.

In particular, it builds the curve `USD.BMK.GVN.CURVE_SHIFTED` shifted by three liquid Bonds:

- Fixed rate USD Bond maturing in August 2023 with id `EJ7706660`.
- Fixed rate USD Bond maturing in September 2049 with id `ZR5330686`.
- Floating Rate Bond maturing in May 2025 with id `AS064441`.

The resulting curve is exhibited in the `curves.csv` output file. Moreover, the results can be cross checked against the NPVs, i.e. prices, of the ZeroBonds comprised in the portfolio.

- `ZeroBond_long`, maturing 2052-03-01 shows a price of 0.2080 akin to the 0.2080 in the curves output at the same date.
- `ZeroBond_short`, maturing 2032-06-01 shows a price of 0.5808 aktin to the 0.808 in the curves output at the same date.

The example can be run calling `python run_bondyieldshifted.py` at the `Examples/CurveBuilding` level.

4.5.6 Central Bank Meeting Dates

This example demonstrates the build of a GBP OIS curve using MPC Swaps at the short end, i.e. using OIS Swaps with concrete forward start and end date.

4.5.7 SABR Volatility Surfaces

This example demonstrates the pricing of a Swaption and a Cap on volatility surfaces that are interpolated in smile direction using a SABR model flavour. As usual the example is run by calling `python run.py`

The essential configuration is in `curveconfig.xml` where the Interpolation (Swaption) resp. StrikeInterpolation (Caps/Floors) allows the following new SABR types

- Hagan2002Lognormal
- Hagan2002Normal
- Hagan2002NormalZeroBeta
- Antonov2015FreeBoundaryNormal

- KienitzLawsonSwaynePde
- FlochKennedy

SABR parameters can be calibrated or have fixed externally provided values per option tenor and Swap tenor (Swaptions) resp. optionlet (Caps/Floors).

4.6 Market Risk

This directory demonstrates the following collection of market risk analytics:

- Sensitivity analysis in the "raw" and "par" domain: `python run_sensi.py`
- Sensitivity analysis when pricing with smile: `python run_sensismile.py`
- parametric VaR: `run_parametricvar.py`
- Stress testing in the "raw" domain: `run_stress.py`
- Stress testing in the "par" domain: `run_parstress.py`
- Stressed Sensitivity analysis: `python run_sensistress.py`
- Historical Simulation VaR: `run_histsimvar.py`
- Simple Market Risk Capital (SMRC): `run_smrc.py`
- P&L and P&L Explanation: `run_pnlexplain.py`
- Par conversion utility: `run_parconversion.py`
- Base scenario utility: `run_basescenario.py`
- Zero to par shift utility: r̂un_zerotoparshift.py

Run all with: `python run.py`.

4.6.1 Sensitivity Analysis

This example (`python run_sensi.py`) demonstrates the calculation of sensitivities for a portfolio consisting 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.

The sensitivities are first computed in the "raw" (e.g. zero rate and optionlet) domain. The "raw" 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 5.4.

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 are pure differences, i.e. they are not divided by the shift size.

The second file also contains second order mixed differences according to the specified cross gamma filter, along with the shift sizes for the two factors involved.

Raw sensitivities are then converted into the "par" domain (e.g. Swap rates, CDS spreads) via a Jacobi transformation. See a sketch of the methodology in [2] and section 5.4 for configuration details.

To perform a par sensitivity analysis, the following extension in `ore.xml` is required

```
  <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">sensi_scenarios.csv</Parameter>
    <Parameter name="sensitivityOutputFile">sensitivity.csv</Parameter>
    <Parameter name="outputSensitivityThreshold">0.000001</Parameter>
    <!-- Additional parametrisation for par sensitivity analysis -->
    <Parameter name="parSensitivity">Y</Parameter>
    <Parameter name="parSensitivityOutputFile">parsensitivity.csv</Parameter>
    <Parameter name="outputJacobi">Y</Parameter>
    <Parameter name="jacobiOutputFile">jacobi.csv</Parameter>
    <Parameter name="jacobiInverseOutputFile">jacobi_inverse.csv</Parameter>
  </Analytic>
```

The usual "raw" sensitivity analysis is performed by bumping the "raw" rates (zero rates, hazard rates, inflation zero rates, optionlet vols). This is followed by the Jacobi transformation that turns "raw" sensitivities into sensitivities in the par domain (Deposit/FRA/Swap rates, FX Forwards, CC Basis Swap spreads, CDS spreads, ZC and YOY Inflation Swap rates, flat Cap/Floor vols). The conversion is controlled by the additional `ParConversion` data blocks in `sensitivity.xml` where the assumed par instruments and corresponding conventions are coded, as shown below for three types of discount curves.

```
 <DiscountCurves>

  <DiscountCurve ccy="EUR">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
      <!--DEP, FRA, IRS, OIS, FXF, XBS -->
      <Instruments>OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="OIS">EUR-OIS-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>
  </DiscountCurve>

  <DiscountCurve ccy="USD">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
```

```
      <Instruments>FXF,FXF,FXF,FXF,FXF,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="XBS">EUR-USD-XCCY-BASIS-CONVENTIONS</Convention>
        <Convention id="FXF">EUR-USD-FX-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>

  <DiscountCurve ccy="GBP">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
      <Instruments>DEP,DEP,DEP,DEP,DEP,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="DEP">GBP-DEPOSIT</Convention>
        <Convention id="IRS">GBP-6M-SWAP-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>
  </DiscountCurve>

</DiscountCurves>
```

Finally note that par sensitivity analysis requires that the shift tenor grid in the sensitivity data above matches the corresponding grid in the simulation (market) configuration. See also section 5.4.

4.6.2 Sensitivity Analysis and Pricing with Smile

This example (`python run_sensismile.py`) demonstrates the current state of sensitivity calculation in ORE 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 [2] 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;
- 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`".

4.6.3 Parametric VaR

This example (`python run_parametricvar.py`) 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.

4.6.4 Correlation

This example (`python run_correlation.py`) demonstrates a correlation calculation based on an external scenario matrix input. The result in `correlation.csv` shows a breakdown of the correlation between risk factor keys.

4.6.5 Stress Testing

This example (`python run_stress`) uses the same portfolio as the sensitivity analysis in 4.6.1. 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.

4.6.6 Stress Testing in the Par Domain

The stress test in 4.6.5 is performed in the "raw" domain of zero rate shifts, hazard rate shifts, optionlet volatility shifts etc. To analyse the impact of market rate shifts (Swap rates, CDS spreads, flat vols), one would have to manipulate the market data input into ORE and re-run the entire ORE process multiple times.

This example (`python run_parstress.py`) demonstrates the extended stress testing framework that operates in the "par" rate domain in a configurable way i.e without the user's manipulation of the input market data.

4.6.7 Stressed Sensitivity Analysis

This example (`python run_sensistress.py`) demonstrates the stressed sensitivity analysis. The new analytic type *SENSITIVITY_STRESS* utilizes the existing stresstest framework, applies stress scenarios to T0 market and runs sensitivity analysis. The Stresstest scenarios are given in the same input format as for the regular stresstest. Dependent sensitivity analysis is currently supported only in zero domain for all asset classes.

The stressed sensitivity analytic replaces the todaysMarket with a SimulationMarket during the sensitivity calculation. For some risk factors the simulationMarket behaves different as the todaysMarket. Depending on the simulation and stress scenario settings it could use different tenors when building curves or use only the ATM volatilities. It is recommended to activate UseSpreadedTermStructures and simulate SwaptionVolatilities for the stressed sensitivity run.

4.6.8 Historical Simulation VaR

This example (`python run_histsimvar.py`) demonstrates a historical simulation VaR calculation given a portfolio and externally provided "market scenarios" covering one or several historical observation period(s). The analytic is specified as usual in `ore.xml` with the following parameters:

- outputFile: csv file name of the resulting VaR report
- tradePnl: boolean, if true the VaR report will contain a breakdown by tradeID, risk class and risk type, otherwise the report shows the portfolio-level VaR only.
- riskFactorBreakdown: boolean, if true the VaR report will contain a breakdown by risk factor.
- quantiles: comma separated list of quantiles to be reported
- portfolioFilter (optional): Only trades with `portfolioId` equal to the provided filter name are processed, see `portfolio.xml`; the entire portfolio is processed, if omitted
- historicalPeriod: comma-separated date list, an even number of ordered dates is required (d1, d2, d3, d4, ...), where each pair (d1-d2, d3-d4, ...) defines the start and end of historical observation periods used
- mporDays: Number of calendar days between historical scenarios taken from the observation periods in order to compute P&L effects (typically 1 or 10)
- mporCalendar: Calendar applied in the scenario date calculation
- mporOverlappingPeriods: Boolean, if true we use overlapping periods of length mporDays (t to t + 10 calendar days, t+1 to t+11, t+2 to t+12, ...), otherwise consecutive periods (t to t+10, t+10 to t+20, ...)
- simulationConfigFile: defines the structure of the simulation market applied in the P&L calculation, e.g. discount and index curves, yield curve tenor points used, FX pairs etc.

- historicalScenarioFile: csv file containing the market scenarios for each date in the observation periods defined below; the granularity of the scenarios (e.g. discount and index curves, number of yield curve tenors) needs to match the simulation market definition above; each yield curve tenor scenario is represented as a discount factor

4.6.9 SMRC - Basic Market Risk Capital

Rn with: `run_smrc.py`

4.6.10 P&L and P&L Explain

This example (`run_pnlexplain.py`) demonstrates the P&L and P&L explain analytics on a very simple test portfolio that consists of two single-leg swaps. Main output is the P&L report in `Output/Pnl/pnl.csv` with the following columns

- TradeId
- Maturity and MaturityTime
- StartDate and EndDate of the P&L period, referred to as t0 and t1 below
- NPV(t0)
- NPV(asof=t0; mkt=t1)
- NPV(asof=t1; mkt=t0)
- NPV(t1)
- PeriodCashFlow: Aggregate of trade flows in the period, converted into the P&L currency below
- Theta: NPV(asof=t1; mkt=t0) - NPV(t0) + PeriodCashFLow
- HypotheticalCleanPnL: NPV(asof=t0; mkt=t1) - NPV(t0)
- CleanPnL: NPV(t1) - NPV(t0) + PeriodCashFlow
- DirtyPnL: NPV(t1) - NPV(t0)
- Currency

Moreover we write

- Four "flavours" of NPV reports used here
- Four related additional results reports
- Two reports for the market scenarios used in the two "lagged" NPV calculations

The second batch included in this example explains the P&L above in terms of portfolio sensitivities and changes in related market moves. The main output of this is in `Output/PnlExplain/pnl_explain.csv`. The PnlExplain analytic contains the Pnl analytic as dependent analytic, i.e. the PnlExplain analytic is self-sufficient kicking off Pnl calculation internally. The only additional piece of input for the explainer run is `sensitivity.xml`.

4.6.11 Stand-alone Par Conversion Utility

This example (`python run_parconversion.py`) demonstrates ORE's capability to convert external computed zero sensitivities (e.g Zero rates) to par sensitivities (e.g. to Swap rates) that is implemented by means of a Jacobi transformation of the "raw" sensitivities (e.g. to zero rates), see a sketch of the methodology in [2] and section 5.4 for configuration details.

To perform a par sensitivity analysis, the following required change in `ore.xml` is required

```
<Analytic type="zeroToParSensiConversion">
  <Parameter name="active">Y</Parameter>
  <Parameter name="marketConfigFile">simulation.xml</Parameter>
  <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
  <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
      <!-- Input file with the raw sensitivities -->
  <Parameter name="sensitivityInputFile">sensitivity.csv</Parameter>
  <Parameter name="idColumn">TradeId</Parameter>
  <Parameter name="riskFactorColumn">Factor_1</Parameter>
  <Parameter name="deltaColumn">Delta</Parameter>
      <Parameter name="currencyColumn">Currency</Parameter>
      <Parameter name="baseNpvColumn">Base NPV</Parameter>
       <Parameter name="shiftSizeColumn">ShiftSize_1</Parameter>
  <Parameter name="outputThreshold">0.000001</Parameter>
  <Parameter name="outputFile">parconversion_sensitivity.csv</Parameter>
  <Parameter name="outputJacobi">Y</Parameter>
  <Parameter name="jacobiOutputFile">jacobi.csv</Parameter>
  <Parameter name="jacobiInverseOutputFile">jacobi_inverse.csv</Parameter>
</Analytic>
```

The portfolio used in this example includes zero sensitivities of

- Discount and index curves
- Credit curves
- Inflation curves
- CapFloor volatilities

ORE reads the raw sensitivities from the csv input file *sensitivityInputFile*. The input file needs to have six columns, the column names can be user configured. Here is a description of each of the columns:

1. idColumn : Column with a unique identifier for the trade / nettingset / portfolio.
2. riskFactorColumn: Column with the identifier of the zero/raw sensitivity. The risk factor name needs to follow the ORE naming convention, e.g. DiscountCurve/EUR/5/1Y (the 6th bucket in EUR discount curve as specified in the sensitivity.xml)
3. deltaColumn: The raw sensitivity of the trade/nettingset / portfolio with respect to the risk factor
4. currencyColumn: The currency in which the raw sensitivity is expressed, need to be the same as the BaseCurrency in the simulation settings.
5. shiftSizeColumn: The shift size applied to compute the raw sensitivity, need to be consistent to the sensitivity configuration.

6. baseNpvColumn: The base npv of the trade / nettingset / portfolio in currency.

This is followed by the Jacobi transformation that turns "raw" sensitivities into sensitivities in the par domain (Deposit/FRA/Swap rates, FX Forwards, CC Basis Swap spreads, CDS spreads, ZC and YOY Inflation Swap rates, flat Cap/Floor vols). The conversion is controlled by the additional `ParConversion` data blocks in `sensitivity.xml` where the assumed par instruments and corresponding conventions are coded, as shown below for three types of discount curves.

```
<DiscountCurves>

  <DiscountCurve ccy="EUR">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
      <!--DEP, FRA, IRS, OIS, FXF, XBS -->
      <Instruments>OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="OIS">EUR-OIS-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>
  </DiscountCurve>

  <DiscountCurve ccy="USD">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
      <Instruments>FXF,FXF,FXF,FXF,FXF,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS,XBS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="XBS">EUR-USD-XCCY-BASIS-CONVENTIONS</Convention>
        <Convention id="FXF">EUR-USD-FX-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>

  <DiscountCurve ccy="GBP">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
    <ParConversion>
      <Instruments>DEP,DEP,DEP,DEP,DEP,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS,IRS</Instruments>
      <SingleCurve>true</SingleCurve>
      <Conventions>
        <Convention id="DEP">GBP-DEPOSIT</Convention>
        <Convention id="IRS">GBP-6M-SWAP-CONVENTIONS</Convention>
      </Conventions>
    </ParConversion>
  </DiscountCurve>

</DiscountCurves>
```

Finally note that par sensitivity analysis requires that the shift tenor grid in the sensitivity data above matches the corresponding grid in the simulation (market) configuration. See also section 5.4.

4.6.12 Base Scenario Utility

This example (`python run_basescenario.py`) demonstrates the `Scenario` analytic which has been added to export the simulation market's base scenario as a file. This analytic is e.g. used internally in P&L analytics 4.6.10, and it can be used to extract historical scenarios for the HistSim Var 4.6.8.

4.6.13 Zero Shift to Par Shift Conversion Utility

This example (`python run_zerotoparshift.py`) demonstrates the conversion of zero shifts to par rate shifts. ORE applies the zero rate shifts to the zero curves and computes the resulting shifts in the implied fair rate of a given set of par instruments. The zero rate shifts are defined as stresstests and the par instruments are defined in the usual sensitivity configuration. This analytic is used internally in the par-sensitivity-based P&L explainer.

4.7 Initial Margin

This section covers two groups of batches

- ISDA SIMM and IM Schedule
- a Dynamic Initial Margin case study

discussed in the following subsection.

4.7.1 ISDA SIMM and IM Schedule

This example (`python run_simm.py`) demonstrates the calculation of initial margin using ISDA's Standard Initial Margin Model (SIMM) based on a provided sensitivity file in ISDA's Common Risk Interchange Format (CRIF). In addition, we show how to use the standard "IM Schedule" method to compute initial margin.

ORE covers all SIMM versions since inception to date, i.e. 1.0, 1.1, 1.2, 1.3, 1.3.38, 2.0, 2.1, 2.2, 2.3, 2.4 (=2.3.8), 2.5, 2.5A, 2.6 (=2.5.6). All versions have been tested against the respective ISDA SIMM model unit test suites and pass these tests. Any new SIMM versions will be added with each ORE release.

For SIMM versions >= 2.2 we support SIMM calculation for both MPoR horizons, 1d and 10d.

Note that you need to purchase a SIMM model license from ISDA if you want to use the model in production, and the unit test suites mentioned above are provided to licensed vendors only. Therefore we unfortunately cannot share our ORE SIMM model test suite here either.

By running

```
python run_simm.py
```

ORE will pick up the small example CRIF file in `Input/crif.csv` (i.e. par sensitivities rebucketed and reformatted to match the ISDA CRIF template) and generate the resulting SIMM report in a `simm.csv` file. This report shows ISDA SIMM results with the usual breakdown by product class, risk class, margin type, bucket and SIMM "side" (IM to call or post). The SIMM calculation in this example is done for SIMM version 2.4 and 2.6, with MPoR 1d and 10d:

- SIMM 2.4, 1-day MPoR
- SIMM 2.4, 10-day MPoR
- SIMM 2.6, 1-day MPoR

- SIMM 2.6, 10-day MPoR

There are four SIMM-related input files – `ore_SIMM2.4_1D.xml`, `ore_SIMM2.4_10D.xml`, `ore_SIMM2.6_1D.xml`, `ore_SIMM2.6_10D.xml` – with corresponding folders in the `Output/` directory. The relevant inputs in the files are:

- SIMM version
- name of the CRIF file to be loaded
- calculation currency - this determines which Risk_FX entries of the CRIF will be ignored in the SIMM calculation
- result currency (optional) - currency of the resulting SIMM amounts in the report, by default equal to the calculation currency
- MPoR horizon, in terms of days

The market data input and todays's market configuration required here is minimal - limited to FX rates for conversions from base/calculation currency into USD and into the result currency.

IM Schedule

As an additional case in this example we demonstrate how to use the IM Schedule method to compute initial margin. The related input file is `Input/ore_schedule.xml`. It is also run when calling `python run.py`, and results are written to folder `Output/IM_SCHEDULE`. The basic input is provided in CRIF file format where ORE expects two lines per trade, one with RiskClass = PV and one with RiskClass = Notional, so that the amounts in these CRIF lines are interpreted as NPV respectively notional. Further required columns are product class and end date, as shown in the example `Input/crif_schedule.csv`. Note that the product class has to be in

- Rates
- FX
- Equity
- Credit
- Commodity

in contrast to SIMM where we use the combined RatesFX.

To run the IM Schedule analytic, the following minimal addition to `Input/ore_schedule.xml` is required.

```
<Analytics>
  <Analytic type="imschedule">
    <Parameter name="active">Y</Parameter>
    <Parameter name="crif">crif_schedule.csv</Parameter>
    <Parameter name="calculationCurrency">USD</Parameter>
  </Analytic>
</Analytics>
```

4.7.2 Dynamic Initial Margin and MVA

This example (`python run_dim.py`) demonstrates Dynamic Initial Margin calculations (see also [2]) for a number of basic products:

- 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),
- a EUR/USD cross currency Swap (case D),
- a EUR/USD FX Option (case E).

The essential results of each run are visualised in the form of

- evolution of expected DIM which feeds into the MVA calculation
- regression plots at selected future times

illustrated for cases A, B and E in figures 2 - 6.

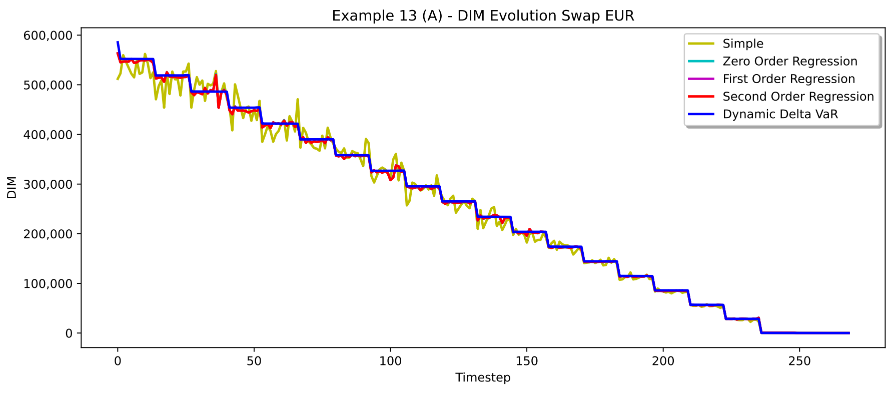

Figure 2: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swap of Example 13 A. Regression DIM is evaluated using regression of NPV change variances versus the simulated 3M Euribor fixing; regression polynomials are zero, first and second order (first and second order curves are not noticebly different in this case). The simulation uses 1000 samples and a time grid with bi-weekly steps in line with the Margin Period of Risk.

The DIM evolution graphs compare a subset of the following Initial Margin projection methods in ORE

- Simple: 99% quantile of NPV changes (Δ) over the Margin Period of Risk across all paths, i.e. same IM applied across paths
- Zero Order "Regression": Standard deviation of Δs scaled to the 99% quantile with factor 2.33 (normal distribution assumption); same IM applied across paths
- First/Second Order Regression: Conditional standard deviation of Δ computed by polynomial first/second order regression of Δ variances, scaled to the 99% quantile as avove; different IM amounts applied across paths, graphs show the expected DIM i.e. avergae across paths

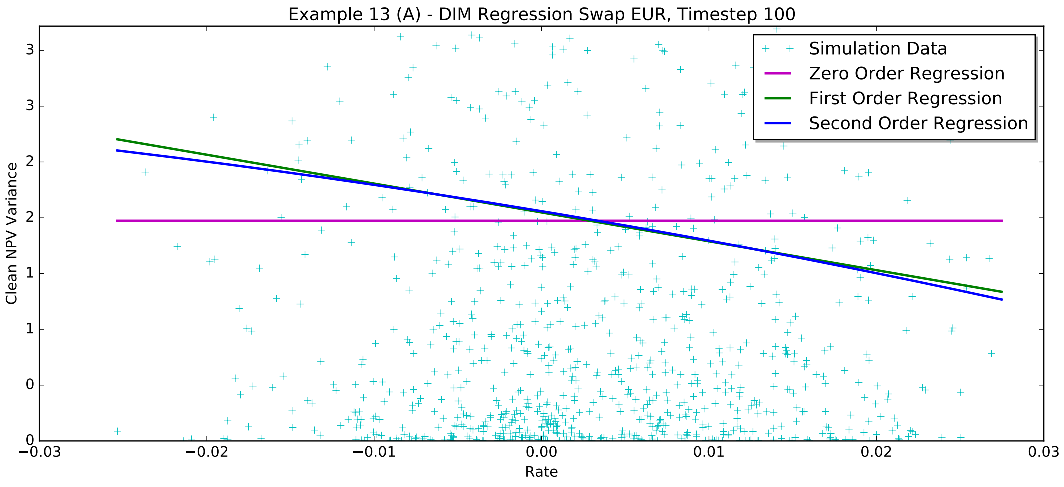

Figure 3: Regression snapshot at time step 100 for the EUR Swap of Example 13 A.

- Dynamic Delta VaR: 99% quantile VaR based on analytic deltas and vegas computed under scenarios; different IM amounts applied across paths, graphs show the expected DIM i.e. average across paths

For a discussion of the regression model performance see [18, 20]. The cases A–E are associated with various ORE master input files `Input/ore_A*.xml`, `Input/ore_B*.xml`, ..., which demonstrate the required simulation and xva analytic configurations.

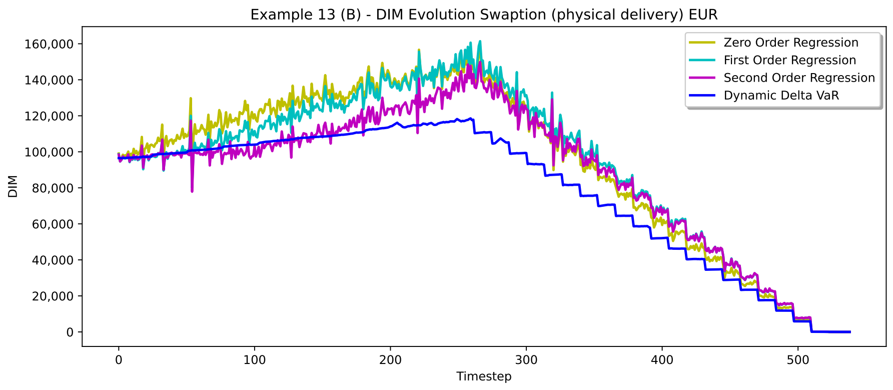

Figure 4: Evolution of expected Dynamic Initial Margin (DIM) for the EUR Swaption of Example 13 B with expiry in 10Y around time step 100.

4.7.3 Dynamic SIMM

The batch kicked off with (`python run_dim2.py`) demonstrates a prototype "Dynamic SIMM" implementation, see Input/Dim2/ore_amccg.xml with

```
<Analytic type="xva">
  ...
  <Parameter name="dimModel">DynamicIM</Parameter>
  ...
</Analytic>
```

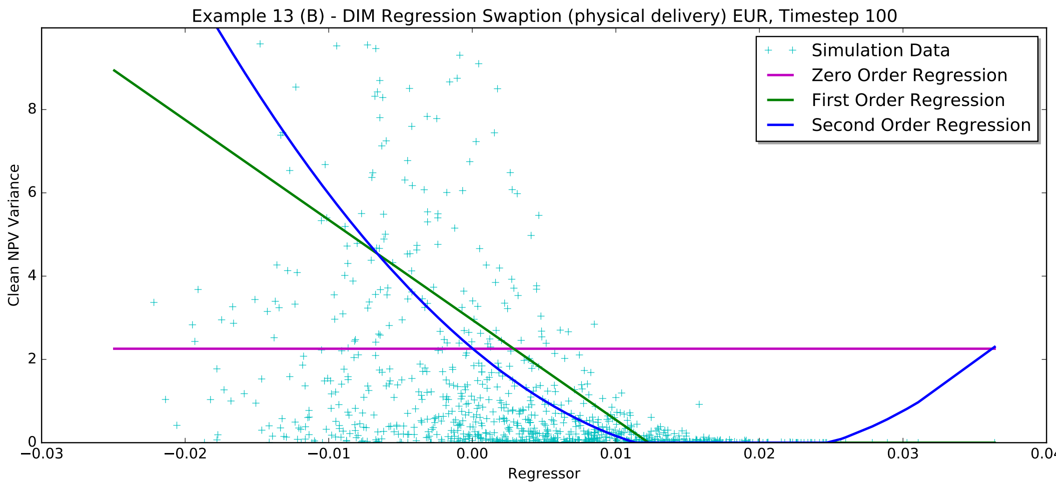

Figure 5: Regression snapshot at time step 100 (before expiry) for the EUR Swaption of Example 13 B.

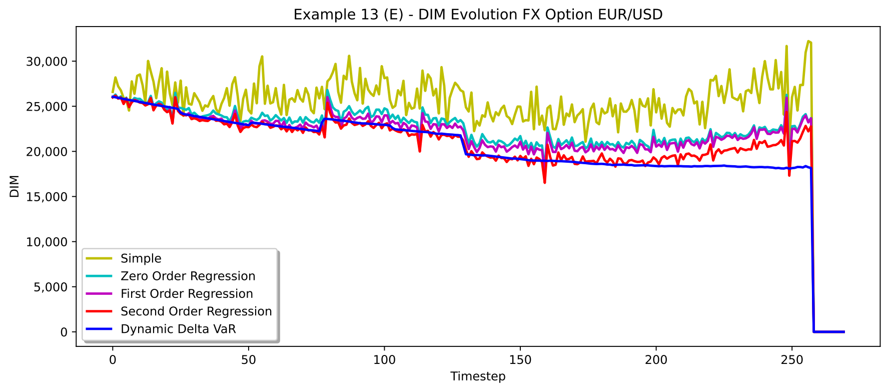

Figure 6: Evolution of expected Dynamic Initial Margin (DIM) for the EUR/USD FX Option of Example 13 (case E) with expiry in 10Y around time step 100.

and several related new parameters in the `simulation` section. The new method is embedded into AMC simulation and uses Algorithmic Differentiation to generate sensitivities along paths which feed into the dynamic SIMM calculation. Results are – among others – a `dim_evolution.csv` report and a new `dim_distribution.csv` report in Output/Dim2/AmcCg.

To validate the new method we apply the "conventional" SIMM calculator in ORE that we have enhanced with a CRIF generator limited to IR/FX risks for this purpose. The `run_dim2.py` script simulates a few paths, picks one of the paths and extracts the simulated market data (discount and index curves, FX rates, swaption volatilities) and writes them to `Input/DimValidation/marketdata.csv`, a market data file in ORE format, and likewise simulated fixings to `Input/DimValidation/fixings.csv`. The script then loops over simulation dates of that single path and performs a conventional SIMM run for each date (see `Input/DimValidation/ore_simm.xml`). This is used to check the new method's output on individual paths: When switching off model calibration and setting model vols close to zero in `Input/Dim2/simulation.xml` and `Input/Dim2/simulation_amccg.xml`, then we basically "roll down the forward curve", and both calculations should yield the same result. Figure 7 shows the comparison and expected outcome.

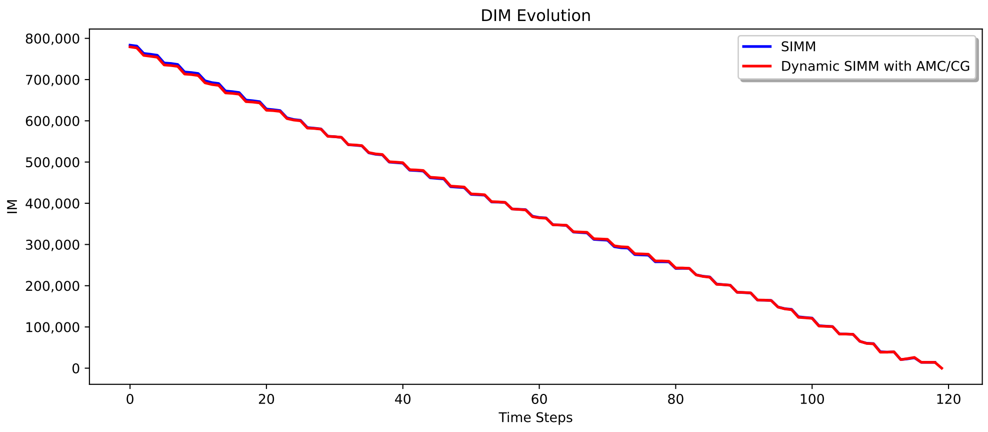

Figure 7: Evolution of SIMM vs Expected Dynamic Initial Margin (DIM) for a Euribor and SOFR-3M Swap, simulation with zero volatility, single path in case of the SIMM benchmark, 10k paths in case of DIM.

Next, we use a model with non-zero vols, i.e. re-activate calibration in `Input/Dim2/simulation.xml` and `Input/Dim2/simulation_amccg.xml`. Running another script (`python run_dim2_cube.py`) we now generate a few SIMM benchmark paths and compare to the expected DIM from 10k paths, see Figure 8

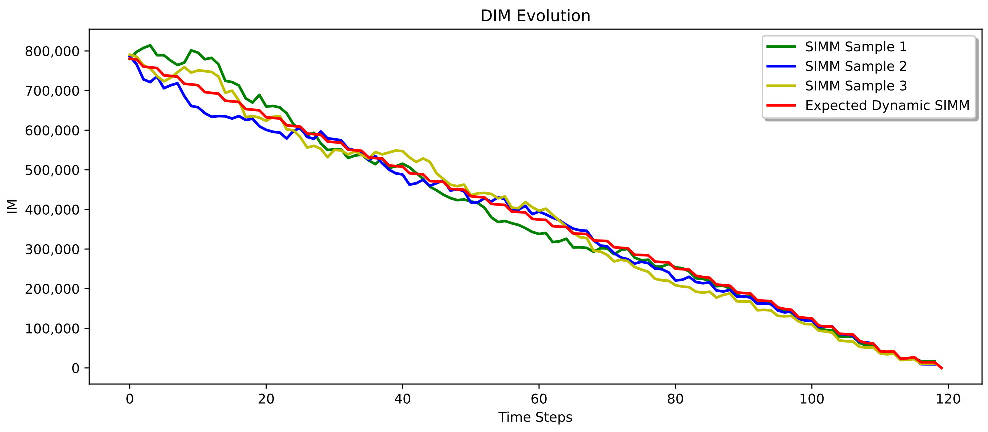

Figure 8: Evolution of SIMM vs Expected Dynamic Initial Margin (DIM) for a Euribor and SOFR-3M Swap; simulation with non-zero volatility, three paths in case of the SIMM benchmark, 10k paths in case of DIM.

Preliminary note on performance:

- The script takes about one minute to generate the benchmark SIMM evolution on a single path with 120 time steps for the example above
- Dynamic SIMM takes about 2 minutes for the expectation across 10k paths, on the same hardware (single core, Macbook Pro M2 Max), a speedup factor of 5000 compared to the pedestrian benchmark

We now zoom in on four future time points (1M, 1Y, 4Y, 8Y) and illustrate the SIMM distribution from the new report `dim_distribution.csv`. To benchmark the distributions we re-run `python run_dim2_cube.py` again, but with number of paths increased to 1000 and time steps reduced to the four dates we want to analyse, i.e. changing Input/Dim2/simulation.xml accordingly. This generates an updated output in Output/DimValidation/simm_cube.csv. Running the script `python plot_dim_distribution.py` then generates the comparison in Figure 9. Note that the Dynamic SIMM calculation takes a few seconds with 10k paths and a handful of time points, while the brute force benchmark calculation crunches 1k paths on four dates in about half an hour.

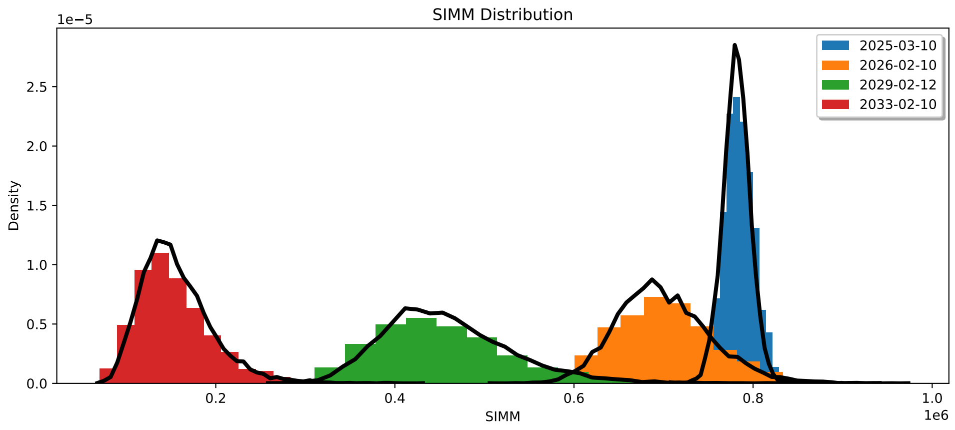

Figure 9: SIMM distribution from Dynamic SIMM (lines) vs benchmark (histograms) at time points 1M, 1Y, 4Y, 8Y from as of date.

Further investigation of this new Dynamic SIMM method is work in progress:

- path-wise benchmarking
- more products - FX Options, Swaptions, FX TaRF as a complex scripted example
- investigation of alternative regression methods in AMC and their impact on quality of AAD sensitivities
- comparison to the simple regression DIM model of section 4.7.2

4.8 Exposure

This section demonstrates exposure simulation and CVA for uncollateralised single trades across ORE's product range, mostly vanilla products.

The examples can be run individually (see below) or all together with `python run.py`.

- Swap with flat yield curve: `python run_swapflat.py`
- Swap with normal yield curves: `python run_swap.py`
- Swap with normal yield curves in HW2F: `python run_swap_hw2f.py`
- FRA: `python run_fra.py`

- European/American/Bermudan Swation and CallableSwap: `python run_swaption.py`
- Caps/Floors: `python run_capfloor.py`
- FX Forward and FX Option: `python run_fx.py`
- Resetting and Non-Resetting Cross Currency Swaps: `python run_ccs.py`
- Equity Forwards and Option: `python run_equity.py`
- Commodity Forward, Option, Swaption, APO: `python run_commodity.py`
- Inflation CPI and YOY Swap, the simulation is run twice using a Dodgson-Kainth and Jarrow-Yildirim model: `python run_inflation.py`
- Credit Default Swap: `python run_credit.py`

Somewhat more complex single-trade examples:

- Capped/Floored CMS Spread, Digital CMS Spread: `python run_cmsspread`
- Capped CMS Spread and Digital CMS Spread with formula-based payoff (slow): `python run_fbc.py`

Further exposure simulation features:

- Long-term simulation with and without horizon shift in the Linear Gauss Markov model: `python run_longterm.py`
- Simulation in different measures: `python run_measures.py`
- Simulation in the two-factor Hull-White model, calibrated to historical and risk neutral volatilities: `python run_hw2f.py`
- Wrong-Way-Risk: `python run_wwr.py`
- Flip View, switch perspectives easily for XVA: `python run_flipview.py`

Parameter calibration for exposure simulation model:

- HW n-factor historical calibration: `python run_hwhistoricalcalibration.py`

All cases are discussed in the following subsections.

4.8.1 Swap with flat yield curve

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

```
python run_swapflat.py
```

yields the exposure evolution in

```
Examples/swapflat/Output/*.pdf
```

and shown in figure 10. Both Swap simulation and Swaption pricing are run with calls

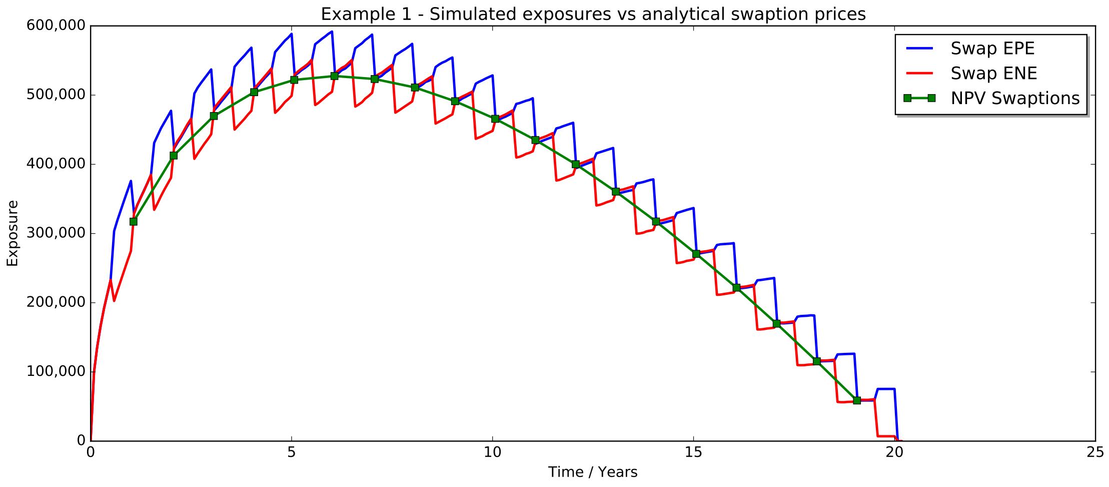

Figure 10: Vanilla ATM Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are European Swaption prices. The simulation was run with monthly time steps and 10,000 Monte Carlo samples to demonstrate the convergence of EPE and ENE profiles. A similar outcome can be obtained more quickly with 5,000 samples on a quarterly time grid which is the default setting of Example_ 1.

to the ORE executable, essentially

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

which are wrapped into the script `Examples/Exposure/run_swapflat.py` provided with the ORE release. It is instructive to look into the input folder in Examples/Exposure/Output/swapflat, the content of the main input file `ore.xml`, together with the explanations in section 5.

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 [2]. 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[1]. 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 figure 10. 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.

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

4.8.2 Swap with normal yield curves

Moving to the next case we see what changes when using a realistic (non-flat) market environment. Running the example with

```
python run_swap.py
```

yields the exposure evolution in

```
Examples/Exposure/Output/swap/*.pdf
```

shown in figure 11. In this case, where the curves (discount and forward) are upward

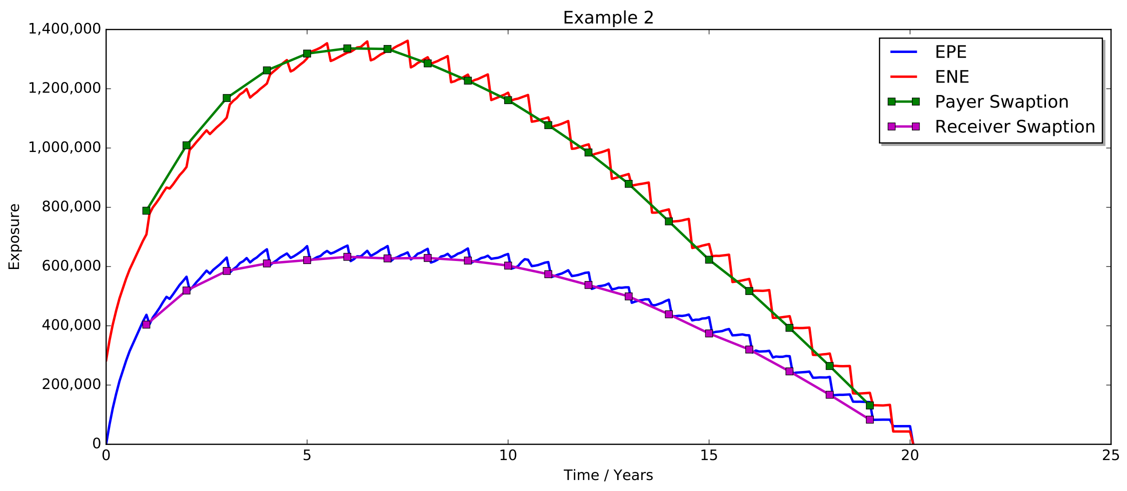

Figure 11: Vanilla ATM Swap expected exposure in a realistic market environment as of 05/02/2016 from both parties' perspectives. The Swap is the same as in figure 10 but receiving fixed 1%, roughly at the money. The symbols are the prices of European payer and receiver Swaptions. Simulation with 5000 paths and monthly time steps.

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

4.8.3 Forward Rate Agreement

The example in `run_fra.py` 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 in folder `Examples/Exposure/Output/fra`.

4.8.4 Swaptions

The batch in `python run_swaption.py` covers three cases

- European Swaptions with cash and physical settlement, compared to the underlying Forward Swap, see figure 12:
  The delivery type (cash vs physical) yields significantly different valuations as of today due to the steepness of the relevant yield curves (EUR). The cash settled Swaption's exposure graph is truncated at the exercise date, whereas the physically settled Swaption exposure turns into a Swap-like exposure after expiry. For comparison, the example also provides the exposure evolution of the underlying forward starting Swap which yields a somewhat higher exposure after the forward start date than the physically settled Swaption. This is due to scenarios with negative Swap NPV at expiry (hence not exercised) and positive NPVs thereafter. Note the reduced EPE in case of a Swaption with settlement of the option premium on exercise date.

- Bermudan and American Swaption, see figure 13:
  The underlying Swap is the same as in the European Swaption example above. 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.

- 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 14. The example is an extreme case where the underlying Swap is deeply in the money (receiving fixed 5%), and hence the call exercise probability is close to one. Modify the Swap and Swaption fixed rates closer to the money ($\approx$ 1%) to see the deviation between net exposure of the callable Swap and the exposure of a 'short' Swap with maturity on exercise. We have added more recently the combined CallableSwap instrument representation, check `portfolio.xml` and compare the CallableSwap NPV in `npv.csv` to the package NPV of Swap and Swaption.

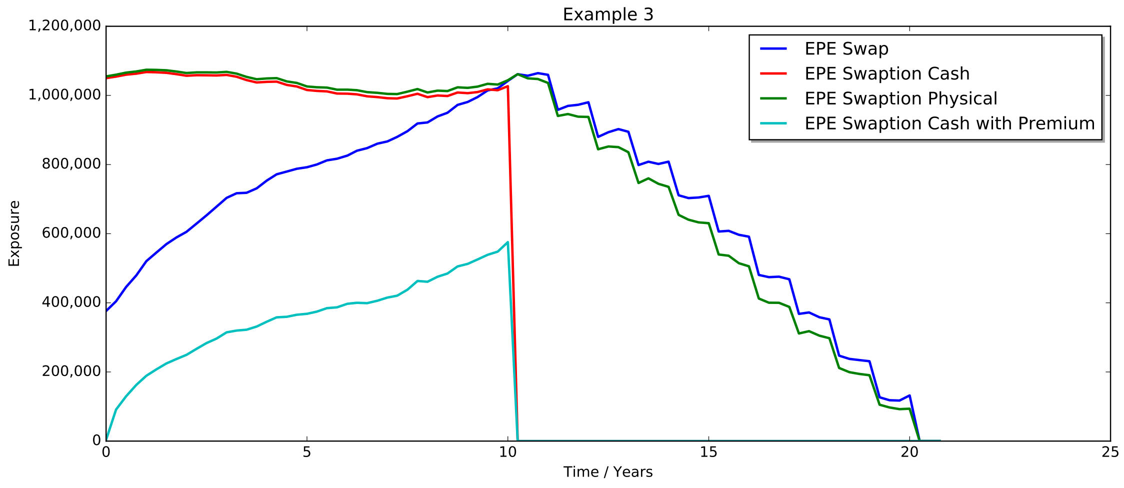

Figure 12: European Swaption exposure evolution, expiry in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

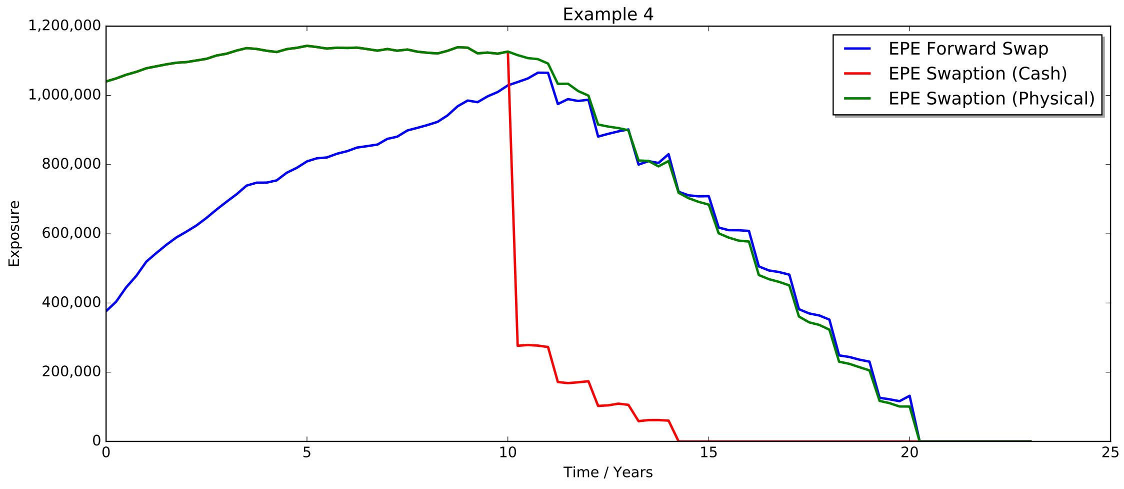

Figure 13: Bermudan Swaption exposure evolution, 5 annual exercise dates starting in 10 years, final maturity in 20 years, for cash and physical delivery. Simulation with 1000 paths and quarterly time steps.

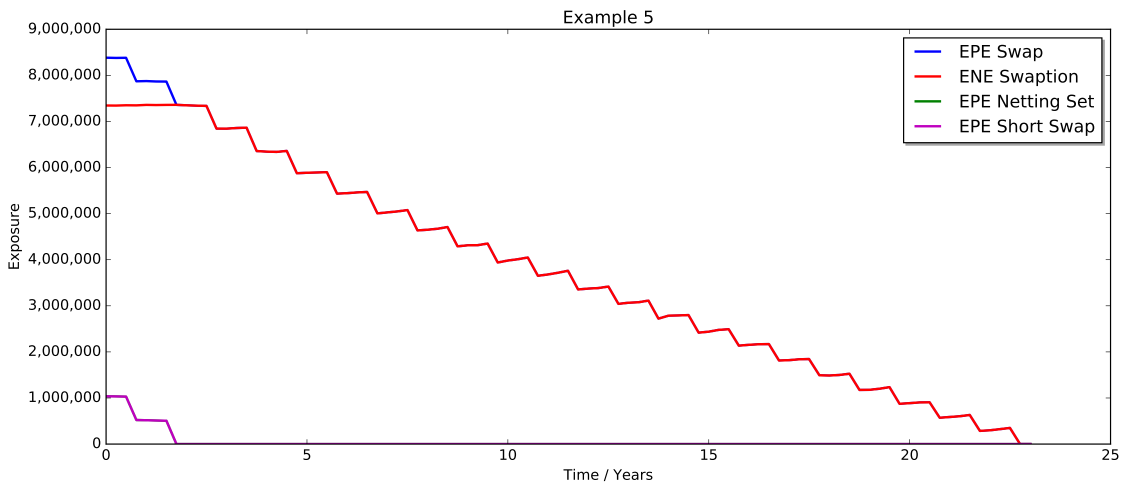

Figure 14: European callable Swap represented as a package consisting of non-callable Swap and Swaption. The Swaption has physical delivery and offsets all future Swap cash flows if exercised. The exposure evolution of the package is shown here as 'EPE Netting Set' (green line). This is covered by the 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.

4.8.5 Cap/Floor

The example `python run_capfloor.py` generates exposure evolutions of several Swaps, Caps and Floors. The example shown in figure 15 ('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.

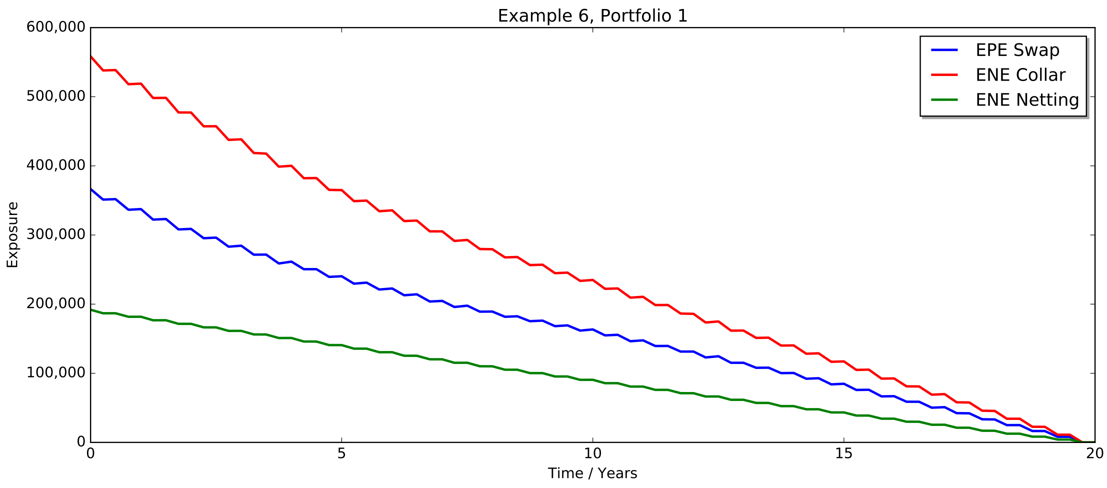

Figure 15: Swap+Collar, portfolio 1. The Collar has identical cap and floor rates at 4% so that it corresponds to a fixed leg which reduces the exposure of the Swap, which receives 3% fixed. Simulation with 1000 paths and quarterly time steps.

The second example in this folder shown in figure 16 ('portfolio 2') consists of a short Cap, long Floor and a long Collar that exactly offsets the netted Cap and Floor.

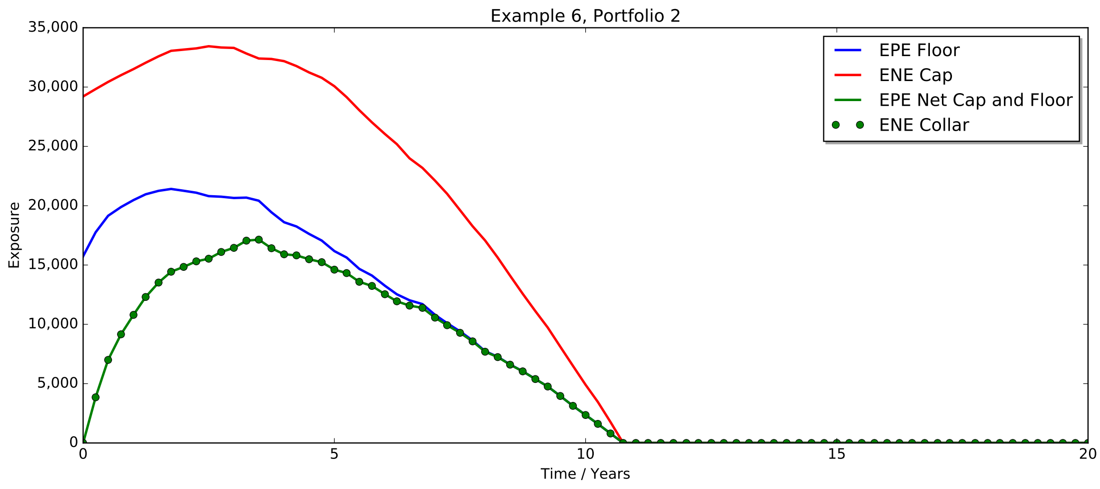

Figure 16: Short Cap and long Floor vs long Collar, portfolio 2. Simulation with 1000 paths and quarterly time steps.

Further three test portfolios are provided as part of this example. Run the example and inspect the respective output directories
`Examples/Exposure/Output/capfloor/portfolio_#`.

4.8.6 FX Forward and FX Option

Example `python run_fx.py` 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 17.

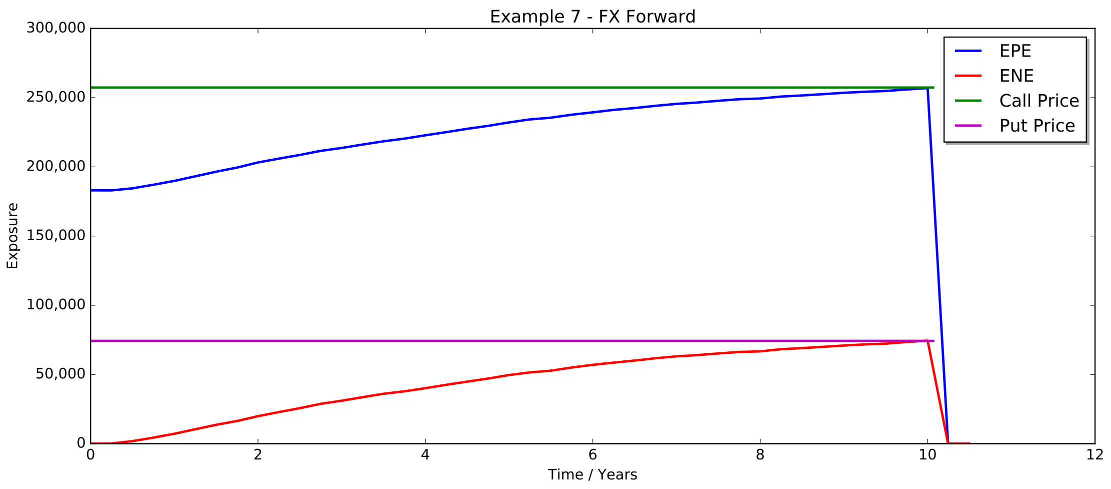

Figure 17: EUR/USD FX Forward expected exposure in a realistic market environment as of 26/02/2016 from both parties' perspectives. Value date is obviously in 10Y. The flat lines are FX Option prices which coincide with EPE and ENE, respectively, on the value date. Simulation with 5000 paths and quarterly time steps.

The same batch illustrates the exposure evolution for an FX Option, see figure 18. Recall that the FX Option value $NPV(t)$ as of time $0 \le t \le T$ satisfies

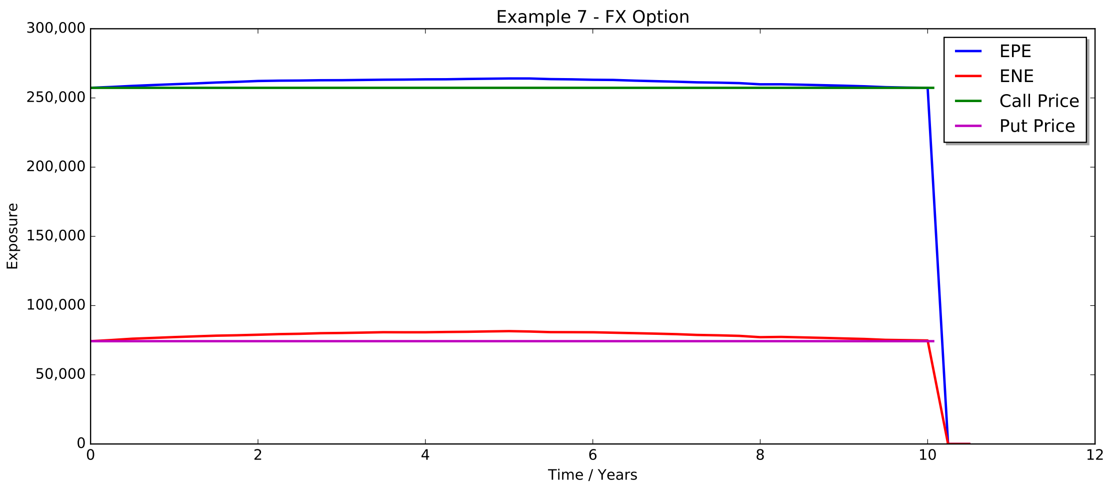

Figure 18: EUR/USD FX Call and Put Option exposure evolution, same underlying and market data as above, compared to the call and put option price as of today (flat line). Simulation with 5000 paths and quarterly time steps.

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

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

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 rolled forward, see section 5.3.3). The deviation can be removed by extending the volatility modelling, e.g. implying model consistent Black volatilities in each simulation step on each path.

4.8.7 Non-Resetting and Resetting Cross Currency Swaps

Batch `run_ccs.py` demonstrates a vanilla non-resetting cross currency Swap exposure. 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 19.

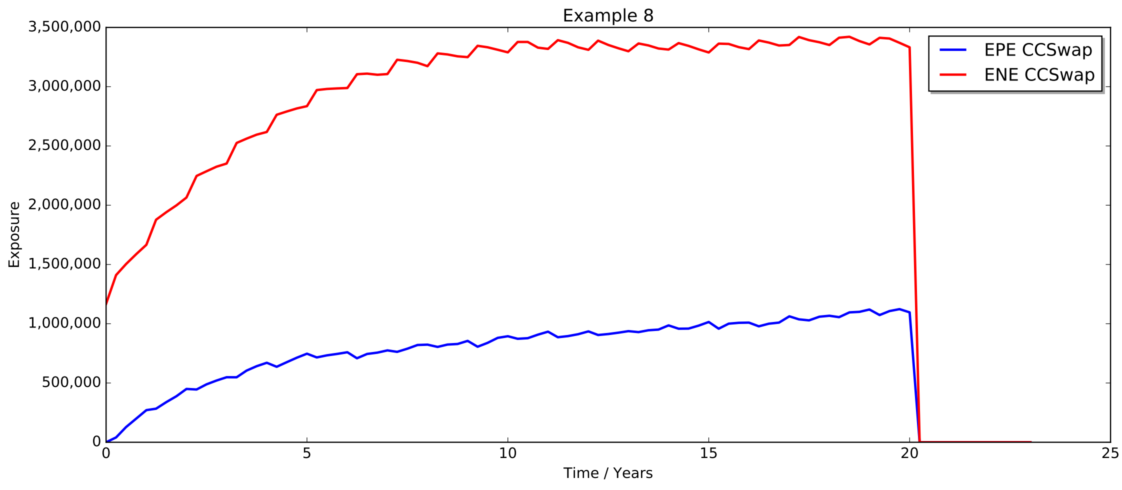

Figure 19: Cross Currency Swap exposure evolution without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

This run also demonstrates the "serialization" of the calibrated simulation model, see `Output/swaption/calibration.xml` and `Output/swaption/calibration.csv`, as well as the brief documentation of the `calibration` analytic in section 5.1.1.

Finally, the effect of the FX resetting feature, common in Cross Currency Swaps nowadays, is also demonstrated here using a separate instrument (see `portfolio_ccs.xml`). The example shows the exposure evolution of a EUR/USD cross currency basis Swap with FX reset at each interest period start, see figure 20. 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.

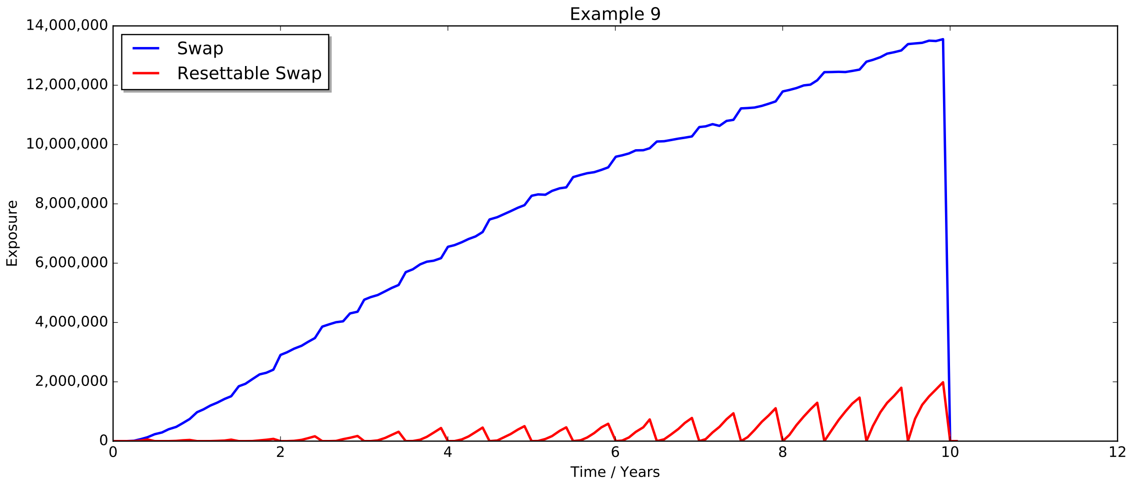

Figure 20: Cross Currency Basis Swap exposure evolution with and without mark-to-market notional reset. Simulation with 1000 paths and quarterly time steps.

4.8.8 Equity Derivatives

This example in `python run_equity.py` 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 21 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 section 4.8.6.

4.8.9 Commodity Derivatives

Calling

```
python run_commodity.py
```

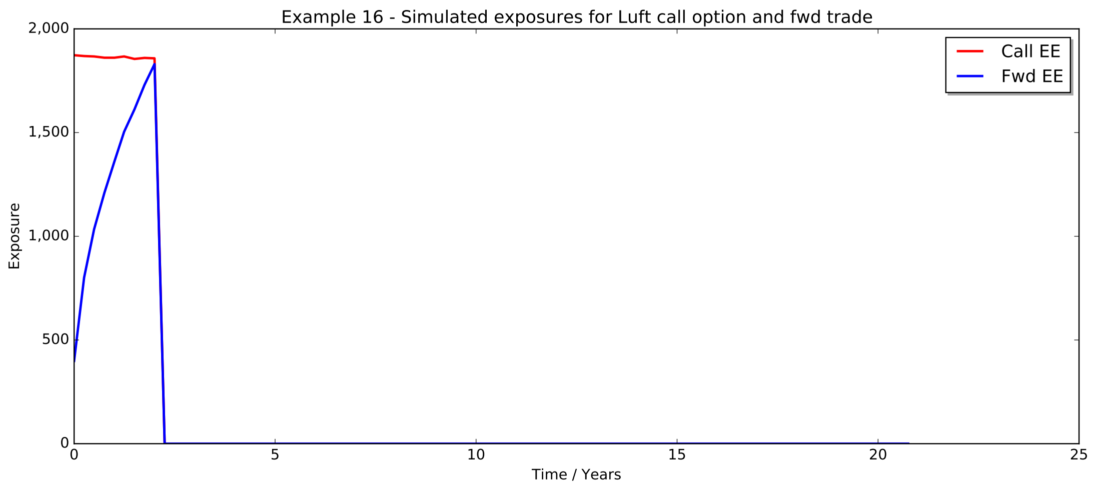

Figure 21: Equity ("Luft") call option and OTC forward exposure evolution, maturity in approximately 2.5 years. Simulation with 10000 paths and quarterly time steps.

demonstrates pricing and exposure simulation for a portfolio including a

- Commodity Forward
- Commodity Swap
- European Commodity Option
- Commodity Average Price Option
- Commodity Swaption

with the usual results, exposure reports and graphs.

4.8.10 Inflation CPI and YOY Swap - TODO

The example called with `python run_inflation.py` runs two exposure simulation batches for CPI and YOY Inflation Swaps

- using the Dodgson-Kainth (DK) inflation model
- using the Jarrow-Yildirim (JY) inflation model

We use the same portfolio in both batches that comprises four trades

- EU and UK CPI Swaps
- EU and UK YOY Swaps

Figures 22 and 23 show the exposure (EPE) graphs for all trades.

TODO: Discuss differences, check model calibrations

4.8.11 Credit Default Swap

Calling

```
python run_credit.py
```

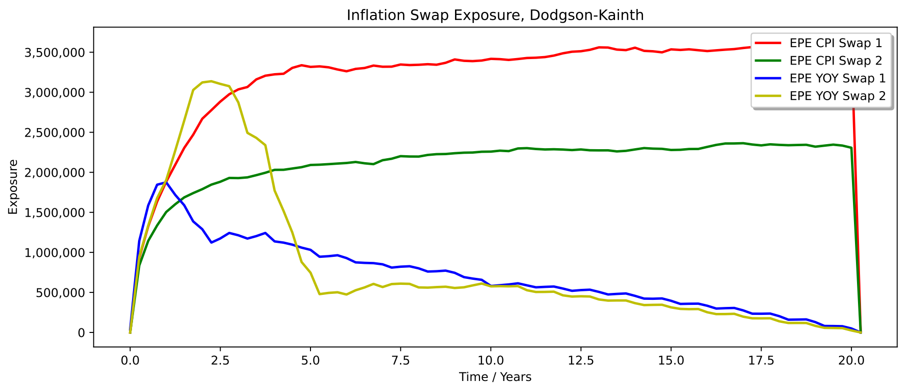

Figure 22: CPI and YOY exposure evolution using the Dodgson-Kainth model.

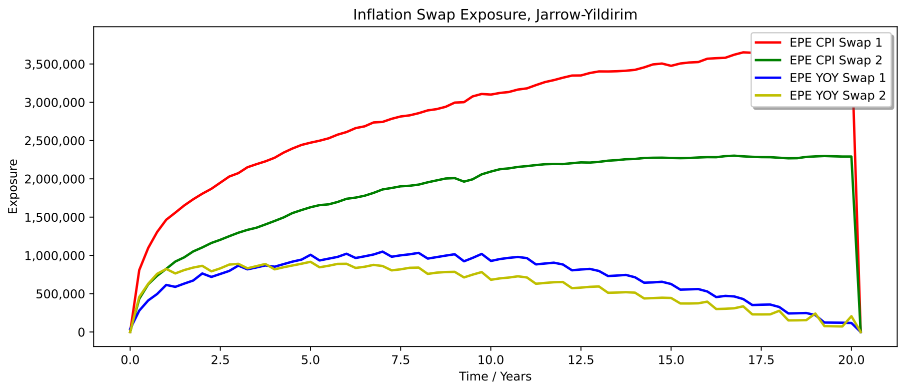

Figure 23: CPI and YOY exposure evolution using the Jarrow-Yildirim model.

runs the credit variant of the Swap exposure in 4.8.1. Running ORE here yields the exposure evolution shown in figure 24. Both CDS simulation and CDS Option pricing

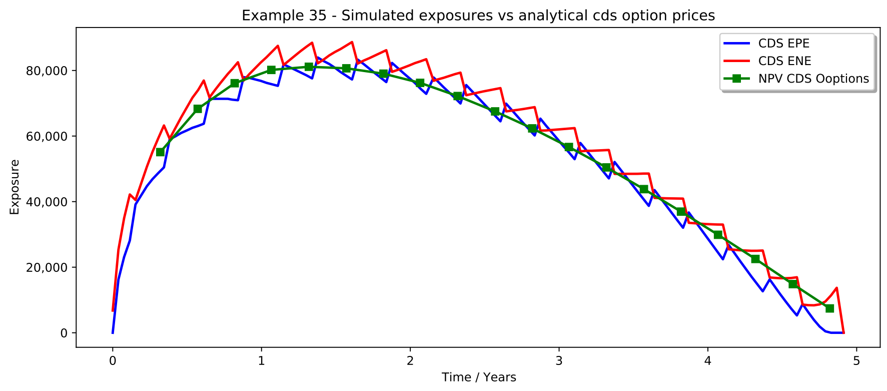

Figure 24: Credit Default Swap expected exposure in a flat market environment from both parties' perspectives. The symbols are CDS Option prices. The simulation was run with bi-weekly time steps and 10,000 Monte Carlo samples to demonstrate the convergence of EPE and ENE profiles. A similar outcome can be obtained more quickly with 5,000 samples on a monthly time grid which is the default setting here.

are run with calls to the ORE executable, essentially

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

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

which are wrapped into the script `Examples/Eposure/run_credit.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 6: CDS XVA results with LGM model

4.8.12 Capped/Floored (Digital) CMS Spread

Calling

```
python run_cmsspread.py
```

runs pricing and exosure simulation for

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

Results are found in folder `Examples/Exposure/Output/cmsspread/*` and exposure graphs in `Examples/Exposure/Output/mpl_cmsspread.pdf`.

4.8.13 Capped (Digital) CMS Spread with Formula-based Payoff

Calling

```
python run_fbc.py
```

runs pricing, exosure simulation and XVA for Swaps with CMS Spread and Digital CMS Spread legs using both formula-based payoff and "classic" ORE XML. The example portfolio also includes a Bond with formula-based payoff. Results are found in folder `Examples/Exposure/Output/fbc/*`

The formula-based leg can be seen as a predecessor of the more versatile scripted trade framework. However, the formula leg may be used to apply a multi-factor Log-normal Swap Rate model instead of the Gaussian interest rate models currently applied in the scripted trade framework.

4.8.14 Long-term simulation

Calling

```
python run_longterm.py
```

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 25 ('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 `Exposure/Input/simulation_longterm_2.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 [2]) *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 [17], 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 25 ('shifted horizon'). Figure 26 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

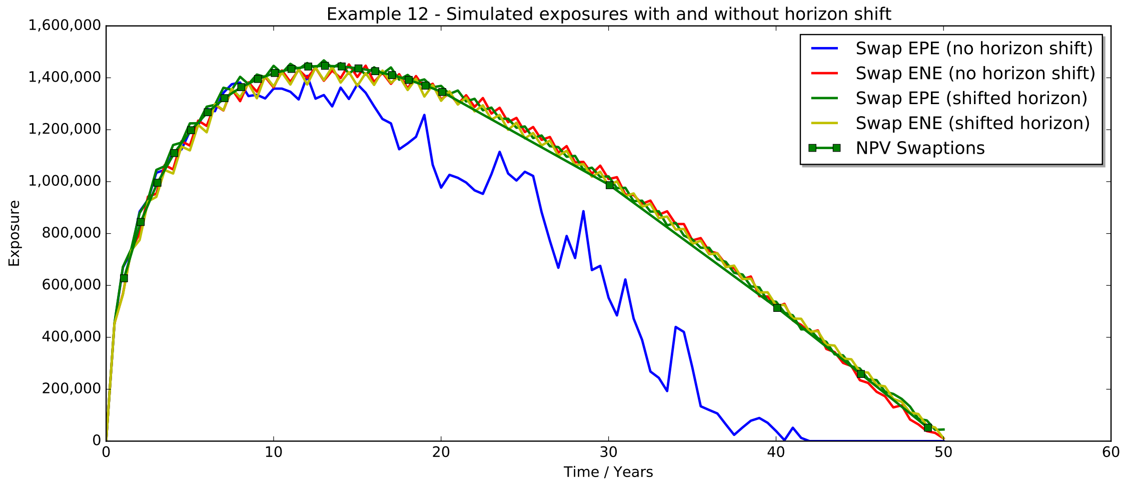

Figure 25: Long term Swap exposure simulation with and without horizon shift.

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.

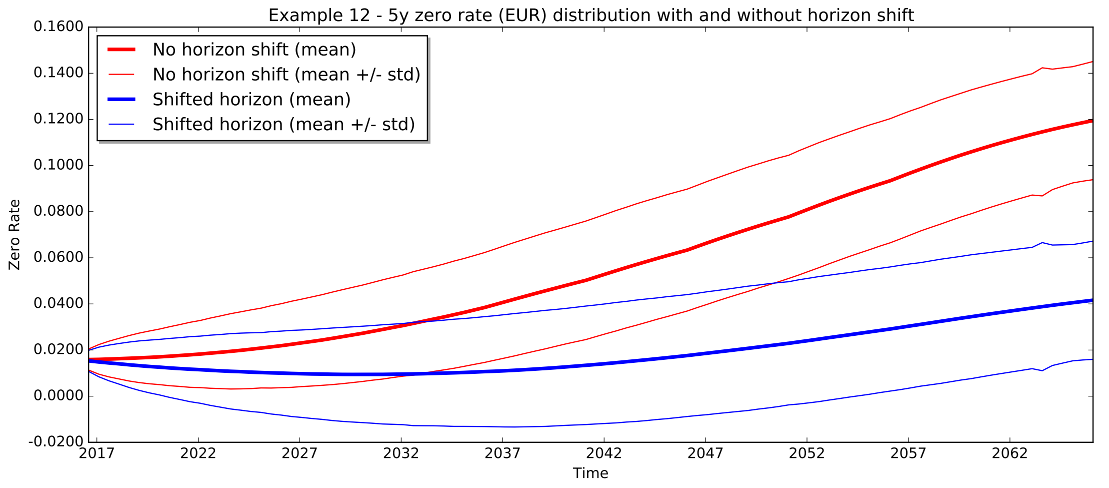

Figure 26: Evolution of rate distributions with and without horizon shift (change of measure). Thick lines indicate mean values, thin lines are contours of the rate distribution at ± one standard deviation.

4.8.15 Choice of Measure

Calling

```
python run_measures.py
```

illustrates the effect of measure changes on simulated expected and peak exposures. For that purpose we reuse the example in section 4.8.1 (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 27. 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.

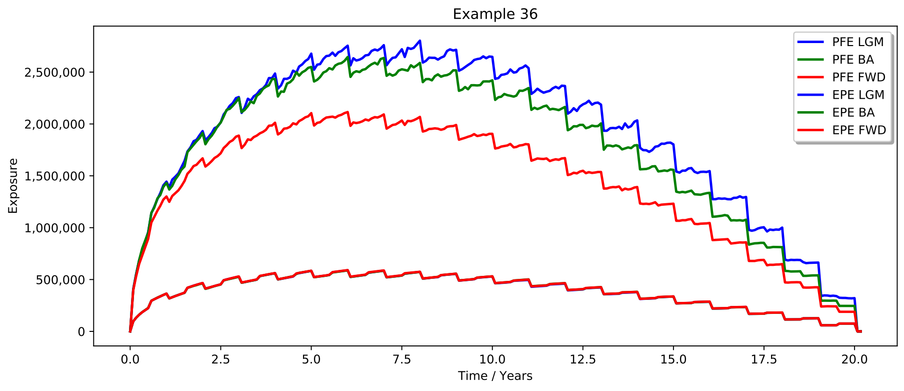

Figure 27: Evolution of expected exposures (EPE) and peak exposures (PFE at the 95% quantile) in three measures, LGM, Bank Account, T-Forward with T=20, with 10k Monte Carlo samples.

4.8.16 Simulation in the two-factor Hull-White model

Calling

```
python run_hw2f.py
```

kicks off two batches, a first model calibration batch and a second exposure simulation batch.

The first batch illustrates the model calibration and scenario generation under a Hull-White multifactor model with output in folder `Examples/Exposure/Output/hw2f_calibration`. The model is driven by two independent Brownian motions and has four states. The diffusion matrix sigma is therefore 2 x 4. The reversion matrix is a 4 x 4 diagonal matrix and entered as an array. Both diffusion and reversion are constant in time. Their values are not calibrated to the option market, but hardcoded in simulation.xml.

The values for the diffusion and reversion matrices were fitted to the first two principal components of a (hypothetical) analysis of absolute rate curve movements. These input principal components can be found in inputeigenvectors.csv in the input folder.

The tenor is given in years, and the two components are given as column vectors, see table 7.

tenor	eigenvector 1	eigenvector 2
1	0.353553390593	-0.537955502871
2	0.353553390593	-0.374924478795
3	0.353553390593	-0.252916811525
5	0.353553390593	-0.087587539893
10	0.353553390593	0.12267800393
15	0.353553390593	0.240659435416
20	0.353553390593	0.339148675322
30	0.353553390593	0.552478951238

Table 7: Input principal components

The first eigenvector represent perfectly parallel movements. The second eigenvector represent a rotation around the 7y point of the curve. Furthermore we prescribe an annual volatility of 0.0070 for the first components and 0.0030 for the second one. The values can be compared to normal (bp) volatilities.

We follow [16] chapter 12.1.5 “Multi-Factor Statistical Gaussian Model” to calibrate the diffusion and reversion matrices to the prescribed components and volatilities. We do not detail the procedure here and refer the interested reader to the given reference.

The example generates a single monte carlo path with 5000 daily steps and outputs the generated scenarios in scenariodump.csv. The python script pca.py performs a principal component analysis on this output. The model implied eigenvalues are given in table 8.

number	value
1	4.91449366493193346e-05
2	8.846877641067412e-06
3	5.82566039467854e-10
4	2.12989482255714 15e-10
5	9.254913949332787e-11
6	1.0861256211767673e-11
7	8.478795662698618e-14
8	9.74468069377584e-13

Table 8: Input principal components

Only the first two values are relevant, the following are all close to zero. The square root of the first two eigenvalues is given in table 9.

number	sqrt(value)
1	0.0070103449736 31422
2	0.0029743701250966414

Table 9: Input principal components

matching the prescribed input values of 0.0070 and 0.0030 quite well. The corresponding eigenvectors are given in etable 10.

tenor	eigenvector 1	eigenvector 2
1	0.34688826736335926	0.5441204725042812
2	0.3489303472083185	0.380259707350115
3	0.350362134519783	0.2581408080614405
5	0.3523983915961889	0.09230899007104967
10	0.3550169593982022	-0.11856777284904292
15	0.35647835947136625	-0.23676104168229614
20	0.3577146190751303	-0.3354486339442275
30	0.36042236352102563	-0.549124709243042

Table 10: Input principal components

again matching the input principal components quite well. The second eigenvector is the negative of the input vector here (the principal component analysis can not distinguish these of course).

The example also produces a plot comparing the input eigenvectors and the model implied eigenvectors as shown in figure 28.

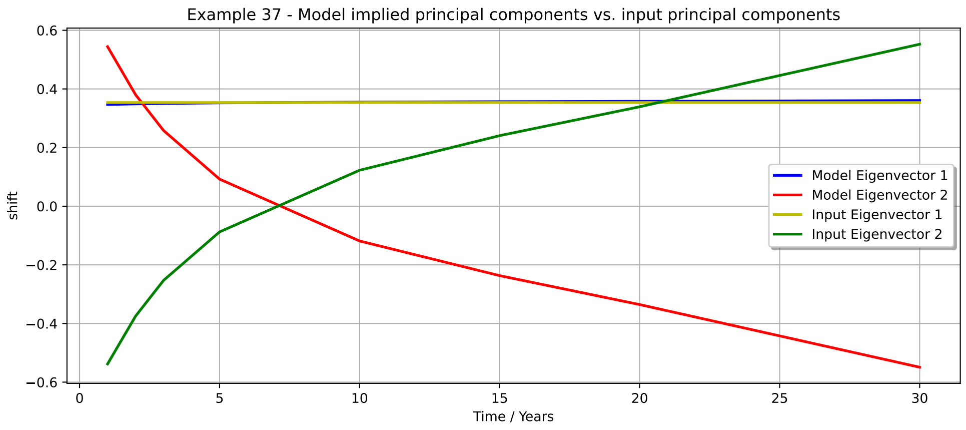

Figure 28: Input and model implied eigenvectors for a Hull-White 4-factor model calibrated to 2 principal components of rate curve movements (parallel + rotation). Notice that the model implied 2nd eigenvector is the negative of the input vector.

The second batch is similar to the Example in section 4.8.7 (EPE, ENE for a xccy swap), but uses a multifactor HW model for EUR and USD to generate scenarios. The parametrization of the HW models is taken from the previous run, resukts are found in folder `Examples/Exposure/Output/hw2f`.

Each of the two factors of each HW model is correlated with each of the two factors of the other currency's HW model and with the FX factors. Remember that the factors represent principal components of interest rate movements and so the correlations can be interpreted as correlations of these principal components with each other and the fx rate processes.

4.8.17 Wrong-Way-Risk

Calling

```
python run_wwr.py
```

runs an extension of the example in section 4.8.1 (single uncollateralised Swap) with dynamic credit and non-zero IR-CR correlation. Results are found in folder `Exposure/Output/wwr`. 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 11: IR Swap XVA results with LGM model

4.8.18 Flip View

Calling

```
python run_flipview.py
```

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 avoids manipulating the portfolio input or the netting set.

4.8.19 HW n-Factor Historical Calibration

Calling

```
python run_hwhistoricalcalibration.py
```

demonstrates how ORE provides functionality to calibrate the mean reversion speed(κ) and volatility(σ) parameters for the Hull-White n-factor model using historical market data. Two approaches are available:

1. Full Calibration Using Historical Data
   - ORE performs Principal Component Analysis (PCA) on historical interest rate curves and FX spot data.
   - Based on PCA results, ORE will use eigenvalues and eigenvectors to calibrate constant parameters for the HW n-factor model, including κ and σ.
2. Mean Reversion Calibration Only
   - Users can provide their own eigenvalues and eigenvectors for each curve.
   - ORE will then perform mean reversion calibration using these inputs without recalculating PCA.

4.9 Netting Set Exposure and Collateral

In this section we demonstrate exposure calculation and XVA at the level of a small netting set consisting of three Swaps in different currencies. We ilustrate the effect of several collateral choices on the resulting exposure.

4.9.1 MPoR (Biweekly) Grid

Move to folder `Examples/ExposureWithCollateral`. Calling

```
python run_biweekly.py
```

performs several calculations on a bi-weekly date grid with grid spacing that matches the Margin Period of Risk (MPoR). The results of the related ORE runs are found in sub-directories of `ExposureWithCollateral/Output/` (iah_0, iah_1, nocollateral, vm_threshold_*, vm_mta, vm_mpor)

The effect of the various collateral choices is illustrated in three plots

- no collateral - figure 29,
- collateral with threshold (THR) 1m EUR, minimum transfer amount (MTA) 100k EUR, margin period of risk (MPOR) 2 weeks - figure 30
- collateral with zero THR and MTA, and MPOR 2w - figure 31

The exposure graphs with collateral and positive margin period of risk show typical spikes. What is causing these? As sketched in [2], 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 [19] for a recent detailed discussion of collateral modelling. The approach currently implemented in ORE corresponds to *Classical+* in [19], 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 30 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.

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

We use one of the cases to demonstrate the XVA outputs, see folder `Examples/ExposureWithCollateral/Output/vm_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.

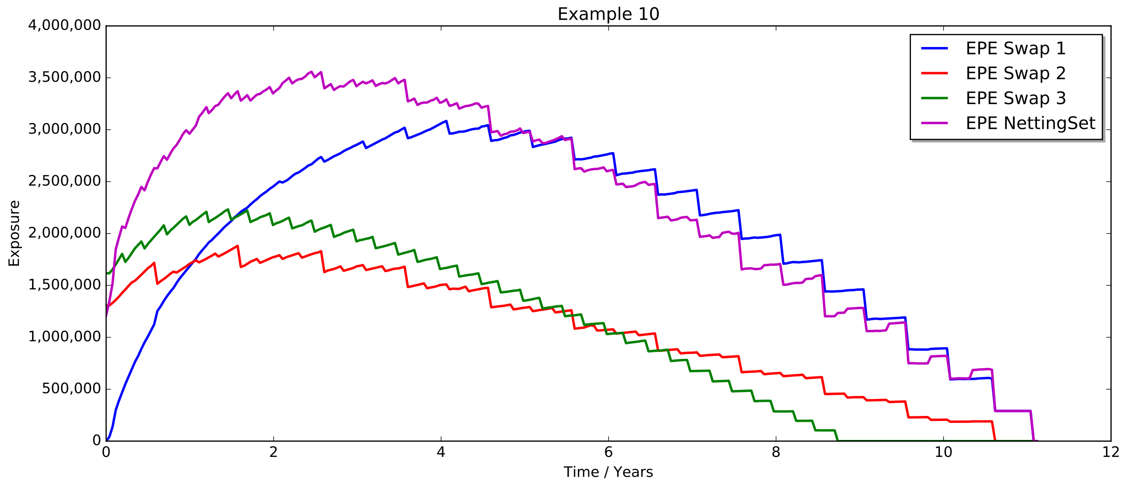

Figure 29: Three Swaps netting set, no collateral. Simulation with 5000 paths and bi-weekly time steps.

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

We also illustrate here the layout of an exposure report produced by ORE. The report shows the exposure evolution of Swap_1 without collateral which is found in folder `Examples/ExposureWithCollateral/Output/nocollateral/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, the Basel exposure measures EE_B and EEE_B, as well as allocated exposures are defined in [2]. The PFE quantile and allocation method are chosen as described in section 5.1.1.
In addition to single trade exposure files, ORE produces an exposure file per netting set. The example from the same folder as above is:

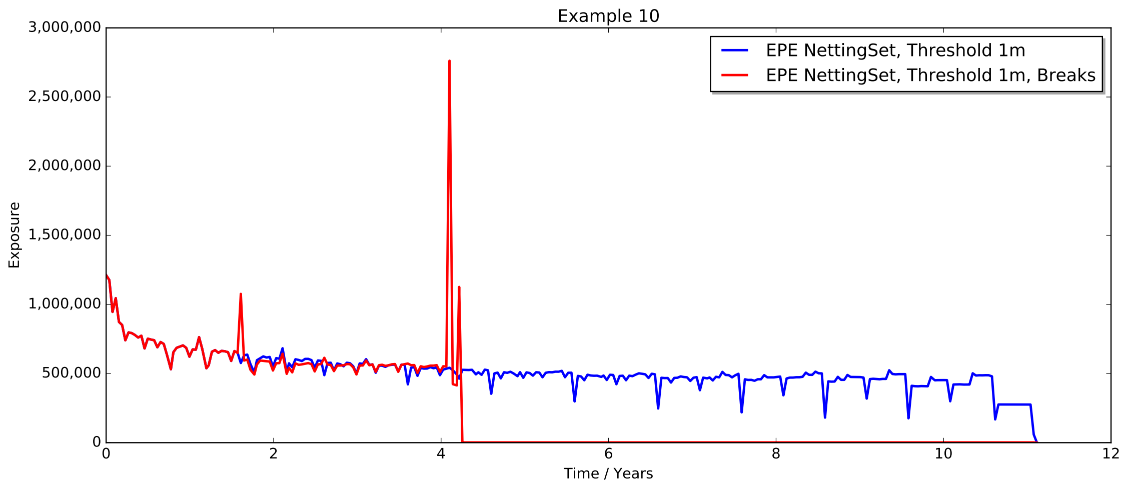

Figure 30: Three Swaps netting set, THR=1m EUR, MTA=100k EUR, MPOR=2w. The red evolution assumes that the each trade is terminated at the next break date. The blue evolution ignores break dates. Simulation with 5000 paths and bi-weekly time steps.

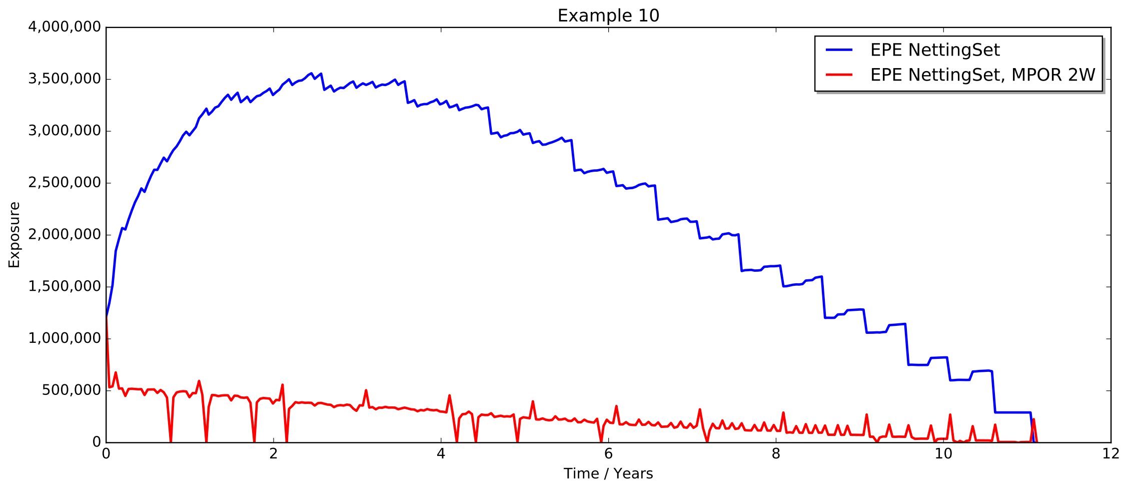

Figure 31: Three Swaps, THR=MTA=0, MPOR=2w. Simulation with 5000 paths and bi-weekly time steps.

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 here. We start again with the uncollateralised case in figure 32, followed by the case with threshold 1m EUR in figure 33. In both cases we apply the *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

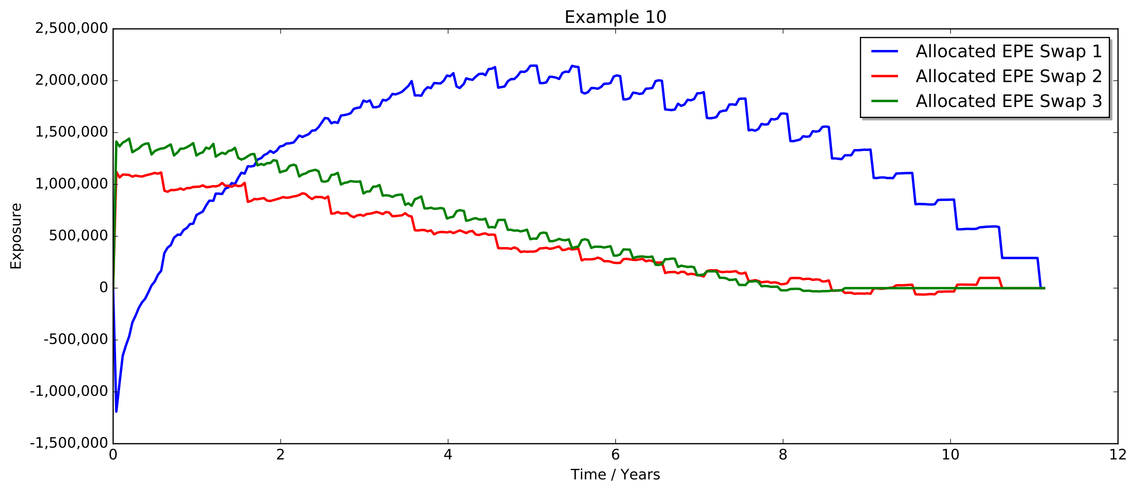

Figure 32: Exposure allocation without collateral. Simulation with 5000 paths and bi-weekly time steps.

moreover shows the typical spikes in the allocated exposures. The analytics results also

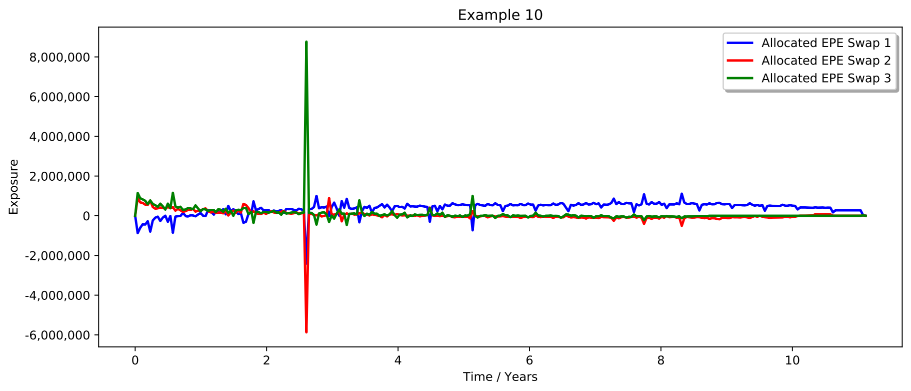

Figure 33: Exposure allocation with collateral and threshold 1m EUR. Simulation with 5000 paths and bi-weekly time steps.

feature allocated XVAs in file `xva.csv` which are derived from the allocated exposure profiles. Note that ORE also offers alternative allocation methods to the marginal method by Pykhtin/Rosen, which can be explored by modifying this Example.

4.9.2 Close-out Grid

So far we have used a "lagged" collateral approach, described at the end of the collateral section in [2], to take the Margin Period of Risk into account in exposure modelling. This used to have 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.

Calling

```
python run_closeout.py
```

we demonstrate an alternative approach supported by ORE since release 6, with results in folders `ExposureWithCollateral/Output/closeout_*`. 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: Simulation parameters

```
<Parameters>
  <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: XVA analytic

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

The resulting exposure graphs, with comparison of the uncollateralised to the collateralised case (with Variation Margin only) is shown in figure 34.

In figure 35 we have added Initial Margin based on ORE's regression model, and we compare the exposure with VM only to the exposure with both VM and IM. Figure 36 shows the related evolution of expected DIM, benchmarked against the Delta VaR DIM model, used in section 4.7.2 and described in [2].

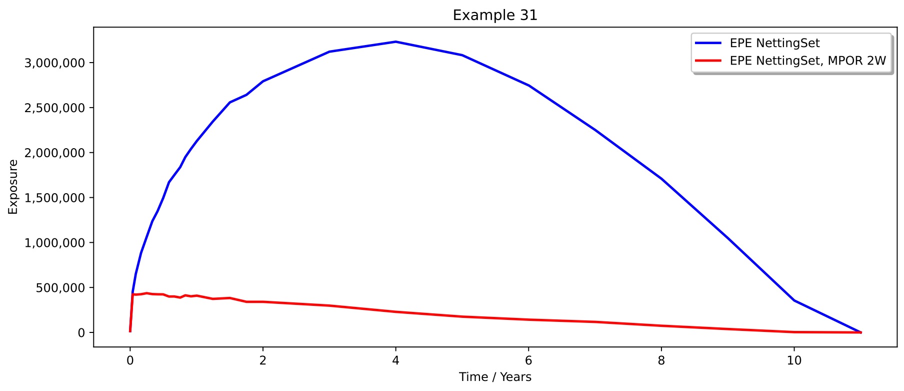

Figure 34: Uncollateralized Swap exposure vs exposure with Variation Margin, zero threshold and MTA. The simulation uses a variable grid with an auxiliary grid for closeout value calculation which is offset by the MPOR.

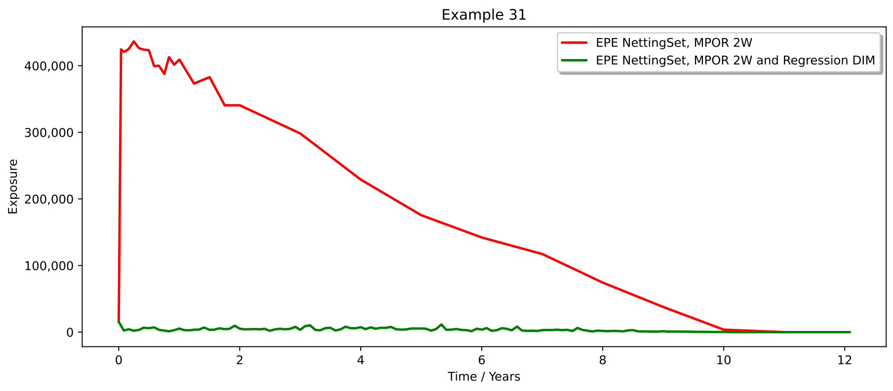

Figure 35: Comparison of Swap exposure with VM only as above vs VM+IM using first-order regression.

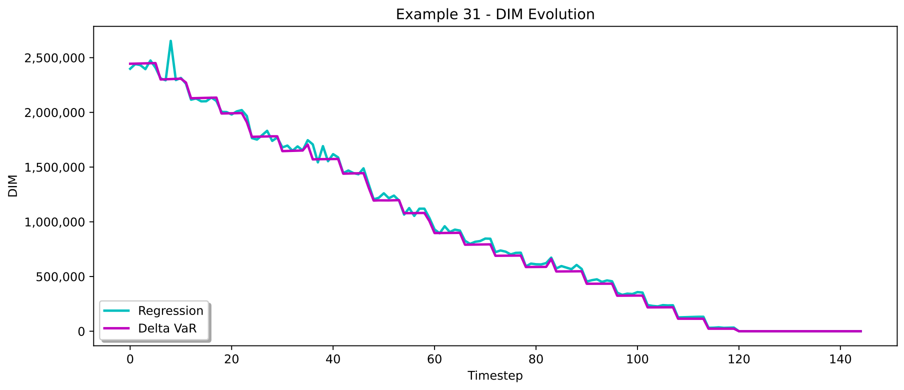

Figure 36: Evolution of expected IM from the regression model vs Delta VaR benchmark.

4.9.3 First MPoR Adjustment

Calling

```
python run_firstmpor.py
```

demonstrates a simple XVA calculation for a Receiver Swap (in netting set CPTY_A) and a Payer Swap (in netting set CPTY_B) with two initial VM balances each such that the netting set is fully collateralised resp. over/under-collateralised.

The new flag firstMporCollateralAdjustment in ore.xml's XVA section affects the first MPoR, 2 weeks in the example. If set to true, the difference between initial collateral balance and mtm is carried over during the first MPoRr period, if the difference increases our exposure - a conservative measure.

4.10 Scripted Trade

The scripted trade was added to ORE to gain more flexibility in representing exotic products, with hybrid payoffs across asset classes, path-dependence, multiple kinds of early termination options. The scripted trade module uses Monte Carlo and Finite Difference pricing approaches, it is an evolving interface to implement parallel processing with GPUs and a central interface to implement AD methods in ORE. See the separate documentation in folder Docs/ScriptedTrade for an introduction to trade representation, scripting language, model and pricing engine configuration.

The example in this folder `Examples/ScriptedTrade` is a basic demonstration of ORE's scripted trade functionality. In this example we provide a self-contained case that can be run as usual calling

```
python run.py
```

This generates an NPV and cash flow report for the following portfolio

- Trade 1: Vanilla European Equity Option, represented as standard ORE XML with analytical pricing
- Trade 2: Same Option as above, represented as "generic" scripted trade with scripted payoff embedded into the trade XML, pricing via Monte Carlo
- Trade 3: Same Option as above, same representation, pricing via Finite Differences triggered by a `ProductTag` assigned to the script and used in `pricingengine.xml`
- Trade 4: Same Option as above, the scripted trade now refers to an "external" script in `scriptlibrary.xml`, MC pricing
- Trade 4b: Same as trade 4, but "compact" scripted trade representation (uncomment trade 4b in `portfolio.xml`)
- Trade 5: Barrier Option with single continuously observed Up & Out barrier, represented as standard ORE XML with analytical pricing
- Trade 6: Same Barrier Option as above, approximated as generic scripted trade with daily barrier observation
- Trade 6b: Same Barrier Option as above, approximated as "compact" scripted trade with daily barrier observation (uncomment trade 6b in `portfolio.xml`)
- Trade 7: Same Barrier Option as above, represented as generic scripted trade with continuously observed barrier, i.e. adjusting for the probability of knock-out between daily observations

- Trade 7b: Same Barrier Option as of above, represented as "compact" scripted trade (uncomment trade 7b in `portfolio.xml`)
- Trade 8: Equity Accumulator, represented as generic scripted trade with external payoff script
- Trade 8b: Same Equity Accumulator as above, represented as compact scripted trade with external payoff script (uncomment trade 8b in `portfolio.xml`)

Note:

- In all cases we use the Black-Scholes model to drive the Equity process.
- The Barrier Option pricing using the scripted trade deviates noticeably from the analytical pricing when we use daily observations (trade 6 and 6b), but matches quite closely when we adjust for the probability of knock-out between observation dates (trade 7 and 7b)
- We are not aware of analytical pricing for the Accumulator product in trade 8 to benchmark against; trade 8 is priced with MC, FD pricing of the Accumulator is possible as well but requires a separate payoff script, only in the vanilla European option case we can utilize the same script for both MC and FD pricing

Though this initial example shows only single-asset Equity cases, the scripted trade in its current version is significantly more versatile, more examples and scripts to follow.

4.11 American Monte Carlo

The cases in this section, folder `AmericanMonteCarlo`, demonstrate how to use American Monte Carlo Simulation (AMC) to generate exposures in ORE.

- We start with benchmarking against "classic" exposure simulation, i.e. we run both AMC and classic simulation on a small IR/FX portfolio, almost vanilla, that consists of a Bermudan Swaption, Single and Cross Currency Swaps, FX Swap and FX Option, and we compare the resulting AMC vs. classic exposures. Run with `python run_benchmark.py`
- Scripted Bermudan Swaption and LPI Swap: This case shows that the scripted trade framework works with AMC too, demonstrated here with a scripted Bermudan Swaption and an LPI Swap.
  Run with: `python run_scriptedberm.py`
- FX TaRF: FxTaRF product is implemented using the scripted trade framework "under the hood" with the payoff script embedded into C++, so that it is neither explicit in the trade XML nor in the script library.
  Run with: `python run_fxtarf.py`
- Forward Bond with AMC, run with: `python run_forwardbond.py`
- Overlapping Close-Out Grid, run with: `python run_overlapping.py`
- Scenario Statistics, run with: `python run_scenariostatistics.py`

4.11.1 Benchmarking AMC vs Classic Simulation

This example demonstrates how to use American Monte Carlo simulation (AMC) to generate exposures in ORE. For a sketch of the methodology and comments on its implementation in ORE see [2]. Moreover we discuss the essential configuration changes for applying AMC.

Calling

```
python run_benchmark.py
```

performs two ORE runs, a 'classical' exposure simulation and an American Monte Carlo simulation, both on a quarterly simulation grid and for the same portfolio consisting of four trades:

- Bermudan swaption
- Single Currency Swap
- Cross Currency Swap
- FX Option

We use a 'flat' market here (yield curve and Swaption volatility surface). The number of simulation paths is 2k in the classic simulations. If not stated otherwise below, the number of training paths and simulation paths is 10k in the AMC simulations.

In the following we compare the AMC exposure profiles to those produced by the 'classic' valuation engine for each trade and the netting set.

Figure 37 shows the EPE and ENE for a Bermudan Swaption 10y into 10y in (base ccy) EUR with physical settlement. The classic run uses the LGM grid engine for valuation. We observe close agreement between the two runs. To achieve the observed agreement, it is essential to set the LGM model's mean reversion speed to zero in both

- the Bermudan Swaption LGM pricing model (see Input/pricingengine.xml), and
- the Cross Asset Model's IR model components (see Input/simulation.xml and Input/simulation_amc.xml)

and to use a high order 6 of the regression polynomials (see Input/pricingengine_amc.xml).

Figure 38 shows the EPE and ENE for a 20y vanilla Swap in USD. The currency of the amc calculator is USD in this case, i.e. it is different from the base ccy of the simulation (EUR). The consistency of the classic and amc runs in particular demonstrates the correct application of the currency conversion factor (see methodology guide). To get a better accuracy for purposes of the plot in this document we increased the number of training paths for this example to 50k and the order of the basis functions to 6.

Figure 39 shows the EPE and ENE for a 20y cross currency Swap EUR-USD.

Figure 40 shows the EPE and ENE for a vanilla FX Option EUR-USD with 10y1m expiry. For the classic run the FX volatility surface is not implied by the cross asset model but kept flat, which yields a slight hump in the profile. The AMC profile is flat

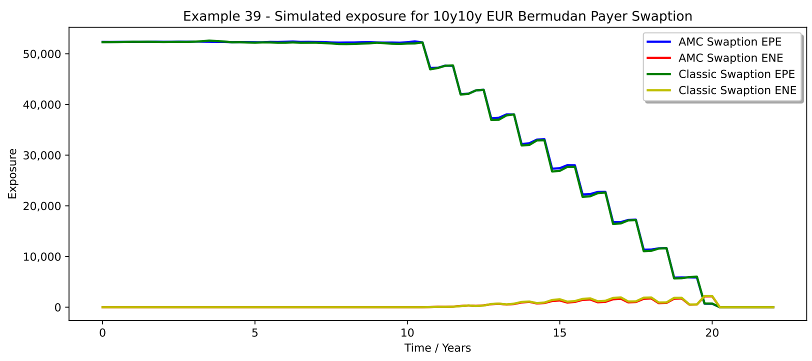

Figure 37: EPE of a EUR Bermudan Swaption computed with the classic and AMC valuation engines, using 50k training paths for the AMC simulation.

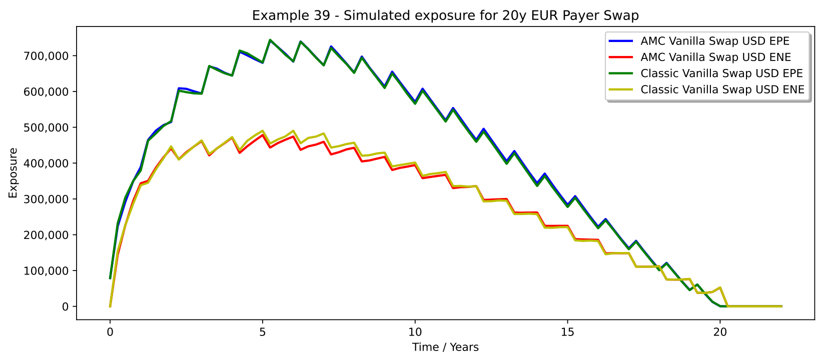

Figure 38: EPE of a USD swap computed with the classic and AMC valuation engines

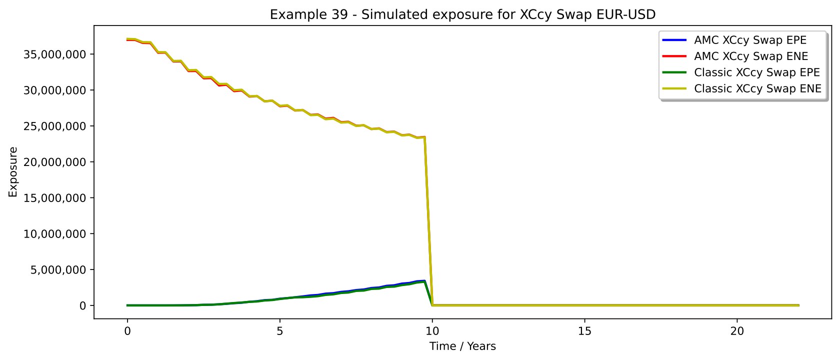

Figure 39: EPE of a EUR-USD cross currency swap computed with the classic and AMC valuation engines

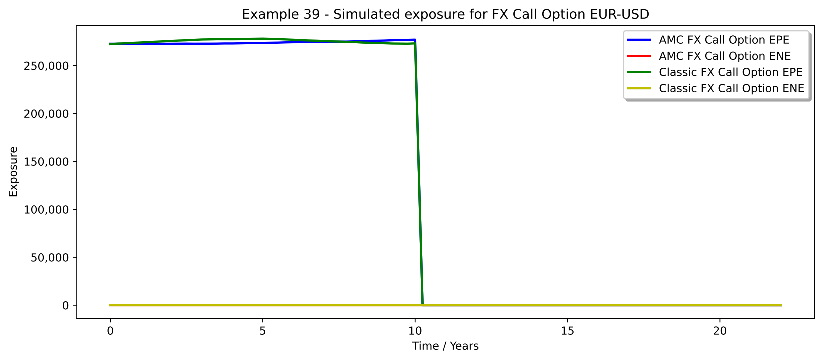

Figure 40: EPE of a EUR-USD FX option computed with the classic and AMC valuation engines

on the other hand which demonstrates the consistency of the FX Option pricing with the risk factor evolution model.

Analytic Configuration

To use the AMC engine for an XVA simulation the following needs to be added to the `simulation` analytic in `ore.xml`:

```
<Analytic type="simulation">
  ...
  <Parameter name="amc">Y</Parameter>
  <Parameter name="amcPricingEnginesFile">pricingengine_amc.xml</Parameter>
  <Parameter name="amcTradeTypes">Swaption</Parameter>
  ...
</Analytic>
```

The trades which have a trade type matching one of the types in the `amcTradeTypes` list, will be built against the pricing engine config provided and processed in the AMC engine. As a naming convention, pricing engines with engine type AMC provide the required functionality to be processed by the AMC engine, for technical details cf. [2].

All other trades are processed by the classic simulation engine in ORE. The resulting cubes from the classic and AMC simulation are joined and passed to the post processor in the usual way.

Note that since sometimes the AMC pricing engines have a different base ccy than the risk factor evolution model (see below), a horizon shift parameter in the simulation set up should be set for all currencies, so that the shift also applies to these reduced models.

Pricing Engine Configuration

At this point we assume that the reader is generally familiar with the configuration section 5, in particular pricing engine configuration in section in [1].

The pricing engine configuration is similar for all AMC enabled products, e.g. for Bermudan Swaptions:

```
<Product type="BermudanSwaption">
```

```
  <Model>LGM</Model>
  <ModelParameters/>
  <Engine>AMC</Engine>
  <EngineParameters>
    <Parameter name="Training.Sequence">MersenneTwisterAntithetic</Parameter>
    <Parameter name="Training.Seed">42</Parameter>
    <Parameter name="Training.Samples">50000</Parameter>
    <Parameter name="Training.BasisFunction">Monomial</Parameter>
    <Parameter name="Training.BasisFunctionOrder">6</Parameter>
    <Parameter name="Pricing.Sequence">SobolBrownianBridge</Parameter>
    <Parameter name="Pricing.Seed">17</Parameter>
    <Parameter name="Pricing.Samples">0</Parameter>
    <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
    <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
    <Parameter name="MinObsDate">true</Parameter>
    <Parameter name="RegressionOnExerciseOnly">false</Parameter>
  </EngineParameters>
</Product>
```

The `Model` differs by product type, table 12 summarises the supported product types and model and engine types. The engine parameters are the same for all products:

1. `Training.Sequence`: The sequence type for the training phase, can be `MersenneTwister`, `MersenneTwisterAntithetc`, `Sobol`, `Burley2020Sobol`, `SobolBrownianBridge`, `Burley2020SobolBrownianBridge`
2. `Training.Seed`: The seed for the random number generation in the training phase
3. `Training.Samples`: The number of samples to be used for the training phase
4. `Pricing.Sequence`: The sequence type for the pricing phase, same values allowed as for training
5. `Training.BasisFunction`: The type of basis function system to be used for the regression analysis, can be `Monomial`, `Laguerre`, `Hermite`, `Hyperbolic`, `Legendre`, `Chbyshev`, `Chebyshev2nd`
6. `BasisFunctionOrder`: The order of the basis function system to be used
7. `Pricing.Seed`: The seed for the random number generation in the pricing
8. `Pricing.Samples`: The number of samples to be used for the pricing phase. If this number is zero, no pricing run is performed, instead the (T0) NPV is estimated from the training phase (this result is used to fill the T0 slice of the NPV cube)
9. `BrownianBridgeOrdering`: variate ordering for Brownian bridges, can be `Steps`, `Factors`, `Diagonal`
10. `SobolDirectionIntegers`: direction integers for Sobol generator, can be `Unit`, `Jaeckel`, `SobolLevitan`, `SobolLevitanLemieux`, `JoeKuoD5`, `JoeKuoD6`, `JoeKuoD7`, `Kuo`, `Kuo2`, `Kuo3`
11. `MinObsDate`: if true the conditional expectation of each cashflow is taken from the minimum possible observation date (i.e. the latest exercise or simulation date before the cashflow's event date); recommended setting is `true`
12. `RegressionOnExerciseOnly`: if true, regression coefficients are computed only on exercise dates and extrapolated (flat) to earlier exercise dates; only for

backwards compatibility to older versions of the AMC module, recommended setting is `false`

Product Type	Model	Engine
Swap	CrossAssetModel	AMC
CrossCurrencySwap	CrossAssetModel	AMC
FxOption	CrossAssetModel	AMC
BermudanSwaption	LGM	AMC
MultiLegOption	CrossAssetModel	AMC

Table 12: AMC enabled products with engine and model types

Additional Features

As a side product the AMC module provides plain MC pricing engines for Bermudan Swaptions and a new trade type `MultiLegOption` with a corresponding MC pricing engine.

MC pricing engine for Bermudan swaptions

The following listing shows a sample configuration for the MC Bermudan Swaption engine. The model parameters are identical to the LGM Grid engine configuration. The engine parameters on the other hand are the same as for the AMC engine, see 4.11.1.

```xml
<Product type="BermudanSwaption">
  <Model>LGM</Model>
  <ModelParameters>
    <Parameter name="Calibration">Bootstrap</Parameter>
    <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
    <Parameter name="Reversion_EUR">0.0050</Parameter>
    <Parameter name="Reversion_USD">0.0030</Parameter>
    <Parameter name="ReversionType">HullWhite</Parameter>
    <Parameter name="VolatilityType">HullWhite</Parameter>
    <Parameter name="Volatility">0.01</Parameter>
    <Parameter name="ShiftHorizon">0.5</Parameter>
    <Parameter name="Tolerance">1.0</Parameter>
  </ModelParameters>
  <Engine>MC</Engine>
  <EngineParameters>
    <Parameter name="Training.Sequence">MersenneTwisterAntithetic</Parameter>
    <Parameter name="Training.Seed">42</Parameter>
    <Parameter name="Training.Samples">10000</Parameter>
    <Parameter name="Training.BasisFunction">Monomial</Parameter>
    <Parameter name="Training.BasisFunctionOrder">6</Parameter>
    <Parameter name="Pricing.Sequence">SobolBrownianBridge</Parameter>
    <Parameter name="Pricing.Seed">17</Parameter>
    <Parameter name="Pricing.Samples">25000</Parameter>
    <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
    <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
  </EngineParameters>
</Product>
```

Multi Leg Options / MC pricing engine

The following listing shows a sample MultiLegOption trade. It consists of

1. an option data block; this is optional, see below

2. a number of legs; in principle all leg types are supported, the number of legs is arbitrary and they can be in different currencies; if the payment currency of a leg is different from a floating index currency, this is interpreted as a quanto payoff

If the option block is given, the trade represents a Bermudan swaption on the underlying legs. If the option block is missing, the legs themselves represent the trade.

See [2] for limitations of the multileg option pricing engine.

```
<Trade id="Sample_MultiLegOption">
  <TradeType>MultiLegOption</TradeType>
  <Envelope>...</Envelope>
  <MultiLegOptionData>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <Style>Bermudan</Style>
      <Settlement>Physical</Settlement>
      <PayOffAtExpiry>false</PayOffAtExpiry>
      <ExerciseDates>
        <ExerciseDate>2026-02-25</ExerciseDate>
        <ExerciseDate>2027-02-25</ExerciseDate>
        <ExerciseDate>2028-02-25</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <LegData>
      <LegType>Floating</LegType>
      <Payer>false</Payer>
      <Currency>USD</Currency>
      <Notionals>
        <Notional>100000000</Notional>
      </Notionals>
      ...
    </LegData>
    <LegData>
      <LegType>Floating</LegType>
      <Payer>true</Payer>
      <Currency>EUR</Currency>
      <Notionals>
        <Notional>100000000</Notional>
      </Notionals>
      ...
    </LegData>
  </MultiLegOptionData>
</Trade>
```

The pricing engine configuration is similar to that of the MC Bermudan swaption engine, cf. 4.11.1, also see the following listing.

```
  <Product type="MultiLegOption">
  <Model>CrossAssetModel</Model>
  <ModelParameters>
    <Parameter name="Tolerance">0.0001</Parameter>
    <!-- IR -->
    <Parameter name="IrCalibration">Bootstrap</Parameter>
    <Parameter name="IrCalibrationStrategy">CoterminalATM</Parameter>
    <Parameter name="ShiftHorizon">1.0</Parameter>
    <Parameter name="IrReversion_EUR">0.0050</Parameter>
    <Parameter name="IrReversion_GBP">0.0070</Parameter>
    <Parameter name="IrReversion_USD">0.0080</Parameter>
    <Parameter name="IrReversion">0.0030</Parameter>
    <Parameter name="IrReversionType">HullWhite</Parameter>
    <Parameter name="IrVolatilityType">HullWhite</Parameter>
    <Parameter name="IrVolatility">0.0050</Parameter>
    <!-- FX -->
    <Parameter name="FxCalibration">Bootstrap</Parameter>
    <Parameter name="FxVolatility_EURUSD">0.10</Parameter>
    <Parameter name="FxVolatility">0.08</Parameter>
    <Parameter name="ExtrapolateFxVolatility_EURUSD">false</Parameter>
    <Parameter name="ExtrapolateFxVolatility">true</Parameter>
    <!-- Correlations IR-IR, IR-FX, FX-FX -->
    <Parameter name="Corr_IR:EUR_IR:GBP">0.80</Parameter>
    <Parameter name="Corr_IR:EUR_FX:GBPEUR">-0.50</Parameter>
    <Parameter name="Corr_IR:GBP_FX:GBPEUR">-0.15</Parameter>
  </ModelParameters>
  <Engine>MC</Engine>
  <EngineParameters>
    <Parameter name="Training.Sequence">MersenneTwisterAntithetic</Parameter>
    <Parameter name="Training.Seed">42</Parameter>
    <Parameter name="Training.Samples">10000</Parameter>
    <Parameter name="Pricing.Sequence">SobolBrownianBridge</Parameter>
    <Parameter name="Pricing.Seed">17</Parameter>
    <Parameter name="Pricing.Samples">25000</Parameter>
    <Parameter name="Training.BasisFunction">Monomial</Parameter>
    <Parameter name="Training.BasisFunctionOrder">4</Parameter>
    <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
    <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
  </EngineParameters>
</Product>
```

Model Parameters special to that product are

1. `IrCalibrationStrategy` can be `None`, `CoterminalATM`, `UnderlyingATM`
2. `FXCalibration` can be `None` or `Bootstrap`
3. `ExtrapolateFxVolatility` can be `true` or `false`; if false, no calibration instruments are used that require extrapolation of the market fx volatility surface in option expiry direction
4. `Corr_Key1_Key2`: These entries describe the cross asset model correlations to be used; the syntax for `Key1` and `Key2` is the same as in the simulation configuration for the cross asset model

4.11.2 Scripted Bermudan

Calling

```
python run_scriptedberm.py
```

demonstrates exposure simulation using AMC for selected scripted trade types

- Bermudan Swaption
- LPI Swap

Both payoffs are defined in the `scriptlibrary.xml` which are referenced in `portfolio_scriptedberm.xml` by the `ScriptName` tag.
To enable the AMC processing requires the following highlighted settings in `ore.xml`.

```
    <Analytic type="simulation">
      <Parameter name="active">Y</Parameter>
      <!-- Set to Y to trigger AMC processing -->
      <Parameter name="amc">Y</Parameter>
      <Parameter name="simulationConfigFile">simulation.xml</Parameter>
      <Parameter name="pricingEnginesFile">pricingengine.xml</Parameter>
      <!-- Specify a separate pricing engine file for AMC engines -->
      <Parameter name="amcPricingEnginesFile">pricingengine\_amc.xml</Parameter>
      <!-- Specify trade types to be covered by the AMC processing -->
      <Parameter name="amcTradeTypes">ScriptedTrade</Parameter>
      <Parameter name="baseCurrency">EUR</Parameter>
      <Parameter name="cubeFile">cube.csv.gz</Parameter>
      <Parameter name="aggregationScenarioDataFileName">scenariodata.csv.gz</Parameter>
      <Parameter name="aggregationScenarioDataDump">scenariodata.csv</Parameter>
    </Analytic>
```

Note that ORE can handle a mix of trades covered by AMC simulation and covered by "classic" simulation. The respective NPV cubes are combined before generating results such as exposures or XVAs.

4.11.3 Scripted TaRF

Calling

```
python run_scriptedtarf.py
```

demonstrates exposure simulation and XVA for another scripted product, an FX Target Redemption Forward (TaRF). In contrast to the cases presented above, you won't see the payoff script library in the Input folder, nor is the script embedded into the trade XML file. The trade type in this case is `FxTARF` which has its own implementation in `OREData/ored/portfolio/tarf.xpp` and a separate trade schema. However, the scripted trade framework is used under the hood, and the payoff script is embedded into the C++ code in OREData/ored/portfolio/tarf.cpp.

4.11.4 Forward Bond

Calling

```
python run_forwardbond.py
```

demonstrates AMC exposure simulation and XVA for a forward and a plain vanilla bond. The results can be compared to a classical exposure simulation by switchting the parameter *amc* in the main configuration file *ore.xml*, section analytic type *Simulation* to *N*.

For both cases, forward and plain vanilla bond, the security spread is implied from given bond prices. This requires a proper security specification in the curve configuration. Compare *Securities* node within *curveconfig.xml*. Both fields, PriceQuote and SpreadQuote, are mandatory. In the market data file, if the price quote is given in the absence of a spread quote, the spread is implied. Otherwise the spread quote is used or none.

In case of a forward bond, the convention for security specification requires the format *securityId_FWDEXP_expiryDate*.

4.11.5 Overlapping Close-out Grids in AMC

Calling

```
python run_overlapping.py
```

demonstrates the case where the primary and close-out grid overlap, using a daily simulation grid up to 15 days. This was not possible in releases earlier than ORE v13.

4.11.6 Extract Scenario Statistics

Calling

```
python run_scenariostatistics.py
```

demonstrates a basic statistical anylsis of the simulated raw scenario data (simulated yield curve discount factors, FX spot rates etc), generating a report with min, mean, max, standard deviation, skewness and kurtosis for each factor and at each simulated future date.

4.12 XVA Risk

The following XVA Risk-related examples can be found in folder `Examples/XvaRisk`. They can be run individually as discussed below, or all together with: `python run.py`

4.12.1 XVA Stress Testing

Calling

```
python run_stress.py
```

demonstrates the XVA stress testing analytic using both classical and AMC XVA engine, with output in `Examples/XvaRisk/Output/stress`.

The new analytic type *XVA_STRESS* utilizes the existing stresstest framework and supports stress tests in both zero and par domain. The stress test scenarios are given in the same input format as for the regular stresstest.

To analyse the impact of market rate shifts (Swap rates, CDS spreads, flat vols), one had to manipulate the market data input into ORE and re-run the entire ORE process multiple times.

The generated outputs are the xva and exposure reports under each scenario.

The XVA stress analytic replaces the todays market with a simulation market during the XVA calculation. For some risk factors the simulation market behaves different from todays market. Depending on the simulation and stress scenario settings it could use different tenors when building curves or use only ATM volatilities. It is recommended to activate `UseSpreadedTermStructures` in the `stresstest.xml` and to simulate SwaptionVolatilities for the XVA stress run.

4.12.2 XVA P&L Explain

Calling

```
python run_xvaexplain.py
```

demonstrates the XVA Explain Analytic with results in `XvaRisk/Output/explain`.

ORE can compute the market implied XVA change between two evaluation dates. For each risk factor defined in the sensitivity config ORE computes the par rate change between t0 and t0 + mporDays. ORE derives for each risk factor a shift scenario ($ParRate(t_1) - ParRate(t_0)$) and computes the CVA change implied by those risk factor shifts at t0.

Like the XVA Stress analytic, the XVA Explain analytic replaces the todays market with a simulation market during the XVA calculation. For some risk factors the simulation market behaves different from todays market. Depending on the simulation and stress scenario settings it could use different tenors when building curves or use only ATM volatilities. It is recommended to activate `UseSpreadedTermStructures` in `sensitivity_explain.xml` and to simulate SwaptionVolatilities for the XVA Explain run in `xvaexplainmarket.xml`

4.12.3 XVA Sensitivities

Calling

```
python run_sensi.py
```

demonstrates the XVA Sensitivity analytic with results in `XvaRisk/Output/sensi`, in particular `sacva.csv` and `sacvadetail.csv`.

The analytic computes XVA sensitivities by bump & revalue and generates the sensitivity scenarios from a sensitivity configuration similar to the one used in NPV sensitivity calculation. It utilises the existing XVA machinery "under the hood", supports XVA calculation using "classic" and AMC simulation. Sensitivities are generated in both the "raw" (e.g. zero rate) domain as well as in the par domain.

This analysis is run twice here, with "classic" and AMC simulation, for a small proof-of-concept "portfolio" consisting of two swaps. We are running an (unusual) uncollateralised case here. The user can activate both VM and IM as shown in previous examples.

For methodology summaries and the current scope of the implementations see [2].

Note:

- The XVA sensitivity analytic replaces the TodaysMarket with a SimulationMarket during the XVA calculation. This is a template for extending other ORE analytics (such as VaR) for exposing their evaluation under stressed markets.
- Realistic portfolios may require large numbers of sensitivity scenarios and hence XVA simulations which makes SA-CVA computationally challenging in practice. AMC simulation for XVA instead of "classic" simulation helps performance here. As a further performance enhancement we are working on a computation graph framework for XVA which supports GPU parallelization and AAD for XVA sensitivity, see example section 4.14.

4.12.4 CVA Capital: SA-CVA and BA-CVA

Calling

```
python run_sacva.py
```

demonstrates CVA Capital calculation using the Standard Approach (**SA-CVA**) with results in `XvaRisk/Output/sacva`. This analytic utilises the former XVA sensitivity analytic, in particular its CVA par sensitivity output. Alternatively, CVA sensitivity can passed as an input. The demonstration here picks up the sensitivities computed before by the XVA Sesnitivity analytic and written to `Output/sensi/amc/xva_par_sensitivity_cva.csv`, see `Input/ore_sacva.xml`.

Similarly, calling

```
python run_bacva.py
```

demonstrates CVA Capital calculation using the Basic Approach (**BA-CVA**) with results in `XvaRisk/Output/bacva`. BA-CVA is based on the SA-CCR Exposure At Default implementation, hence calls the SA-CCR analytic "under the hood". SA-CCR is covered in section 4.13.

4.13 Credit Risk

4.13.1 SA-CCR

Calling

```
python run_saccr.py
```

demonstrates the "Standard Approach Counterparty Credit Risk (SA-CCR)" Capital calculation for derivatives in ORE with results in `CreditRisk/Output/SA-CCR`, in particular `saccr.csv` and `saccr_detail.csv`. See [2] for the current implementaion's scope.

4.13.2 Credit Portfolio Model

The purpose of the credit portfolio model in ORE is to generate an integrated portfolio gain/loss distribution at a given future horizon which is driven by

- credit defaults and rating migrations in Bonds and CDS, and
- the PnL of a portfolio of derivatives over the specified time horizon.

The model integrates Credit and Market Risk by jointly evolving systemic credit risk drivers alongside the usual risk factors in ORE's Cross Asset Model. See also the separate documentation in `Docs/UserGuide/creditmodel.tex|pdf`. As mentioned there, this example will only run if the Eigen installation has been done before building ORE as described in section 3.4.

By running

```
python run_cpm.py
```

this example demonstrates the model's outcome for seven demo portfolios

Case	Credit Mode	Exposure Mode	Evaluation
Single Bond	Migration	Value	Analytic
Bond and Swap	Migration	Value	Analytic
3 Bonds	Migration	Value	Analytic
10 Bonds	Migration	Value	Analytic
10 Bonds	Migration	Value	Terminal Simulation
Bonds and CDS	Migration	Notional	Analytic
100 Bonds	Default	Notional	Analytic

The last demo case in this table can be activated by uncommenting the corresponding section at the end of the `run_cpm.py` script.

4.14 Performance

4.14.1 Multi-threading

Calling

```
python run_multithreading.py
```

demonstrates the parallelisation of exposure simulation for a portfolio of eight Vanilla Swap copies, results in folder `Performance/Output/multi`. The multi-threaded valuation engine in ORE splits the portfolio into n parts where n is the number of threads `nThreads` specified in `ore_multi.xml`.

4.14.2 NPV Sensitivities with AAD and GPUs

Calling

```
python run_sensi.py
```

demonstrates alternative ways of speeding up sensitivity calculations - using AAD or an external compute device. The test portfolio consists of

- Vanilla Equity Option
- Equity Barrier Option
- Equity Accumulator
- Asian Basket Option
- FX TaRFs

The sensitivity analysis is then run in four ways, see `run_sensi.py`,

- with "classic" bump and revalue ((19 sec on Apple M2 Max))
- as above but using the Computation Graph, see `UseCG=true` in `pricingengine_cg.xml`, which is the basis for the following two approaches ((14 sec on Apple M2 Max))
- using AAD, see `pricingengine_ad.xml` ((2.2 sec on Apple M2 Max))
- using the external device if available, see `pricingengine_gpu.xml` (2.3 sec on Apple M2 Max with "OpenCL/Apple/Apple M2 Max" device)

to compare sensitivities and performance. In the latter case we have set the external device in `pricingengine_gpu.xml` to "BasicCpu/Default/Default" which mimics an external device on the CPU. On a macbook pro (2023) with M2 Max processor, we can also choose "OpenCL/Apple/Apple M2 Max" here (a 38 core GPU). The Jupyter notebook `ore_aadsensi.ipynb` in this `Examples/Performance` folder also kicks off these four runs, but adds further commentary and visualises results. To run this notebook you need to build the Python bindings for release 12 (or later) or "pip install" ORE as discussed in section 4.15.

4.14.3 CVA Sensitivities with AAD

Calling

```
python run_cvasensi.py
```

demonstrates a prototype CVA sensitivity calculation applying AAD to a single Swap instrument with results in folder `Performance/Output/cvasensi`.

This script executes four batches

- a "base" CVA calculation using AMC for reference (about 9 sec on Apple M2 Max)
- a bump & revalue CVA sensitivity calculation (about 48 sec on Apple M2 Max)
- CVA sensitivity using AAD (about 2 sec on Apple M2 Max)
- CVA sensitivities with GPU parallelisation, porting the conditional expectation calculations to the external device is work in progress, hence no noticeable speedup to be reported yet

Run the corresponding Juypter notebook with: `python -m jupyterlab ore_cvasensi.ipynb &`

4.15 ORE-Python

Since release 9 (March 2023) we provide easy access to ORE via a pre-compiled Python module. Some example scripts using this ORE module are provided in this example, so change to this directory first

```
cd Examples/ORE-Python
```

The examples require Python 3. The ORE Python module is then installed with a one-liner, see step 3 below. However, to separate ORE from any other Python environments on your machine, we recommend creating a virtual environment first. In that case the steps are as follows.

1. To create a virtual environment: `python -m venv env1`
2. To activate this environment on Windows: `.\env1\Scripts\activate.bat`
   or on macOS/Linux: `source env1/bin/activate`
3. Then install the latest release of ORE:
   `pip install open-source-risk-engine`
4. Try examples:
   - `python ore.py`
     This demonstrates the Python-wrapped version of the ORE application that is also used in the command line application `ore.exe`. We use it here to re-run the Swap exposure of `Example_1`.
   - `python ore2.py`

    This extends the previous example and shows how to access and post-process ORE in-memory results in the Python framework without reading files.

  - `python commodityforward.py`
    The ORE Python module also allows lower-level access to the QuantLib and QuantExt libraries, demonstrated here for a CommodityForward instrument defined in QuantExt. Note that the ORE Python module contains the entire QuantLib Python functionality.

   More use cases of the ORE Python module including Jupyter notebooks can be found in the ORE SWIG repository, in particular in folder OREAnalytics-SWIG/Python/Examples.

5. You can deactivate the environment with `deactivate`
   or even fully remove the environment again by removing the `env1` folder.

Finally, you can build the Python module and installable packages yourself following the instructions in sections 3.5 based on your local ORE code.

Jupyter Notebook Examples

With ORE release 13 we have merged the ORE-SWIG source code into the ORE repository and moved a range of related Jupyter notebook examples into this ORE-Python folder, see folders Notebooks/Example_1 to Notebooks/Example_9.

To try, change e.g. to Notebooks/Example_1 and launch the notebook there by calling

```
python3 -m jupyterlab ore.ipynb &
```

If not installed yet, this might require a

```
python3 -m pip install jupyterlab
```

and some of the Jupyter notebooks require additional packages to be installed with pip:

- matplotlib
- pandas
- numpy
- scipy
- ipywidgets

4.16 ORE-API

Since release 12, ORE comes with a proof-of-concept implementation of a web service around ORE that is written in Python, see folder `Examples/ORE-API`.

The service is based on

- the flask web framework `https://flask.palletsprojects.com`, and

- the ORE Python module which can be installed using
  `pip install open-source-risk-engine`
  or built from sources following the instructions in section 3

Main files in the `Examples/API` directory:

- **restapi.py** runs a flask api as analytics service that takes a post request (e.g. from a server like postman, or from a python script): The request contains a json body which corresponds to the data usually contained in the master input file ore.xml. restapi.py calls the oreApi.py class (see below) to do the work. By default, the analytics service listens for requests on port 5001.
- **oreApi.py** reads the json body, compiles all ORE input parameters, calls into a data service (see below) to retrieve additional data. Then it kicks off an ORE run to process the request. Finally it posts resulting reports through the data service.
- **simplefileserver.py** runs a flask api as a data service. It takes requests from the analytics service above in the form of urls of xml files that contain additional data required by ORE (market data, portfolio, configuration). By default, the data service listens on port 5000, reads from the Input directory in Examples/API and writes reports to the Output directory in Examples/API.

Run a local example:

- start the data service:
  `python3 simplefileserver.py &`
- start the analytics service:
  `python3 restapi.py &`
- send a request to run the equivalent of Example 1:
  `python3 request.py`
  Note that the json equivalent of the usual `ore.xml` is contained in request.py, and all other inputs are retrieved from folder Examples/API/Input via the data service.

5 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 4 respectively Table 5. 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 4 as a boilerplate and just change the folders, see section 5.1, and the trade data, see [1], together with the netting definitions, see section 8.

5.1 Analytics: `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
- Logging
- Markets
- Analytics

which we will explain in the following.

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="useAtParCouponsCurves">Y</Parameter>
  <Parameter name="useAtParCouponsTrades">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="counterpartyFile">counterparty.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>
  <Parameter name="buildFailedTrades">true</Parameter>
  <Parameter name="nThreads">4</Parameter>
  <Parameter name="enrichIndexFixings">false</Parameter>
  <Parameter name="ignoreFixingLead">0</Parameter>
  <Parameter name="ignoreFixingLag">0</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[2]. 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`. The counterparty information in `counterparty.xml` covers minimal counterparty-level information needed in a few capital analytics as shown in example sections 4.6 and 4.13. 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.

The parameters `useAtParCouponsCurves` and `useAtParCouponsTrades` control whether to use par approximation or indexed ibor coupons when building curves or building ore trades, respectively. This goes back to the QuantLib flag `QL_USE_INDEXED_COUPON` and the associated runtime setting. The default is `true` for both flags which is also the default setting when building QuantLib.

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;

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

- 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`.

If the parameter `buildFailedTrades` is set to true, the application will build a dummy trade if loading or building the original trade fails. The dummy trade has trade type "Failed", zero notional and NPV. If not given, the parameter defaults to `false`.

If the parameter `nThreads` is given, multiple threads will be used for valuation engine runs where applicable (Sensitivity, Exposure Classic, Exposure AMC). If not given, the parameter defaults to 1.

If the parameter `enrichIndexFixings` is set to true, the application will fill the gaps in index fixings, by fallback fixings, which are the previous fixings (priority) or the next fixings. If not given, the parameter defaults to `false`.

If the next fixing date leads by more than Parameter `ignoreFixingLead`, the fixing would not be used as the fallback. A `0` `ignoreFixingLead` disables the check. If not given, the parameter defaults to `0`.

If the previous fixing date lags by more than Parameter `ignoreFixingLag`, the fixing would not be used as the fallback. A `0` `ignoreFixingLag` disables the check. If not given, the parameter defaults to `0`.

5.1.1 Logging

The `Logging` section (see listing 4) is used to configure some ORE logging options.

Listing 4: ORE logging

```
<Logging>
  <Parameter name="logFile">log.txt</Parameter>
  <Parameter name="logMask">31</Parameter>
  <Parameter name="progressLogFile">my_log_progress_%N.json</Parameter>
  <Parameter name="progressLogRotationSize">102400</Parameter>
  <Parameter name="progressLogToConsole">false</Parameter>
  <Parameter name="structuredLogFile">my_structured_logs_%N.txt</Parameter>
  <Parameter name="structuredLogRotationSize">102400</Parameter>
</Logging>
```

Parameter `logFile` and `logMask` will override the same parameters in the `Setup` section. Parameters `progressLogFile` and `structuredLogFile` are the filename where progress log messages and structured log messages are written out to, respectively, which supports Boost string patterns.This defaults to "log_progress_%N.json" and "log_structured_%N.json", respectively, where `N` will be an integer (beginning at 0) used for log file rotation. Parameters `progressLogRotationSize` and `structuredLogRotationSize` are the size limit (in bytes) of each log file before applying log file rotation to the progress log file and structured log message file, respectively.. For example, $10 * 1024 * 1024 = 10\text{MiB}$. Defaults to 100 MiB. If the parameter `progressLogToConsole` is set to true, then progress logs will be written to std::cout. This can be used simultaneously with `progressLogFile`, i.e. progress logs can be written out to both file and std::cout.

Markets

The `Markets` section (see listing 5) 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 5.2).

Listing 5: 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.

Analytics

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

- NPV, Cashflows, Curves
- Exposure Simulation, Model Calibration, Scenario Generation
- Value Adjustments (xVA)
- Sensitivity, Stress
- Value at Risk
- P&L, P&L Explain, Scenario
- ISDA SIMM, IM Schedule, IR/FX CRIF generation for ISDA SIMM
- Par Conversion, Par Stress Conversion, Par Scenario, Zero Shift to Par Shift
- XVA Stress, XVA Sensitivities, XVA Explain
- SA-CCR, SA-CVA, BA-CVA, SMRC
- Portfolio Details
- Correlation

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.

5.1.2 Pricing, Cashflows, Curves

The following listing 6 shows the parametrisation of the first three basic analytics in the list above.

Listing 6: ORE analytics: npv, cashflow, curves

```
<Analytics>
  <Analytic type="npv">
    <Parameter name="active">Y</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="outputFileName">npv.csv</Parameter>
    <Parameter name="additionalResults">Y</Parameter>
    <Parameter name="additionalResultsReportPrecision">12</Parameter>
  </Analytic>
  <Analytic type="cashflow">
    <Parameter name="active">Y</Parameter>
    <Parameter name="outputFileName">flows.csv</Parameter>
    <Parameter name="includePastCashflows">N</Parameter>
  </Analytic>
  <Analytic type="curves">
    <Parameter name="active">Y</Parameter>
    <Parameter name="configuration">default</Parameter>
    <Parameter name="grid">240,1M</Parameter>
    <Parameter name="calendar">TARGET</Parameter>
    <Parameter name="outputFileName">curves.csv</Parameter>
    <Parameter name="outputTodaysMarketCalibration">Y</Parameter>
    <Parameter name="todaysMarketCalibrationPrecision">8</Parameter>
  </Analytic>
  <Analytic type="...">
    <!-- ... -->
  </Analytic>
</Analytics>
```

The cashflow analytic writes a report containing all future (and optionally past) cashflows of the portfolio. Table 13 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 13: 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 optional key `calendar` defines the calendar that is used to roll out the grid, it defaults to TARGET. 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 `outputTodaysMarketCalibration` parameter controls whether calibration information for the built market objects is written to a report. When set to Y, a detailed calibration report is generated. The `todaysMarketCalibrationPrecision` parameter specifies the number of decimal places to use when formatting floating-point values in the market calibration report (default: 8). This allows users to control the precision of numerical output in the calibration report, similar to the `additionalResultsReportPrecision` parameter for the NPV analytic.

The additionalResults analytic writes a report containing any additional results generated for the portfolio. The results are pricing engine specific but Table 14 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 14: AdditionalResults Report

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

5.1.3 Simulation and Model Calibration

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 Exposure and XVA analytics.

ORE supports NPV cube generation using both a "classic" Monte Carlo simulation as sketched above, as well as American Monte Carlo (AMC), see below. The setup in 7 refers to a classic run.

Listing 7: ORE analytic: simulation

```
<Analytics>
  <Analytic type="simulation">
    <Parameter name="simulationConfigFile">simulation.xml</Parameter>
    <Parameter name="pricingEnginesFile">../../Input/pricingengine.xml</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="cubeFile">cube.csv.gz</Parameter>
    <Parameter name="nettingSetCubeFile">nettingSetCube.csv.gz</Parameter>
    <Parameter name="scenariodump">scenariodump.csv</Parameter>
    <Parameter name="aggregationScenarioDataFileName">scenariodata.csv.gz</Parameter>
    <Parameter name="storeFlows">Y</Parameter>
    <Parameter name="cptyCubeFile">cptyCube.csv.gz</Parameter>
    <Parameter name="storeSurvivalProbabilities">Y</Parameter>
    <Parameter name="storeCreditStateNPVs">8</Parameter>
    <Parameter name="cubeNpvOverlay">true</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 for writing the (trade-level) cube and nettingSetCube files. The scenario dump file name, if provided here, causes ORE to export full "raw" market scenarios for later inspection/use, i.e. discount factors for yield curves. The aggregationScenarioData file name, if provided here, causes ORE to write furthermore selected market data (simulated FX rates and index fixings, which might be needed in the XVA postprocessor for Variation Margin calculations) to a zipped csv. The selection is specified in the simulation config file, see AggregationScenarioDataCurrencies, AggregationScenarioDataIndices), see also section 5.3.3. 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 some Dynamic Initial Margin and Variation Margin models. 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. Key 'cubeNpvOverlay' is optional and defaults to false. If true, all raw npv cube entries are corrected by the difference of the T0 npv from the pricing analytic and the T0 npv from the simulation npv.

To use AMC simulation the simulation setup needs the additional elements shown in 8

Listing 8: ORE analytic: simulation

```
<Analytics>
  <Analytic type="simulation">
    ...
    <Parameter name="amc">Y</Parameter>
    <Parameter name="amcTradeTypes">Swap</Parameter>
    <Parameter name="amcPricingEnginesFile">pricingengine_amc.xml</Parameter>
    ...
  </Analytic>
</Analytics>
```

The amcTradeTypes element is a comma-separated list of ORE trade types to be covered by AMC in this run. Permissible types are

- Swap
- Swaption
- FxForward
- FxOption
- ForwardBond
- FxTaRF
- ScriptedTrade
- CompositeTrade

The remaining products in the portfolio (if any) are covered using classic Monte Carlo. ORE then combines the classic and AMC cubes.

Further extensions to the simulation setup are available and demonstrated in related example sections (AmericanMonteCarlo, InitialMargin, Performance), e.g. for

- using the Computation Graph based implementation of AMC
- utilising an external compute device to boost ORE performance
- storing analytical sensitivities for selected products along paths (e.g. for a version of Dynamic Initial Margin)
- generating path-wise sensitivities using AAD (e.g. for a version of Dynamic Initial Margin)
- computing XVA t_0 sensitivities using AAD

The purpose of the `calibration` 'analytics' is to run a subset of the simulation analytic's functionality, i.e. to calibrate the simulation model and output the calibrated model data such that it can be used to initialize a subsequent simulation without recalibration.

Both CAM and HW model are supported for `calibration` 'analytics'. When setting model to CAM, ORE use the following inputs.

Listing 9: ORE analytic: CAM calibration

```
<Analytics>
    <Analytic type="calibration">
      <Parameter name="active">Y</Parameter>
          <Parameter name="model">CAM</Parameter>
      <Parameter name="configFile">simulation.xml</Parameter>
      <Parameter name="outputFileName">calibration.csv</Parameter>
    </Analytic>
</Analytics>
```

See `Example_8` for a demonstration.

When setting model to HW, ORE use the following inputs.

Listing 10: ORE analytic: HW calibration

```
<Analytics>
    <Analytic type="calibration">
      <Parameter name="active">Y</Parameter>
          <Parameter name="model">HW</Parameter>
          <Parameter name="mode">historical</Parameter>
          <Parameter name="foreignCurrencies">EUR,GBP</Parameter>
          <Parameter name="curveTenors">1Y,2Y,3Y,5Y,10Y,15Y,20Y,30Y</Parameter>
          <Parameter name="useForwardOrZeroRate">zero</Parameter>
          <Parameter name="pcaCalibration">Y</Parameter>
      <Parameter name="scenarioInputFile">scenario.csv</Parameter>
          <Parameter name="startDate">2019-09-29</Parameter>
          <Parameter name="endDate">2022-09-27</Parameter>
          <Parameter name="lambda">1</Parameter>
          <Parameter name="varianceRetained">0.90</Parameter>
          <Parameter name="pcaOutputFileName">pca.csv</Parameter>
          <Parameter name="pcaInputFileName">pca_USD.csv,pca_EUR.csv,pca_GBP.csv</Parameter>
          <Parameter name="meanReversionCalibration">Y</Parameter>
          <Parameter name="basisFunctionNumber">2</Parameter>
          <Parameter name="kappaUpperBound">5.0</Parameter>
          <Parameter name="haltonMaxGuess">500</Parameter>
          <Parameter name="meanReversionOutputFileName">meanReversion.csv</Parameter>
    </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `model`: The model for Calibration, can be either `CAM` or `HW`, default to `CAM` when left blank or omitted.
- `mode`: The calibration mode, must be `historical` for now. Will be ignored if `model` node set to `CAM`.
- `foreignCurrencies`: The list of foreign currencies that need the calibration. Will be ignored if `model` node set to `CAM`.
- `curveTenors`: The list of tenors for each IR curve when providing historical discount factors or the tenor list of each value in eigenvectors. Will be ignored if `model` node set to `CAM`.
- `useForwardOrZeroRate`: When `pcaCalibration` set to `true`, this means whether zero rate or forward rate will be used to calculate the covariance matrix for PCA calibration. When `pcaCalibration` set to `false`, this means whether zero rate or forward rate should be used to parse the provided PCA eigenvectors for mean reversion calibration.
- `pcaCalibration`: When set to `true`, ORE will read the historical discount factors and perform PCA calibration. Will be ignored if `model` node set to `CAM`.
- `scenarioInputFile`: The path to the file which contains the historical discount factors and historical FX spot rates. Must be in the scenario.csv ORE format. Only needed when `pcaCalibration` set to `true`.

- `startDate`: Start date of the historical data that will be used in calibration. Only needed when `pcaCalibration` set to `true`.
- `endDate`: End date of the historical data that will be used in calibration. Only needed when `pcaCalibration` set to `true`.
- `lambda`: The lambda used on exponentially-weighted historical rates diffs when computing the covariance matrix. The covariance matrix will be equally weighted when `lambda` is set to `1`. Only needed when `pcaCalibration` set to `true`.
- `varianceRetained`: The ratio between the combined variance of retained principal components and total variance. This decides how many principal components will be retained. Only needed when `pcaCalibration` set to `true`.
- `pcaOutputFileName`: The output file name of eigenvalues and eigenvectors retained for each currency.
- `pcaInputFileName`: The list of file names where eigenvalues and eigenvectors are provided for mean reversion calibration. The files should be one curve per file. Only needed when `pcaCalibration` is set to `false` and `meanReversionCalibration` is set to `true`.
- `meanReversionCalibration`: When set to `true`, ORE will perform mean reversion calibration on principal components retained either from previous pca calibration step or from input files.
- `basisFunctionNumber`: Number of basis functions used in mean reversion calibration.
- `kappaUpperBound`: The upper bound of kappa during mean reversion calibration.
- `haltonMaxGuess`: Number of max guess in optimization in mean reversion calibration.
- `meanReversionOutputFileName`: The output file name for kappa and v for each curve.

For both CAM and HW model calibration, output is the Cross Asset Model data written to `calibration.xml` in the usual output directory, which contains the calibration results in place of the initial values for all parametrizations covered so far (IR, FX, EQ, INF, COM). In a subsequent run one could replace the `CrossAssetModel` section in `simulation.xml` with the output in `calibration.xml` to re-run without re-calibration. Note that the `Calibrate` flags in the output Cross Asset Model data are set to `false`.

Additionally, the Cross Asset Model XML is written as a single XML string to the calibration report `calibration.csv`, also held in memory for further processing e.g. via ORE's Python interface.

For HW model, there is an additional output `calibration_StatisticalWithRiskNeutralVolatility.xml` which is used for multi-factor sigma risk neutral calibration. Similarly, `calibration_StatisticalWithRiskNeutralVolatility.csv` is also outputted.

5.1.4 Scenario Generation

The `scenarioGeneration` 'analytics' generates scenarios and provides the statistics and distribution of the scenarios generated. Listing **??** shows a typical configuration for sensitivity calculation.

Listing 11: ORE analytic: scenarioGeneration

```
<Analytics>
  <Analytic type="scenarioGeneration">
        <Parameter name="active">Y</Parameter>
        <Parameter name="simulationConfigFile">simulation.xml</Parameter>
        <Parameter name="distributionBuckets">20</Parameter>
        <Parameter name="outputZeroRate">Y</Parameter>
        <Parameter name="scenariodump">scenariodump.csv</Parameter>
  </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `simulationConfigFile:` Configuration file defining the simulation market under which sensitivities are computed, see 5.3.
- `distributionBuckets:` Number of buckets used for the distribution histogram.
- `outputZeroRate:` Determine whether the statistics report and distribution report will use zero rate or discount factors. If set to Y, the reports will use zero rates. If set to N, they will use discount factors.
- `scenariodump:` File containing all the scenarios generated through simulation market. If the node is not given, this file will not be outputted.

5.1.5 Value Adjustments

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. The cube file is usually a compressed csv file (using gzip compression, with file ending .csv.gz), except when the file extension is set explicitly to txt or csv in which case an uncompressed version of the file is written to disk.

Listing 12: ORE analytic: xva

```
<Analytics>
  <Analytic type="xva">
    <Parameter name="active">Y</Parameter>
    <Parameter name="csaFile">netting.xml</Parameter>
    <Parameter name="cubeFile">cube.csv.gz</Parameter>
    <Parameter name="useDoublePrecisionCubes">false</Parameter>
    <Parameter name="nettingSetCubeFile">nettingSetCube.csv.gz</Parameter>
    <Parameter name="cptyCubeFile">cptyCube.csv.gz</Parameter>
    <Parameter name="scenarioFile">scenariodata.csv.gz</Parameter>
    <Parameter name="collateralBalancesFile">collateralbalances.xml</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">NoLag</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="dimModel">Regression</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>
    <Parameter name="flipViewXVA">N</Parameter>
    <Parameter name="flipViewBorrowingCurvePostfix">_BORROW</Parameter>
    <Parameter name="flipViewLendingCurvePostfix">_LEND</Parameter>
    <Parameter name="mporCashFlowMode">NonePay</Parameter>
    <Parameter name="generateCorrelations">N</Parameter>
  </Analytic>
</Analytics>
```

The PFE analytic type is the same as the XVA analytic, however it only returns results specific to PFE and only requires a subset of the XVA parameters.

Listing 13: ORE analytic: xva

```
<Analytics>
  <Analytic type="pfe">
    <Parameter name="active">Y</Parameter>
    <Parameter name="csaFile">netting.xml</Parameter>
    <Parameter name="cubeFile">cube.csv.gz</Parameter>
    <Parameter name="useDoublePrecisionCubes">false</Parameter>
    <Parameter name="scenarioFile">scenariodata.csv.gz</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="rawCubeOutputFile">rawcube.csv</Parameter>
    <Parameter name="netCubeOutputFile">netcube.csv</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
- `useDoublePrecisionCubes`: whether NPV cubes are constructed wit double precision, optional, defaults to false (single precision)
- `scenarioFile`: Scenario data previously generated and used in the post-processor (simulated index fixings and FX rates)
- `collateralBalancesFile`: References an xml file that contains current VM and IM balances by netting set
- `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. The admissible choices depend on having a close-out grid, see table 15;
  - *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 5.3.
  - if there isn't any "close-out" grid -see section 5.3-, the 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.
  - If there is a "close-out" grid -see section 5.3-, only choice is:
    * *NoLag* - used to disable any delayed settlement of the margin.

  NoLag is the default configuration.

Grid Type	`calculationType`	Comment
without close-out grid	*NoLag*	Not Supported
	Symmetric	Supported[3]
	AsymmetricCVA	Supported [3]
	AsymmetricDVA	Supported [3]
with close-out grid	*NoLag*	Supported[4]
	Symmetric	Not Supported
	AsymmetricCVA	Not Supported
	AsymmetricDVA	Not Supported

Table 15: Overview of admissible calculation types with combination of grid types.

- `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.

[3] The calculations are correct only if the simulation grid (see section 5.3) is equally-spaced with time steps that match the MPoR defined in netting-set definition (see section 8.2). See [2] for further explanation.

[4]Close-out lag (see section 5.3) must be equal to MPoR defined in netting-set definition (see section 8.2). Otherwise, an error will be thrown.

- `dimModel`: Type of dynamic initial margin model to be applied – *Regression* or *Flat*. *Regression* is applied by default when the dimModel node is omitted (see appendix and further settings related to the regression DIM model below); *Flat* means a simple flat projection of todays's IM amount on each path (this requires providing today's IM using the `collateralBalancesFile` parameter, see above)
- `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 probabilities 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 [2]).
- `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), leave empty to skip generation of this file.
- `netCubeOutputFile`: File name for the aggregated NPV cube in human readable csv file format (per netting set, date, sample) *after* taking collateral into account. Leave empty to skip generation of this file.
- `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.
- `flipViewXVA`: If set to `Y`, the perspective in XVA calculations is switched to the cpty view, the npvs and the netting sets being reverted during calculation. In order to get the lending/borrowing curve, the calculation assumes these curves being set up with the cptyname + the postfix given in the next two settings.
- `flipViewBorrowingCurvePostfix`: postfix for the borrowing curve, the calculation assumes this is curves being set up with cptyname + postfix given.
- `flipViewLendingCurvePostfix`: postfix for the lending curve, the calculation assumes this is curve being set up with cptyname + postfix given.
- `mporCashFlowMode`: Assumption about payment of cashflows within mpor period. One of NonePay, BothPay, WePay, TheyPay, Unspecified. Defaults to Unspecified, in this case PP will assume NonePay if mpor sticky date is used, otherwise to BothPay.

The two cube file outputs `rawCubeOutputFile` and `netCubeOutputFile` are provided for further analysis.

5.1.6 Sensitivity and Stress Testing

The `sensitivity` and `stress` 'analytics' provide computation of bump and revalue sensitivities and NPV changes under user defined stress scenarios. Listing 14 shows a typical configuration for sensitivity calculation.

Listing 14: 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>
   <!-- Additional parametrisation for par sensitivity analysis -->
   <Parameter name="parSensitivity">Y</Parameter>
   <Parameter name="parSensitivityOutputFile">parsensitivity.csv</Parameter>
   <Parameter name="outputJacobi">Y</Parameter>
   <Parameter name="jacobiOutputFile">jacobi.csv</Parameter>
   <Parameter name="jacobiInverseOutputFile">jacobi_inverse.csv</Parameter>
   <Parameter name="decomposeIndexSensitivities">Y</Parameter>
 </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `marketConfigFile:` Configuration file defining the simulation market under which sensitivities are computed, see 5.3. Only a subset of the specification is needed (the one given under `Market`, see 5.3.3 for a detailed description).
- `sensitivityConfigFile:` Configuration file for the sensitivity calculation, see section 5.4.
- `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 together with the risk-factor and the resulting first and (pure) second order finite differences. Also included is a second set of shift sizes together with the risk-factor with 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
- `parSensitivity`: If set to Y, par sensitivity analysis is performed following the "raw" sensitivity analysis; note that in this case the `sensitivityConfigFile` needs to contain `ParConversion` sections, see `Example_40`
- `parSensitivityOutputFile`: Output file name for the par sensitivity report

- `outputJacobi`: If set to Y, then the relevant Jacobi and inverse Jacobi matrix is written to a file, see below
- `jacobiOutputFile`: Output file name for the Jacobi matrix
- `jacobiInverseOutputFile`: Output file name for the inverse Jacobi matrix
- `decomposeIndexSensitivities`: Decompose Credit index and Equity and Commodity index sensitivities into constituent sensitivities

The decomposition of index sensitivities is controlled as follows:

- **Credit Index Decomposition:** Enabled via the `SensitivityDecomposition` flag in the pricing engine settings for `IndexCreditDefaultSwap`, `IndexCreditDefaultSwapOption`, and `SyntheticCDO` (see pricing engine parameterisation, Section **??**).
- **Equity and Commodity Index Decomposition:** Controlled at the trade level by adding the field `index_decomposition` to the trade envelope's `AdditionalFields` section, as shown in Listing 15.

Listing 15: Trade envelope configuration for equity and commodity index decomposition

```
<Envelope>
  ...
  <AdditionalFields>
    <index_decomposition>true</index_decomposition>
  </AdditionalFields>
</Envelope>
```

Equity index decomposition is supported for `EquitySwap` and `GenericTRS` trades, while commodity index decomposition is available for `GenericTRS` trades.

To enable index decomposition, appropriate reference data for credit, equity, or commodity indices must be provided.

The conversion of "raw" to "par" sensitivities, embedded above, can also be performed as a separate step by calling the `zeroToParSensiConversion` analytic as shown in Listing 16. The raw sensitivities are passed in (parameter sensitivityInputFile), all other parameters as above.

Listing 16: ORE analytic: Par Conversion

```
<Analytics>
  <Analytic type="zeroToParSensiConversion">
    <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="sensitivityInputFile">sensitivity.csv</Parameter>
    <Parameter name="outputThreshold">0.000001</Parameter>
    <Parameter name="outputFile">parconversion_sensitivity.csv</Parameter>
    <Parameter name="outputJacobi">Y</Parameter>
    <Parameter name="jacobiOutputFile">jacobi.csv</Parameter>
    <Parameter name="jacobiInverseOutputFile">jacobi_inverse.csv</Parameter>
  </Analytic>
</Analytics>
```

The parameters have the same interpretation as for the sensitivity analytic. There is one new parameter *sensitivityInputFile* which points to a csv file with the raw (zero)sensitivities. Those raw sensitivities will be converted into par sensitivities, using the same methodology as in the embedded approach above, and the configuration is described in 5.4. The raw sensitivities csv input file *sensitivityInputFile* needs to have at least six columns, the column names can be user configured in the master input file. Here is a description of each of the columns:

1. idColumn: Column with a unique identifier for the trade / nettingset / portfolio.
2. riskFactorColumn: Column with the identifier of the zero/raw sensitiviy. The risk factor name needs to follow the ORE naming convention, e.g. DiscountCurve/EUR/5/1Y (the 6th bucket in EUR discount curve as specified in the sensitivity.xml)
3. deltaColumn: The raw sensitivity of the trade/nettingset / portfolio with respect to the risk factor
4. currencyColumn: The currency in which the raw sensitivity is expressed, need to be the same as the BaseCurrency in the simulation settings.
5. shiftSizeColumn: The shift size applied to compute the raw sensitivity, need to be consistent to the sensitivity configuration.
6. baseNpvColumn: The base npv of the trade / nettingset / portfolio in currency.

Here is an example for an input file:

	#TradeId	Factor_1	ShiftSize_1	Currency	Base NPV	Delta
0	Swap	DiscountCurve/EUR/3/6M	0.0001	EUR	1335.27	5.05
1	Swap	DiscountCurve/EUR/4/9M	0.0001	EUR	1335.27	0.35
2	Swap	DiscountCurve/EUR/5/1Y	0.0001	EUR	1335.27	-5.41
3	Swap	DiscountCurve/EUR/6/2Y	0.0001	EUR	1335.27	-0.22
4	Swap	DiscountCurve/EUR/7/3Y	0.0001	EUR	1335.27	-0.32

The `stress` analytics configuration is similar to the one of the sensitivity calculation. Listing 17 shows a configuration example.

Listing 17: 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="precision">6</Parameter>
   <Parameter name="outputThreshold">0.000001</Parameter>
   <Parameter name="stressZeroScenarioDataFile">zeroStressScenarioData.xml</Parameter>
   <Parameter name="generateCashflows">true</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 5.5. That file will also determine whether the stress test is performed in the “raw” or “par” domain. In the latter par stress case, the last parameter (stressZeroScenarioDataFile) causes exporting equivalent stress test definitions in the “raw” domain. The `precision` parameter defines the number of digits for the sensitivities in the stress output file.

Finally, the `parStressConversion` analytic carves out the generation of a stress test configuration in the “raw” domain from a stress test configuration in the par domain, see Listing 18, with same parameters as in Listing 17 above. This analytic does not perform a stress test, just generates the raw domain stress configuration so that it can be applied repeatedly .

Listing 18: ORE analytic: Par Stress Conversion

```
  <Analytics>
    <Analytic type="parStressConversion">
       <Parameter name="active">Y</Parameter>
       <Parameter name="marketConfigFile">simulation.xml</Parameter>
       <Parameter name="stressConfigFile">stresstest.xml</Parameter>
       <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
       <Parameter name="pricingEnginesFile">pricingengine.xml</Parameter>
       <Parameter name="scenarioOutputFile">stresstest.csv</Parameter>
       <Parameter name="outputThreshold">0.000001</Parameter>
       <Parameter name="stressZeroScenarioDataFile">results.xml</Parameter>
     </Analytic>
  </Analytics>
```

The new analytic type *SENSITIVITY_STRESS* combines the existing stresstest and sensitivity analysis frameworks. During the sensitivity calculation it replaces todaysMarket with a SimulationMarket in accordance with the stresstest scenario and runs sensitivity analytic afterwards. The analytic loops over all provided stresstest scenarios.

Listing 19: ORE analytic: Stressed Sensitivity Analysis

```
    <Analytics>
      <Analytic type="sensitivityStress">
        <Parameter name="active">Y</Parameter>
        <Parameter name="marketConfigFile">simulation.xml</Parameter>
        <Parameter name="stressConfigFile">stresstest.xml</Parameter>
        <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
        <Parameter name="calcBaseScenario">N</Parameter>
      </Analytic>
    </Analytics>
```

The parameters have the following interpretation:

- `marketConfigFile`: Configuration file defining the simulation market under which sensitivities are computed, see 5.3. Only a subset of the specification is needed (the one given under `Market`, see 5.3.3 for a detailed description).
- `stressConfigFile`: Stress Scenario definition, see section 5.5
- `sensitivityConfigFile`: Configuration file for the sensitivity calculation, see section 5.4.
- `calcBaseScenario`: If set to true, unshifted BASE scenario will also be calculated. Defaults to false.

See the examples in sections 4.6 for a demonstrations of these analytics.

5.1.7 Value at Risk

The `VaR` analytics provide computation of Value-at-Risk measures based on the sensitivity (delta, gamma, cross gamma) data above. Listing 20 shows a configuration example.

Listing 20: 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="SalvagingAlgorithm">None</Parameter>
      <Parameter name="quantiles">0.01,0.05,0.95,0.99</Parameter>
      <Parameter name="breakdown">Y</Parameter>
      <!-- Delta, DeltaGammaNormal, Cornish-Fisher, Saddlepoint, 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;
- `SalvagingAlgorithm:` Allowable values are: *None*, *Spectral*, *Hypersphere*, *LowerDiagonal* or *Highham*. If omitted, it defaults to None. Compare [15].
- `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 [2]
- `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

See the example in section 4.6.3 for a demonstration.

5.1.8 Correlation

The `Correlation` analytic provide computation of the correlation matrix based on the generated historical scenarios. Listing 21 shows a configuration example.

Listing 21: ORE analytic: Correlation

```
<Analytics>
  <Analytic type="correlation">
      <Parameter name="active">Y</Parameter>
      <Parameter name="correlation_method">Pearson</Parameter>
      <Parameter name="marketConfigFile">simulation.xml</Parameter>
      <Parameter name="historicalScenarioFile">scenarios.csv</Parameter>
      <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
      <Parameter name="historicalPeriod">2016-12-30,2019-12-30</Parameter>
      <Parameter name="mporDays">10</Parameter>
      <Parameter name="mporCalendar">USD</Parameter>
      <Parameter name="outputFile">correlation.csv</Parameter>
    </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `correlation_method`: Regular expression used to filter the portfolio for which VaR is computed; if the filter is not provided, then the full portfolio is processed
- `marketConfigFile`: Configuration file defining the simulation market under which sensitivities are computed, see 5.3. Only a subset of the specification is needed (the one given under `Market`, see 5.3.3 for a detailed description).
- `historicalScenarioFile`: csv file containing the market scenarios for each date in the observation periods defined below; the granularity of the scenarios (e.g. discount and index curves, number of yield curve tenors) needs to match the simulation market definition above; each yield curve tenor scenario is represented as a discount factor.
- `sensitivityConfigFile`: Sensitivity parameterisation for the sensitivity analysis on asofDate.
- `historicalPeriod`: comma-separated date list, an even number of ordered dates is required (d1, d2, d3, d4, ...), where each pair (d1-d2, d3-d4, ...) defines the start and end of historical observation periods used.
- `mporDays`: Alternatively, the second date can be specified in terms of calendar days from asofDate.
- `mporCalendar`: Calendar for computing the mporDate from asofDate and mporDays.
- `outputFile`: Output file name

See the example in section 4.6.4 for a demonstration.

5.1.9 P&L, P&L Explain, ZeroToParShift, Scenario

The `pnl` and `pnlExplain` analytics provide computation of a single-period P&L and its “explanation” in terms of “raw” or “par” sensitivities. Listings 22 and 23 show configuration examples.

Listing 22: ORE analytic: P&L

```
<Setup>
  <Parameter name="asofDate">2023-01-31</Parameter>
  ...
</Setup>

<Analytics>
  <Analytic type="pnl">
    <Parameter name="active">Y</Parameter>
    <!--<Parameter name="mporDate">2023-02-14</Parameter>-->
    <Parameter name="mporDays">10</Parameter>
    <Parameter name="mporCalendar">US</Parameter>
    <Parameter name="simulationConfigFile">simulation.xml</Parameter>
    <Parameter name="curveConfigMporFile">curveconfig.xml</Parameter>
    <Parameter name="conventionsMporFile">conventions.xml</Parameter>
    <Parameter name="portfolioMporFile">mporportfolio.xml</Parameter>
    <Parameter name="outputFileName">pnl.csv</Parameter>
    <Parameter name="dateAdjustedRiskFactors">CommodityCurve</Parameter>
  </Analytic>
</Analytics>
```

The parameters in Listing 22 have the following interpretation:

- `mporDate`: The second (later) of the two valuation dates for the P&L calculation. The first date is given by the asofDate in the Setup section. Note that market data needs to be provided for both dates.
- `mporDays`: Alternatively, the second date can be specified in terms of calendar days from asofDate.
- `mporCalendar`: Calendar for computing the mporDate from asofDate and mporDays.
- `simulationFile`: Parameterisation of the simulation market which determines which market factors are evolved from asofDate to mporDate to compute the P&L.
- `curveConfigMporFile`, `conventionsMporFile`: Parametrisation to be applied at the mporDate; this may be different from the configuration on asofDate which is provided in the Setup section.
- `conventionsMporFile`: Conventions applied at the mporDate
- `portfolioMporFile` [Optional]: The portfolio on mporDate which is usually different from the portfolio on asofDate.
- `dateAdjustedRiskFactors` [Optional]: List of risk factor types (e.g., CommodityCurve) for which scenario dates at the mporDate are adjusted relative to the asofDate when computing P&L Explain. This adjustment is particularly important for commodity curves, so that scenario dates reference the same future contracts at both mporDate and asofDate.

Listing 23: ORE analytic: P&L Explain

```
<Analytics>
  <Analytic type="pnlExplain">
    <Parameter name="active">Y</Parameter>
    <Parameter name="mporDate">2023-02-14</Parameter>
    <Parameter name="simulationConfigFile">simulation.xml</Parameter>
    <Parameter name="curveConfigMporFile">curveconfig.xml</Parameter>
    <Parameter name="conventionsMporFile">conventions.xml</Parameter>
    <Parameter name="portfolioMporFile">mporportfolio.xml</Parameter>
    <Parameter name="outputFileName">pnl_explain.csv</Parameter>
    <Parameter name="dateAdjustedRiskFactors">CommodityCurve</Parameter>
    <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
    <Parameter name="parSensitivity">Y</Parameter>
    <Parameter name="riskFactorLevelReporting">Y</Parameter>
  </Analytic>
</Analytics>
```

The additional parameters in Listing 23 have the following interpretation:

- `sensitivityConfigFile`: Sensitivity parameterisation for the sensitivity analysis on asofDate
- `parSensitivity`: Boolean to specify whether par sesnitivities should be used in the P&L explanation; "raw" sensitivities will be used by default.
- `riskFactorLevelReporting`: Enables P&L explanation with decomposition by risk factor.

See the example in section 4.6.10 for a demonstration of the P&L and P&L Explain analytics.

The `pnlExplain` analytic above - when based on par sensitivities - needs market *par rate* shifts between two dates. ORE's scenario machinery operates in the "raw" domain primarily, so that a utility is useful that converts raw market moves into equivalent par market moves. The analytic in Listing 24 can be used for that purpose.

Listing 24: ORE analytic: Zero to Par Shift

```
<Analytics>
   <Analytic type="zeroToParShift">
       <Parameter name="active">Y</Parameter>
       <Parameter name="marketConfigFile">simulation.xml</Parameter>
       <Parameter name="stressConfigFile">stresstest.xml</Parameter>
       <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
       <Parameter name="pricingEnginesFile">pricingengine.xml</Parameter>
       <Parameter name="scenarioOutputFile">stresstest.csv</Parameter>
       <Parameter name="parShiftsFile">parshifts.csv</Parameter>
   </Analytic>
</Analytics>
```

See the example in section 4.6.13 for a demonstration.

The scenario analytic is a utility to export the simulation market's base scenario as a file. This is also used in the context of P&L calculations. The configuration is minimal as shown n Listing 25.

Listing 25: ORE analytic: Scenario

```
<Analytics>
    <Analytic type="scenario">
      <Parameter name="active">Y</Parameter>
      <Parameter name="simulationConfigFile">simulation.xml</Parameter>
      <Parameter name="scenarioOutputFile">scenario.csv</Parameter>
    </Analytic>
</Analytics>
```

See the example in section 4.6.12.

5.1.10 Initial Margin: ISDA SIMM and IM Schedule, CRIF Generation

The `simm` 'analytic' provides computation of initial margin using ISDA's Standard Initial Margin Model (SIMM) based on sensitivities in the Common Risk Interchange Format (CRIF) defined by ISDA. Listing 26 shows a configuration example.

Listing 26: ORE analytic: ISDA SIMM

```
  <Analytics>
    <Analytic type="simm">
      <Parameter name="active">Y</Parameter>
      <Parameter name="version">2.6.5</Parameter>
      <Parameter name="crif">crif.csv</Parameter>
      <Parameter name="calculationCurrency">USD</Parameter>
      <Parameter name="calculationCurrencyCall">USD</Parameter>
      <Parameter name="calculationCurrencyPost">USD</Parameter>
      <Parameter name="resultCurrency">USD</Parameter>
      <Parameter name="reportingCurrency">USD</Parameter>
      <Parameter name="enforceIMRegulations">Y</Parameter>
      <Parameter name="mporDays">10</Parameter>
      <Parameter name="simmCalibration">simm_calibration.xml</Parameter>
      <Parameter name="writeIntermediateReports">N</Parameter>
    </Analytic>
  </Analytics>
```

The parameters in Listing 26 have the following interpretation:

- `version`: SIMM version, ORE supports versions 1.0, 1.1, 1.2, 1.3, 1.3.38, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.5A, 2.6, 2.6.5; the latest version as of December 2024
- `crif`: CRIF file name.
  If the crif.csv input is omitted, then the ORE crif analytic (see below) is used internally to generate the CRIF input. However, CRIF generation in ORE is limited to IR/FX risks, whereas the SIMM analytic can process a full CRIF across IR/FX/INF/EQ/CR/COM risks.

- `calculationCurrency`: Determines which `Risk_FX` entries of the CRIF will be ignored in the SIMM calculation
- `calculationCurrencyCall` [Optional]: Separate calculation currency for the SIMM to call
- `calculationCurrencyPost` [Optional]: Separate calculation currency for the SIMM to post
- `resultCurrency` [Optional]: Currency of the resulting SIMM amounts in the report, by default equal to the calculation currency
- `reportingCurrency` [Optional]: Adds extra columns to the SIMM report (reporting currency and converted SIMM amount)
- `enforceIMRegulations`: If true, SIMM is calculated per post/collect regulation (passed for each record in the CRIF), and finally the worst case SIMM is reported; the flag is set to false by default i.e. post and collect regulations in the CRIF file are ignored.
- `mporDays`: 1 or 10; ORE supports both choices for versions from 2.2 onwards, only 10 is supported for earlier versions.
- `simmCalibration` [Optional]: SIMM model calibration (in a nutshell: risk weights and correlations) passed as a file; if provided, it overrides the version code above

See the example in section 4.7.

The `crif` analytic generates a `crif.csv`, input for the ISDA SIMM calculation. CRIF generation in ORE is limited to IR/FX risks. Listing 27 shows a configuration example.

Listing 27: ORE analytic: CRIF Generation

```
<Analytics>
  <Analytic type="crif">
    <Parameter name="active">N</Parameter>
    <Parameter name="marketConfigFile">crifmarket.xml</Parameter>
    <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
    <Parameter name="baseCurrency">EUR</Parameter>
    <Parameter name="simmVersion">2.7</Parameter>
    <Parameter name="crifOutputFile">crif.csv</Parameter>
  </Analytic>
</Analytics>
```

CRIF generation is based on par sensitivity analysis with a SIMM specific sensitivity configuration and some subsequent labeleing of the sensitivities so that the resulting output file satisfies the ISDA CRIF format. The resulting crif.csv can be fed into the SIMM analytic in 26.

The `imschedule` analytic computes the basic Initial Margin calculation using the “IM Schedule” method based on minimal trade information (NPV, notional, end date) provided in the CRIF file. Listing 28 shows the analytic parameterisation.

Listing 28: ORE analytic: IM Schedule

```
  <Analytics>
    <Analytic type="imschedule">
      <Parameter name="active">Y</Parameter>
      <Parameter name="crif">crif_schedule.csv</Parameter>
      <Parameter name="calculationCurrency">USD</Parameter>
    </Analytic>
  </Analytics>
```

The specific CRIF file for that case is expected to provide two lines per trade, one with RiskClass = PV and one with RiskClass = Notional, so that the amounts in these CRIF lines are interpeted as NPV respectively notional. Further required columns are product class and end date.

Note that the product class has to be in

- Rates
- FX
- Equity
- Credit
- Commodity

in contrast to SIMM where we use the combined RatesFX product class.

This analytic is also demonstrated and discussed in section 4.7.

5.1.11 XVA Stress Testing

The `XVA stress` and `XVA sensitivity` 'analytics' provide computation of bump and revalue XVA sensitivities and XVA changes under user defined stress scenarios. Listing 29 shows a typical configuration for XVA stress calculation.

Listing 29: ORE analytic: XVA stress

```
<Analytics>
 <Analytic type="xvaStress">
    <Parameter name="active">Y</Parameter>
    <Parameter name="marketConfigFile">simulation.xml</Parameter>
    <Parameter name="stressConfigFile">stresstest.xml</Parameter>
    <Parameter name="sensitivityConfigFile">sensitivity_stress.xml</Parameter>
    <Parameter name="writeCubes">N</Parameter>
  </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `marketConfigFile`: Configuration file defining the simulation market under which sensitivities are computed, see 5.3. Only a subset of the specification is needed (the one given under `Market`, see 5.3.3 for a detailed description).

- `stressConfigFile`: Stress Scenario definition, see section 5.5
- `sensitivityConfigFile`: Configuration file for the sensitivity calculation, see section 5.4.
- `writeCubes`: Boolean flag, if true ORE outputs the raw and net cube under each scenario, defaults to false.

Stress Tests can be used to compute stressed value adjustments. The stress tests for the XVA stress test analytic are defined in the regular NPV stress test format (see 5.5).

The XVA stress analytic builds a stress scenario generator and a scenario simulation market. The simulation market replaces the todaysMarket in the XVA analytic to compute the value adjustments under a stress scenario.

For performance reasons it is recommended to use AMC simulation if possible.

For some risk factors the simulation market behaves a different to the todays market. For example it could use different tenor structure for the curves or it uses only the swaption ATM vols if the volatilities aren't simulated. Therefore it is recommended to activate
t UseSpreadedTermStructures in the stress tests scenario parameerisation and activate the swaption volatility simulation for the stress test run.

There is no dedicated parametrisations for the xva and exposure settings for the stress test, ORE reuses the existing ones for the regular exposure and xva analytics. But the xva and exposure analytic themselves can be deactivated.

For an example see 4.12.1.

5.1.12 XVA Sensitivity

The XVA sensitivity analytics configuration is similar to the one of the XVA stress calculation. Listing 30 shows an example.

Listing 30: ORE analytic: XVA Sensitivity

```
<Analytics>
    <Analytic type="xvaSensitivity">
      <Parameter name="active">Y</Parameter>
      <Parameter name="marketConfigFile">simulation.xml</Parameter>
      <Parameter name="sensitivityConfigFile">sensitivity.xml</Parameter>
      <Parameter name="parSensitivity">Y</Parameter>
    </Analytic>
</Analytics>
```

For both analytics ORE reuses the parameterisations of the exposure and XVA analtyic. One needs to setup those analytics as usual but one can deactivate them. See listings 13 and 7. If set to active, ORE will run the regular XVA analytic and the XVA sensitivity or XVA stress analytic, respectively.

Similar to the XVA Stress Analytic ORE can compute bump and revaluation sensitivities for value adjustments. The XVA Sensitivitiy Analytic uses the same sensitivity scenario input as the regular sensitivity analytic (see 5.4).

ORE computes the XVA and exposure measures under each sensitivity scenario. If the parSensitivity flag is set to true, an additional set of par sensitivity outputs is generated.

The XVA Sensitivity Analytic replaces the todaysMarket in the exposure simulation with a ScenarioSimMarket. For some risk factors the simulation market behaves different to the todays market, e.g. uses a different tenor structure for building the curves or uses only the swaption ATM vols if the volatilities aren't simulated. To minimize some of the effects, it is recommended to activate
t UseSpreadedTermStructures in the stress test scenarios and activate the swaption volatility simulation for the stress test run (see also [2]).

As in the XVA Stress analytic, there is no dedicated parametrisation for the xva and exposure settings for the XVA Sensitivity Analytic, ORE reuses the existing ones for the regular exposure and xva analytics.

For an example see 4.12.3.

5.1.13 XVA Explain

The `XVA explain` 'analytic' provide the computation of the market implied changes of the value adjustments between to evaluation dates $XVA(t_0, marketdata(t_1)) - XVA(t_0, marketdata(t_0))$. Listing 31 shows a typical configuration for XVA explain calculation.

Listing 31: ORE analytic: XVA Explain

```
<Analytics>
 <Analytic type="xvaExplain">
      <Parameter name="active">Y</Parameter>
      <Parameter name="marketConfigFile">simulation.xml</Parameter>
      <Parameter name="stressConfigFile">stresstest.xml</Parameter>
      <Parameter name="sensitivityConfigFile">sensitivity_stress.xml</Parameter>
      <Parameter name="writeCubes">N</Parameter>
      <Parameter name="shiftThreshold">1e-4</Parameter>
      <Parameter name="mporDays">1</Parameter>
      <Parameter name="mporCalendar">EUR</Parameter>
    </Analytic>
</Analytics>
```

The parameters have the following interpretation:

- `marketConfigFile`: Configuration file defining the simulation market under which sensitivities are computed, see 5.3. Only a subset of the specification is needed (the one given under `Market`, see 5.3.3 for a detailed description).
- `stressConfigFile`: Stress Scenario definition, see section 5.5
- `sensitivityConfigFile`: Configuration file for the sensitivity calculation, see section 5.4.
- `writeCubes`: Boolean flag, if true ORE outputs the raw and net cube under each scenario, defaults to false.

	RiskFacto	rTradeId	NettingSetId	CVA_Base	CVA	Change
0	ALL			91876.7431	81990.9740	-9885.7691
1	ALL	Swap_20y	Swap_20y	159509.2148	150278.8754	-9230.3394
2	ALL	Swap_20y_USD	Swap_20y_USD	42065.8853	45831.0601	3765.1748
3	IndexCurve/EUR-EURIBOR-6M/19			91876.7431	92953.7582	1077.0151
4	IndexCurve/EUR-EURIBOR-6M/19	Swap_20y	Swap_20y	159509.2148	161063.8061	1554.5913

Table 16: CVA Explain results

- `shiftThreshold`: Par Rate shifts below this threshold are ignored.
- `mporDays`: Derives the 2nd evaluation date t_1 from $t_0 + mporDates$.
- `mporCalendar`: Calendar used to ensure that t_1 is a valid business day.

ORE reuses the parameterisations of the exposure and XVA analtyic. One needs to setup those analytics as usual but one can deactivate them. See listings 13 and 7. If set to active, ORE will run the regular XVA analytic and the XVA sensitivity or XVA stress analytic respectively.

ORE can compute the market implied XVA change between two evaluation dates. For each risk factor defined in the sensitivity config ORE computes the par rate change between t0 and t0 + mporDays. ORE derives for each risk factor a shift scenario $(ParRate(t_1) - ParRate(t_0))$ and computes the CVA change implied by those risk factors shifts at t0.

The output is a csv file with the all the value adjustments under each scenario similar the regular XVA outputs, added with an extra column for the scenario name (shifted risk factor). Further ORE generate the CVA explain output, which contains the name of the shifted risk factor the base and scenario CVA value and the change between base and scenario CVA value (see 16).

The XVA Explain Analytic replaces the todaysMarket in the exposure simulation with a ScenarioSimMarket. For some risk factors the simulation market behaves a different to the todays market, e.g. uses different tenor structure for building the curves or uses only the swaption ATM vols if the volatilities aren't simulated. To minimize some of the effects, it is recommended to activate
t UseSpreadedTermStructures in the stress tests scenarios and activate the swaption volatility simulation for the stress test run (see also [2]).

For an example see 4.12.2.

5.1.14 CCR Capital: SA-CCR

The `saccr` 'analytic' provides computation of SA-CCR capital calculation, see 32. The required input in addition to the setup section is a `csaFile` (as in XVA analytics) and a `collateralBalancesFile` which provides variation and initial margin balances as of today.

Listing 32: ORE analytic: SA-CCR

```
<Analytics>
    <Analytic type="saccr">
      <Parameter name="active">Y</Parameter>
      <Parameter name="csaFile">netting.xml</Parameter>
      <Parameter name="collateralBalancesFile">collateralbalances.xml</Parameter>
    </Analytic>
  </Analytic>
</Analytics>
```

For an example see 4.13.1.

5.1.15 CVA Capital: SA-CVA and BA-CVA

The `sacva` 'analytic' provides computation of SA-CVA capital calculation based on CVA sensitivities. These can be computed on-the-fly or can be externally provided, see listing 33. In the former case, the `xvaSensitivity` analytic needs to be specified with all reqired inputs as shown in Listing 30 above, likewise the `simulation` and `xva` analytic which are utilised under the hood.

Listing 33: ORE analytic: SA-CVA

```
<Analytics>
 <Analytic type="sacva">
    <Parameter name="active">Y</Parameter>
      <!-- If none of the following is provided, ORE builds on-the-fly sensis using
           the configuration above -->
      <!-- Load net sensitivities that can be passed into the SA-CVA calculator directly -->
      <!--<Parameter name="saCvaNetSensitivitiesFile">sacva_sensitivity.csv</Parameter>-->
      <!-- Load CVA sensitivities, output of the xva sensi analytic, that needs mapping
           and aggregation -->
      <!--<Parameter name="cvaSensitivitiesFile">xva_par_sensitivity_cva.csv</Parameter>-->
  </Analytic>
</Analytics>
```

The `bacva` 'analytic' provides computation of BA-CVA capital calculation, see 34. The required input in addition to the setup section is a `csaFile` (as in XVA analytics) and a `collateralBalancesFile` which provides variation and initial margin balances as of today, the same as in the `saccr` analytic which is used under the hood to compute exposures at default.

Listing 34: ORE analytic: BA-CVA

```
<Analytics>
    <Analytic type="bacva">
      <Parameter name="active">Y</Parameter>
      <Parameter name="csaFile">netting.xml</Parameter>
      <Parameter name="collateralBalancesFile">collateralbalances.xml</Parameter>
    </Analytic>
  </Analytic>
</Analytics>
```

The smrc 'analytic' provides computation of the basic SMRC market risk captial calculation, without additional inputs, see 35.

Listing 35: ORE analytic: SMRC

```
<Analytics>
    <Analytic type="smrc">
      <Parameter name="active">Y</Parameter>
    </Analytic>
  </Analytic>
</Analytics>
```

For an example see 4.12.4.

5.2 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 5.1.

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.

5.2.1 Discounting Curves

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

Listing 36: 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 5.7 below. In this case ORE is instructed to build an Eonia Swap curve made of Overnight Deposit and Eonia Swap quotes. The right most token of the string Yield/EUR/EUR1D (EUR1D) is user defined, the first two tokens Yield/EUR have to be used to point to a yield curve in currency EUR.

5.2.2 Index Curves

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

Listing 37: 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/EUR12M</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).

5.2.3 Yield Curves

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

Listing 38: 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

5.2.4 Swap Index Curves

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

Listing 39: Swap index curve block with ID 'default'

```xml
<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.

5.2.5 FX Spot

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

Listing 40: FX spot block with ID 'default'

```xml
<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.

5.2.6 FX Volatilities

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

Listing 41: 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.

5.2.7 Swaption Volatilities

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

Listing 42: 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.

5.2.8 Cap/Floor Volatilities

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

Listing 43: 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.

5.2.9 Default Curves

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

Listing 44: 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.

5.2.10 Securities

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

Listing 45: 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.

5.2.11 Equity Curves

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

Listing 46: 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.

5.2.12 Equity Volatilities

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

Listing 47: 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.

5.2.13 Inflation Index Curves

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

Listing 48: 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 49: 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.

5.2.14 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 50: Inflation Cap/Floor Volatility Surfaces block with ID 'default'

```
<YYInflationCapFloorVolatilities id="default">
  <YYInflationCapFloorVolatility 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 index name. The last token (e.g. EUHICPXT_ZC_CF) is user defined and will be found in the curve configuration's CurveId tag.

5.2.15 CDS Volatility Structures

CDS volatility structures are configured as follows

Listing 51: 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.

5.2.16 Base Correlation Structures

Base correlation structures are configured as follows

Listing 52: 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.

5.2.17 Correlation Structures

Correlation structures are configured as follows

Listing 53: 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.

5.2.18 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 54: 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 5.1) 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.

5.3 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 55: Simulation configuration

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

Each of the three blocks is sketched in the following.

5.3.1 Parameters

Let us discuss this section using the following example

Listing 56: Simulation configuration

```
<Parameters>
  <Grid>80,3M</Grid>
  <TimeStepsPerYear>24</TimeStepsPerYear>
  <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>
```

- `Grid`: Specifies the simulation time grid, here 80 quarterly steps.[5]
- `TimeStepsPerYear`: Minimum number of time steps per year to be used to evolve the stochastic process. This is useful if Euler discretization with a coarse Grid is used. Optional, defaults to null which means only the points specified under Grid are used to evolve the stochastic process.
- `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. item `Sequence`: Choose random sequence generator (*MersenneTwister, MersenneTwisterAntithetic, Sobol, Burley2020Sobol, SobolBrownianBridge, Burley2020SobolBrownianBridge*).

[5]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`.

- `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* or *Burley2020SobolBrownianBridge* is used, ordering of variates (*Factors, Steps, Diagonal*)
- `DirectionIntegers:` If the sequence type *SobolBrownianBridge*, *Burley2020SobolBrownianBridge*, *Sobol* or *Burley2020Sobol* 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 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.

5.3.2 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,
- n_{cr} credit models,
- n_{com} commodity 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.

When defining the currencies in the cross asset model, the domestic currency always has to be given as the first currency.

Listing 57: Simulation model currencies configuration

```
<CrossAssetModel>
  <DomesticCcy>EUR</DomesticCcy>
  <Currencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
    <Currency>GBP</Currency>
    <Currency>CHF</Currency>
    <Currency>JPY</Currency>
  </Currencies>
  <Equities>
        <!-- ... -->
  </Equities>
  <InflationIndices>
        <!-- ... -->
  </InflationIndices>
  <CreditNames>
        <!-- ... -->
  </CreditNames>
  <Commodities>
        <!-- ... -->
  </Commodities>
  <BootstrapTolerance>0.0001</BootstrapTolerance>
  <Measure>LGM</Measure><!-- Choices: LGM, BA -->
  <Discretization>Exact</Discretization>
  <SalvagingAlgorithm>Spectral</SalvagingAlgorithm>
  <!-- ... -->
</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.

The Discretization tag chooses between time discretization schemes for the risk factor evolution. *Exact* means exploiting the analytical 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 or if more sophisticated component models are used). If *Euler* is used, you should consider setting TimeStepsPerYear (see above) to a large enough value.

The SalvagingAlgorithm tag specifies the preprocessing of the input correlation matrix (None, Spectral, Hypersphere, LowerDiagonal, Higham). Optional, defaults to None.

Each interest rate model is specified by a block as follows

Listing 58: 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>
      <FloatSpreadMapping>proRata</FloatSpreadMapping>
    </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 (for CalibrationType *BestFit* this should be one more entry than the `Volatility/TimeGrid` entries, for *Bootstrap* this is ignored), 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. There have to be at least one more calibration options configured than `Volatility/TimeGrid` entries were given.
- `ParameterTransformation:` LGM model prices are invariant under scaling and shift transformations [17] 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.
- `FloatSpreadMapping:` mapping of float spreads in analytic swaption pricing for model calibration: proRata, nextCoupon, simple, optional, defaults to proRata.

The reason for having to specify one more `Volatility/InitialValue` entries than `Volatility/TimeGrid` entries (and at least one more calibration option than `Volatility/TimeGrid` entries) is the fact that the intervals defined by the `Volatility/TimeGrid` entries are spanning from $[0, t_1], [t_1, t_2] \ldots [t_n, \infty]$, which results in $n + 1$ intervals.

Each FX model is specified by a block as follows

Listing 59: 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 (for CalibrationType *BestFit* this should be one more entry than the `Sigma/TimeGrid` entries, for *Bootstrap* this is ignored), 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). There have to be at least one more calibration options configured than `Sigma/TimeGrid` entries were given

Each equity model is specified by a block as follows

Listing 60: 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 (for CalibrationType *BestFit* this should be one more entry than the `Sigma/TimeGrid` entries, for *Bootstrap* this is ignored), 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). There have to be at least one more calibration options configured than `Sigma/TimeGrid` entries were given

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 61. 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 61: 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 61 could be replaced with the `CalibrationBaskets` node in 62.

Listing 62: 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 63. 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 63: 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 63. A `YoYCapFloor` is also allowed and it has the structure shown in Listing 64. 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 64: Layout for `YoYCapFloor` calibration instrument.

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

For commodity simulation we currently provide one model, as described in the methodology appendix. Commodity model components are specified by commodity name, by a block as follows

Listing 65: Simulation model commodity configuration

```
<CrossAssetModel>
  <!-- ... -->
  <CommodityModels>
    <CommoditySchwartz name="default">
      <Currency>EUR</Currency>
      <CalibrationType>None</CalibrationType>
      <Sigma>
        <Calibrate>Y</Calibrate>
        <InitialValue>0.1</InitialValue>
      </Sigma>
      <Kappa>
        <Calibrate>Y</Calibrate>
        <InitialValue>0.1</InitialValue>
      </Kappa>
      <CalibrationOptions>
          ...
      </CalibrationOptions>
      <DriftFreeState>false</DriftFreeState>
    </CommoditySchwartz>
    <CommoditySchwartz name="WTI">
      <!-- ... -->
    </CommoditySchwartz>
    <CommoditySchwartz name="NG">
      <!-- ... -->
    </CommoditySchwartz>
      <!-- ... -->
  </CommodityModels>
  <!-- ... -->
<CrossAssetModel>
```

CommoditySchwartz sections are defined by commodity name, but we also support a default configuration as above for the IR and FX model parameterisations. Each component is parameterised in terms of two constant, non time-dependent parameters σ and κ so far (see appendix). Within each CommoditySchwartz section, the interpretation of elements is as follows:

- `Currency`: Currency of denomination

- `CalibrationType`: Choose between *BestFit*, *Bootstrap*, *FirstBestFitThanBootstrap* and *None*. The choice *None* will deactivate calibration as usual. *BestFit* will attempt to set the model parameter(s) such that the error in matching calibration instrument prices is minimised. The option *Bootstrap* will iteratively attempt to calibrate the time dependent seasonality factor to the calibration instrument prices. The option *FirstBestFitThanBootstrap* will first attempt to fit the model parameters (kappa and sigma) as in *BestFit* option. Then, time dependent seasonality parameter is bootstrapped as in *Bootstrap* option.
- `Sigma/Calibrate`: Flag to enable/disable calibration of this particular parameter
- `Sigma/InitialValue`: Initial value of the constant parameter
- `Kappa/Calibrate`: Flag to enable/disable calibration of this particular parameter
- `Kappa/InitialValue`: Initial value of the constant parameter
- `Seasonality/Calibrate`: Flag to enable/disable calibration of this particular parameter
- `Seasonality/ParamType`: Initial time grid for this parameter, can be left empty if ParamType is Constant
- `Seasonality/TimeGrid`: Initial time grid for this parameter. If ParamType is Constant, there should be a single entry
- `Seasonality/InitialValue`: Vector of initial values. Size of the vector should be identical to the size of TimeGrid vector. For *Bootstrap* this is ignored. If ParamType is Constant, there should be a single entry
- `CalibrationOptions`: Choice of calibration instruments by expiry and strike, strikes can be empty (implying the default, ATMF options), or explicitly specified (in terms of commodity prices as absolute strike values).
- `DriftFreeState[Optional]`: Boolean to switch between the two implementations of the state variable, see appendix. By default this is set to `false`.

Finally, the instantaneous correlation structure is specified as follows.

Listing 66: Simulation model correlation configuration

```xml
<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. Note that the commodity components can have non-zero correlations among each other, but correlations to all other CAM components must remain set to zero for the time being.

5.3.3 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.

The specified currencies need to contain at least the currencies both defined above in the cross asset model and used in the portfolio, the order however is not important.

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.

Commodity volatilities are taken as ATM slice by default. To simulate commodity volatility surface with smile a set of `<Moneyness>` must be supplied. There are two methods in ORE for commodity 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. 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 and no surface node is given.
- Simulating the full volatility surface. The node `<SimulateATMOnly>` should be omitted or set to false, and explicit moneyness levels 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 risk factor using the node `<DayCounter name="EXAMPLE_CURVE">`. If no day counter is specified for a given risk factor then the default Actual365 is used. To specify a new default for a risk factor 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.
- Extrapolation: This can be FlatFwd or FlatZero. If not given, the value defaults to FlatFwd.

For Default Curve, there is a similar choice for the extrapolation:

- Extrapolation: This can be FlatFwd or FlatZero. If not given, the value defaults to FlatFwd.

For swaption, yield, interest rate cap-floor, yoy inflation cap-floor, zc inflation cap-floor, cds, fx, equity, commodity volatilities the smile dynamics can be specified as shown in listing 67 for swaption vols. The empty key serves as a default configuration for all keys for which no own smile dynamics node is present. The allowed smile dynamics values are StickyStrike, StickyMoneyness and StickySABR. If not given, the smile dynamics defaults to StickyStrike.

Note that StickySABR is only available for swaption volatilities, yield volatilities and interest rate cap-floor volatilities, and can only be used if the corresponding T0 surface has been calibrated using a SABR model. The SABR volatility surface in the simulation market is recalibrated to the expiry/term/strike grid specified in the simulation configuration, using the initial SABR configuration from the T0 surface. Any subsequent recalibration will keep all SABR parameters except for α fixed.

Listing 67: Smile Configuration Node

```
    <SmileDynamics key="">StickyStrike</SmileDynamics>
    <SmileDynamics key="EUR-ESTER">StickyMoneyness</SmileDynamics>
    <SmileDynamics key="USD-SOFR-3M">StickySABR</SmileDynamics>
```

Listing 68: CurveAlgebra Node

```
    <CurveAlgebra>
      <Curve>
        <Key>DiscountCurve/EUR</Key>
        <Operation>
          <Type>Spreaded</Type>
          <Arguments>
            <Argument>IndexCurve/EUR-EURIBOR-1M,1</Argument>
            <Argument>IndexCurve/EUR-EURIBOR-3M,-2</Argument>
          </Arguments>
        </Operation>
      </Curve>
    </CurveAlgebra>
```

We can specify a CurveAlgebra node as a subnode of Market to link curves in the scenario sim market to other curves using predefined operations. Listing 68 shows an example for a spread declaration, where the instantaneous forward rate of the EUR discount curve is linked to both EURIBOR-1M and 3M index curves with multipliers 1 and -2, respectively.

```
<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>
    <Extrapolation>FlatFwd</Extrapolation>
</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>
  <SimulateATMOnly>false</SimulateATMOnly>
  <StrikeSpreads>-0.02,-0.01,0.0,0.01,0.02</StrikeSpreads>
  <!-- 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>
      <Moneyness>0.1,0.5,1.0,1.5,2.0,3.0</Moneyness>
    </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>
  <AggregationScenarioDataCurrencies>
    <Currency>EUR</Currency>
    <Currency>USD</Currency>
  </AggregationScenarioDataCurrencies>
  <AggregationScenarioDataIndices>
    <Index>EUR-EURIBOR-3M</Index>
    <Index>EUR-EONIA</Index>
    <Index>USD-LIBOR-3M</Index>
  </AggregationScenarioDataIndices>
</Market>
```

Listing 69: Simulation market configuration

5.4 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 72, 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.
- `ShiftScheme:` The finite difference scheme to use (`Forward`, `Backward`, `Central`), if not given, this parameter defaults to `Forward`
- `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.
- `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

- `ShiftTerms`: For swaption volatility surfaces, the underlying terms to apply shifts to, given as a comma separated list of periods.
- `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 ShiftType, ShiftSize, ShiftScheme nodes take an optional attribute key that allows to configure different values for different sensitivity templates. The sensitivity templates are defined in the pricing engine configuration. This is best explained by an example: In Example 15 the product type BermudanSwaption has a sensitivity template `IR_FD` attached, see 70. This can be used to specify different shifts for trades that were built against this engine configuration, see 71: For Bermudan swaptions a larger shift size of 10bp and a central difference scheme is used to compute discount curve sensitivities in EUR. Since no separate shift type is specified, the default shift type `Absolute` is used. Note regarding the reports:

- the sensi scenario report contains scenario NPVs related to the possibly product specific configured shift sizes
- the sensi report contains renormalized sensitivities, i.e. sensitivities are always expressed w.r.t. the default shift sizes
- the sensi config report only contains the default configuration

```xml
<Product type="BermudanSwaption">
  <Model>LGM</Model>
  <ModelParameters>
    ...
  </ModelParameters>
  <Engine>Grid</Engine>
  <EngineParameters>
    ...
    <Parameter name="SensitivityTemplate">IR_FD</Parameter>
  </EngineParameters>
</Product>
```

Listing 70: Sensitivity template definition

```xml
  <DiscountCurve ccy="EUR">
    <ShiftType>Absolute</ShiftType>
    <ShiftSize>0.0001</ShiftSize>
    <ShiftScheme>Forward</ShiftScheme>
    <ShiftSize key="IR_FD">0.001</ShiftSize>
    <ShiftScheme key="IR_FD">Central</ShiftScheme>
    <ShiftTenors>6M,1Y,2Y,3Y,5Y,7Y,10Y,15Y,20Y</ShiftTenors>
  </DiscountCurve>
```

Listing 71: Sensitivity template definition

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>
      <ShiftStrikes/>
      <ShiftTerms>1Y,5Y,10Y</ShiftTerms>
    </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 72: Sensitivity configuration

Par Sensitivity Analysis

To perform a par sensitivity analysis, additional sensitivity configuration is required that describes the assumed par instruments and related conventions. This additional data is required for:

- DiscountCurves

- IndexCurves
- CapFloorVolatilities
- CreditCurves
- ZeroInflationIndexCurves
- YYInflationIndexCurves
- YYCapFloorVolatilities

By default, par conversion is applied to all risk factor types listed above. However, it is possible to exclude selected risk factor types by adding an optional block to the sensitivity configuration as shown in listing 73. Uncomment the risk factor type(s) here that should be excluded from par conversion.

```
<SensitivityAnalysis>

  <ParConversionExcludes>
    <!--<Type>DiscountCurve</Type>-->
    <!--<Type>YieldCurve</Type>-->
    <!--<Type>IndexCurve</Type>-->
    <!--<Type>OptionletVolatility</Type>-->
    <!--<Type>SurvivalProbability</Type>-->
    <!--<Type>ZeroInflationCurve</Type>-->
    <!--<Type>YearOnYearInflationCurve</Type>-->
    <!--<Type>YoYInflationCapFloorVolatility</Type>-->
  </ParConversionExcludes>

  ...

</SensitivityAnalysis>
```

Listing 73: Par conversion excludes

By default, par sensitivity excludes index historical fixings on valuation date when calculating the sensitivities of par instruments to zero rates. However, it is possible to exclude selected indexes by adding an optional block to the sensitivity configuration as shown in listing 74. It handles a regex.

```
<SensitivityAnalysis>

  <ParSensiRemoveFixing>.*</ParSensiRemoveFixing>

</SensitivityAnalysis>
```

Listing 74: Par Sensitivity excludes fixings

By default, par sensitivity analysis sets small diagonal elements in the par conversion matrix to 0.01 silently to avoid matrix inversion issues. However, it is possible to control this behaviour by adding an optional block to the sensitivity configuration as shown in listing 75. The available options are:

- `Silent`: Set small diagonal elements to 0.01 without warnings (default behaviour)
- `Warning`: Set small diagonal elements to 0.01 but issue structured warnings

- `Disable`: Disable regularisation, use original diagonal elements as-is

```
<SensitivityAnalysis>

  <ParConversionMatrixRegularisation>Warning</ParConversionMatrixRegularisation>

</SensitivityAnalysis>
```

Listing 75: Par conversion matrix regularisation

Using DiscountCurves as an example, the full sensitivity specification including par conversion data is as follows:

```
    <DiscountCurve ccy="EUR">
      <ShiftType>Absolute</ShiftType>
      <ShiftSize>0.0001</ShiftSize>
      <ShiftTenors>2W,1M,3M,6M,9M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y</ShiftTenors>
      <ParConversion>
        <!--DEP, FRA, IRS, OIS, FXF, XBS -->
        <Instruments>OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS,OIS</Instruments>
        <SingleCurve>true</SingleCurve>
  <Conventions>
          <Convention id="DEP">EUR-EURIBOR-CONVENTIONS</Convention>
          <Convention id="IRS">EUR-6M-SWAP-CONVENTIONS</Convention>
          <Convention id="OIS">EUR-OIS-CONVENTIONS</Convention>
        </Conventions>
      </ParConversion>
    </DiscountCurve>
```

Listing 76: Par sensitivity configuration

Using CapFloors as an example:

```
                <CapFloorVolatility key="USD-SOFR">
                        <ShiftType>Absolute</ShiftType>
                        <ShiftSize>0.0001</ShiftSize>
                        <ShiftScheme>Forward</ShiftScheme>
                        <ShiftExpiries>1Y,2Y,3Y,5Y,10Y,15Y,20Y,30Y</ShiftExpiries>
                        <ShiftStrikes>-0.0075,-0.005,-0.0025,-0.0015,0.0,0.0025,0.005,0.0075,0.01,0.
                        <Index>USD-SOFR</Index>
                        <IsRelative>false</IsRelative>
                        <ParConversion>
                                <Instruments>CapFloor,CapFloor,CapFloor,CapFloor,CapFloor,CapFloor,C
                                <DiscountCurve>USD-SOFR</DiscountCurve>
                                <RateComputationPeriod>3M</RateComputationPeriod>
                        </ParConversion>
                </CapFloorVolatility>
```

Listing 77: Par sensitivity configuration

ParConversion Fields:

- **Instruments** a comma separated list of par instrument types, see below for the possible values. The 3-letter instrument code can be extended by an arbitrary suffix to reference different conventions. For example, you can use FRA1, FRA2 in the instrument list to build FRA instruments with different convention ids FRA1, FRA2.

- **DiscountCurve** *optional*: discount curve used for pricing the par instrument.
- **RateComputationPeriod** *optional*: required for OIS CapFloors, sepcify the period of the optionlet.

Note

- The list of shift tenors needs to match the list of tenors matches the corresponding grid in the simulation (market) configuration
- The length of list of (par) instruments needs to match the length of the list of shift tenors
- Permissible codes for the assumed par instruments:
  - DEP, FRA, IRS, OIS, TBS, FXF, XBS in the case of DiscountCurves
  - DEP, FRA, IRS, OIS, TBS in the case of IndexCurves
  - DEP, FRA, IRS, OIS, TBS, XBS in the case of YieldCurves
  - ZIS, YYS for YYInflationIndexCurves, interpreted as Year-on-Year Inflation Swaps linked to Zero Inflation resp. YoY Inflation curves
  - ZIS, YYS for YYCapFloorVolatilities, interpreted as Year-on-Year Inflation Cap Floor linked to Zero Inflation resp. YoY Inflation curves
  - Any code for CreditCurves, interpreted as CDS
  - Any code for ZeroInflationIndexCurves, interpreted as CPI Swaps linked to Zero Inflation curves
  - Any code for CapFloorVolatilities, interpreted as flat Cap/Floor
- One convention needs to be referenced for each of the instrument codes

Listing 78: Stress configuration with explicit shifts

```
  <FxVolatilities>
      <FxVolatility ccypair="USDEUR">
        <ShiftType>Absolute</ShiftType>
        <Shifts>0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.3</Shifts>
        <ShiftExpiries>6M, 1Y, 2Y, 4Y, 5Y, 7Y, 10Y</ShiftExpiries>
      </FxVolatility>
    </FxVolatilities>
```

5.5 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 5.4.

This file `stressconfig.xml` specifies how stress tests can be configured. The general structure is shown in listing 81.

In this example, two zero stress scenarios “parallel_rates” and “twist” and one par rate “par_parallel” 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 sensitivitya configuration, see 5.4. Components that should not be shifted, can just be omitted in the definition of the scenario. Shifts for rate curves, credit curves and interest rate cap/floors can be given as par or zero rate shifts. By default shifts are zero rate shifts. If shifts are marked as par rate shifts all components (rate/credit/caps) shifts are par shifts in that category, for example it is not possible to have par rate first for one yield curve and zero rate for another curve in the same scenario. In case of par stress scenario, the shifted par instruments and related conventions are defined in a sensitivity configuration. The number number stress shifts (tenors/expiries and strikes) need to be align with the tenors/expiries and strikes of par instruments 5.4.

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.

For FX volatility shifts, the configuration can be specified either explicitly or using a weighting schema. In the explicit approach, exact shift values for each expiry and tenor can be provided, as demonstrated in listing 78. If only one tenor and shift value are provided, a parallel shift will be applied across all other tenors.

Two weighting schemas are supported for FX volatility shifts:

- `Unadjusted`: The shift is applied solely to the specified tenor, leaving all other tenors unchanged.
- `Weighted`: The given shift s_i is applied to the specified tenor i, while shifts for other tenors are determined using the formula $s_j = s_i \frac{w_j}{w_i}$, where w_j and w_i represent the weights for tenors j and i, respectively.

Listing 79: Stress configuration with weighted shifts

```
  <FxVolatilities>
      <FxVolatility ccypair="USDEUR">
        <ShiftType>Absolute</ShiftType>
        <WeightedShifts>
          <WeightingSchema>Weighted</WeightingSchema>
          <Shift>0.1</Shift>
          <Tenor>1Y</Tenor>
          <ShiftWeights>0.5,1.0,2.0,3.0,4.0,5.0,6.0</ShiftWeights>
          <WeightTenors>6M,1Y,2Y,4Y,5Y,7Y,10Y</WeightTenors>
        </WeightedShifts>
      </FxVolatility>
```

Listing 80: Stress configuration with unadjusted weighting schema

```
  <FxVolatilities>
      <FxVolatility ccypair="USDEUR">
        <ShiftType>Absolute</ShiftType>
        <WeightedShifts>
          <WeightingSchema>Unadjusted</WeightingSchema>
          <Shift>0.1</Shift>
          <Tenor>1Y</Tenor>
        </WeightedShifts>
      </FxVolatility>
    </FxVolatilities>
```

For commodity volatility currently only single shift per expiry is supported, which implies same shift size for all moneyness levels provided.

UseSpreadedTermStructures: If set to true, spreaded termstructures over t0 will be used for the scenario calculation, to improve the alignment of the scenario sim market and t0 curves.

```
<StressTesting>
  <UseSpreadedTermStructures>false</UseSpreadedTermStructures>
  <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>
    <CommodityCurves>
      <CommodityCurve commodity="NYMEX:CL">
        <Currency>USD</Currency>
        <ShiftType>Relative</ShiftType>
        <Shifts>0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.
        <ShiftTenors>1D,1W,2W,1M,2M,3M,4M,5M,6M,7M,8M,9M,10M,11M,12M,13M,14M,15M,16M,17M,18M,19M,20M,21M,22M,23M,24M</Shift
      </CommodityCurve>
    </CommodityCurves>
    <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>
    <CommodityVolatilities>
      <CommodityVolatility commodity="NYMEX:CL">
        <ShiftType>Relative</ShiftType>
        <Shifts>0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01</Shifts>
        <ShiftExpiries>2W, 1M, 3M, 6M, 9M, 1Y, 2Y, 3Y, 4Y, 5Y, 6Y, 7Y, 8Y, 9Y, 10Y</ShiftExpiries>
        <ShiftMoneyness>1.0</ShiftMoneyness>
      </CommodityVolatility>
    </CommodityVolatilities>
  </StressTest>
  <StressTest id="twist">
    ...
  </StressTest>
  <StressTest id="par_parallel">
    <ParShifts>
      <IRCurves>true</IRCurves>
      <SurvivalProbability>true</SurvivalProbability>
      <CapFloorVolatilities>true</CapFloorVolatilities>
    </ParShifts>
    <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 />
    <FxSpots />
    <FxVolatilities />
    <SwaptionVolatilities />
    <CapFloorVolatilities>
      <CapFloorVolatility key="EUR-EURIBOR-6M">
        <ShiftType>Absolute</ShiftType>
        <ShiftExpiries>1Y, 2Y, 3Y, 4Y, 5Y, 6Y, 7Y, 8Y, 9Y</ShiftExpiries>
        <Shifts>
          <Shift tenor="1Y">0.01</Shift>
          <Shift tenor="2Y">0.01</Shift>
          <Shift tenor="3Y">0.01</Shift>
          <Shift tenor="4Y">0.01</Shift>
          <Shift tenor="5Y">0.01</Shift>
          <Shift tenor="6Y">0.01</Shift>
          <Shift tenor="7Y">0.01</Shift>
```

```
          <Shift tenor="8Y">0.01</Shift>
          <Shift tenor="9Y">0.01</Shift>
        </Shifts>
      </CapFloorVolatility>
    </CapFloorVolatilities>
    <EquitySpots />
    <EquityVolatilities />
    <SecuritySpreads />
    <RecoveryRates />
    <SurvivalProbabilities>
      <SurvivalProbability name="Underlying1">
        <ShiftType>Absolute</ShiftType>
        <Shifts>0.01, 0.01, 0.01, 0.01, 0.01</Shifts>
        <ShiftTenors>1Y, 2Y, 3Y, 5Y, 10Y</ShiftTenors>
      </SurvivalProbability>
    </SurvivalProbabilities>
  </StressTest>
</StressTesting>
```

Listing 81: Stress configuration

5.6 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 82. In this example, two additional dates had been added to the calendar "Japan", one additional holiday and one additional business day. If the user is not certain whether 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 82: 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 83: Calendar Adjustment creating a new calendar

5.7 Curves: curveconfig.xml

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

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

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

5.7.1 Yield Curves

The top level XML elements for each `YieldCurve` node are shown in Listing 84.

Listing 84: Top level yield curve node

```
<YieldCurve>
  <CurveId> </CurveId>
  <CurveDescription> </CurveDescription>
  <Currency> </Currency>
  <DiscountCurve> </DiscountCurve>
  <Segments> </Segments>
  <InterpolationVariable> </InterpolationVariable>
  <InterpolationMethod> </InterpolationMethod>
  <MixedInterpolationCutoff> </MixedInterpolationCutoff>
  <YieldCurveDayCounter> </YieldCurveDayCounter>
  <Tolerance> </Tolerance>
  <Extrapolation> </Extrapolation>
  <ExcludeT0FromInterpolation> </ExcludeT0FromInterpolation>
  <BootstrapConfig>
    ...
  </BootstrapConfig>
</YieldCurve>
```

The meaning of each of the top level elements in Listing 84 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 17. 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 18. If the element is omitted or left blank, then it defaults to *LogLinear*.
- MixedInterpolationCutoff [Optional]: If a mixed interpolation method is used, the number of segments to which the first interpolation method is applied. Defaults to 1.
- YieldCurveDayCounter [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*.
- ExtrapolationMethod [Optional]: Only applies to bootstrapped curves and interpolated zero / discount curves. Controls how the curve is extrapolated:
  - *ContinuousForward*: The instantaneous forward rate at the last curve pillar date T is kept constant
  - *DiscreteForward*: The discrete forward rate between $T-1$ and T is kept constant, where T is the last curve pillar date and $T-1$ lies one calendar day before T.

- ExcludeT0FromInterpolation [Optional]: Only applies to bootstrapped curves and interpolated zero / discount curves. Set to *True* to exclude the synthetic time-zero (t0) point from yield curve interpolation. When enabled, the curve interpolates only on actual market pillar dates, while still ensuring proper behavior at the reference date (e.g., $DF(t_0) = 1.0$ for discount curves). This is useful when the first market pillar date does not coincide with the as-of date and you want to avoid including a synthetic point in the interpolation.

  This feature is supported for all three interpolation variables:

  – `InterpolationVariable=Discount`: Interpolates discount factors on pillar dates. At $t = 0$, returns $DF(t_0) = 1.0$ by definition. For $0 < t < t_1$ (between reference date and first pillar), uses flat forward extrapolation from the first pillar to ensure continuity at $t = 0$.

  – `InterpolationVariable=Zero`: Interpolates continuously compounded zero rates on pillar dates. For $t < t_1$, uses flat extrapolation with the first pillar's zero rate.

  – `InterpolationVariable=Forward`: Interpolates instantaneous forward rates on pillar dates. For $t < t_1$, uses flat forward rate from the first pillar.

  All standard interpolation methods are supported (e.g., Linear, LogLinear, NaturalCubic, etc.,).

  If the element is omitted or left blank, then it defaults to *False*, which preserves the traditional behavior of including a t0 point (with $DF = 1.0$ for discount curves, or extrapolated rate for zero/forward curves) in the interpolation input.

- `BootstrapConfig` [Optional]: this node holds configuration details for the iterative bootstrap that are described in section 5.7.20. If omitted, this node's default values described in section 5.7.20 are used.

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

Table 17: Allowable interpolation variables.

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 85. 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 5.11. The `Conventions` node associates conventions with the quotes.

For both *Zero* and *Discount* type segments, the quotes can be given using a wildcard. Any valid and matching quotes will then be loaded from the provided market data. Example wildcards are:

- ZERO/RATE/EUR/* - matches all EUR zero rate quotes
- DISCOUNT/RATE/EUR/EUR3M/* - matches all EUR discount factor quotes for the EUR3M curve

Listing 85: 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 86. 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. The `PillarChoice` node determines the bootstrap pillars that are used (MaturityDate, LastRelevantDate, NoPillar, StartDate, StartDateAndMaturityDate, StartDateAndLastRelevantDate; if not given 'LastRelevantDate' is the default value). The `DuplicatePillarPolcicy` node determined how curve instruments within the same segment with identical pillar dates are handled (Keep Last, KeepFirst, KeepAll, ThrowError; if not given 'KeepLast' is used).

The `Priority` node determines the priority of the segment, this has to be a non-negative integer. A lower number means a higher priority (more "important") segment. If two adjacent segments overlap w.r.t. the pillar dates of their instruments, instruments from the segment with lower priority are removed until the overlap is resolved. In addition, a minimum distance (measured in calendar days) between the segments is preserved. This distance is given in the `MinDistance` node for the instruments of the current and following segment. If not given, the priority of a segment defaults to 0 (highest possible priority), the minimum distance defaults to 1. Consider the example given in 87. In this case:

- instruments from the start of the second segment with pillar date strictly earlier than $d_1 + 5$, where d_1 is the maximum pillar date of instruments in the first segment, will be removed
- instruments from the end of the second segment with pillar date strictly later than $d_3 - 10$, where d_3 is the minimum pillar date of instruments in the third segment, will be removed

Listing 86: Simple yield curve segment

```
<Simple>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <PillarChoice> </PillarChoice>
  <DuplicatePillarPolicy> </DuplicatePillarPolicy>
  <Priority> </Priority>
  <MinDistance> </MinDistance>
  <ProjectionCurve> </ProjectionCurve>
</Simple>
```

Listing 87: Example for priorities and min distances

```
<Simple>
  ...
  <Priority>0</Priority>
  <MinDistance>5</MinDistance>
</Simple>
<Simple>
  ...
  <Priority>2</Priority>
  <MinDistance>10</MinDistance>
</Simple>
<Simple>
  ...
  <Priority>1</Priority>
</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 88. This segment is used to hold quotes for Average OIS swap instruments. The `Quotes` node has the structure shown in Listing 89. 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.

For the `PillarChoice`, `DuplicatePillarPolicy`, `Priority` and `MinDistance` nodes see the explanation under "Simple Segment".

Listing 88: Average OIS yield curve segment

```
<AverageOIS>
  <Type> </Type>
  <Quotes>
    <CompositeQuote> </CompositeQuote>
    <CompositeQuote> </CompositeQuote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <PillarChoice> </PillarChoice>
  <DuplicatePillarPolicy> </DuplicatePillarPolicy>
  <Priority> </Priority>
  <MinDistance> </MinDistance>
  <ProjectionCurve> </ProjectionCurve>
</AverageOIS>
```

Listing 89: 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 90. 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 86 except that there are two projection curve nodes. There is a ProjectionCurveReceive node for the index with the shorter tenor. This node holds the CurveId of a curve for projecting the floating rates on the receiving side. Similarly, there is a ProjectionCurvePay node for the index of the pay side. The deprecated values are short for receive, and long for pay. 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.

For the PillarChoice, DuplicatePillarPolicy, Priority and MinDistance nodes see the explanation under "Simple Segment".

Listing 90: Tenor basis yield curve segment

```
<TenorBasis>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
    <!--...-->
  </Quotes>
  <Conventions> </Conventions>
  <PillarChoice> </PillarChoice>
  <DuplicatePillarPolicy> </DuplicatePillarPolicy>
  <Priority> </Priority>
  <MinDistance> </MinDistance>
  <ProjectionCurvePay> </ProjectionCurvePay>
  <ProjectionCurveReceive> </ProjectionCurveReceive>
</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 91. 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 84. 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 91: FX forward yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <PillarChoice> </PillarChoice>
  <DuplicatePillarPolicy> </DuplicatePillarPolicy>
  <Priority> </Priority>
  <MinDistance> </MinDistance>
  <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 92. 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 84. 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 84. 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.

For the `PillarChoice`, `DuplicatePillarPolicy`, `Priority` and `MinDistance` nodes see the explanation under "Simple Segment".

Listing 92: Cross currency basis yield curve segment

```
<CrossCurrency>
  <Type> </Type>
  <Quotes>
    <Quote> </Quote>
    <Quote> </Quote>
          ...
  </Quotes>
  <Conventions> </Conventions>
  <PillarChoice> </PillarChoice>
  <DuplicatePillarPolicy> </DuplicatePillarPolicy>
  <Priority> </Priority>
  <MinDistance> </MinDistance>
  <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 93. This segment is used to build yield curves which are expressed as a spread over some reference yield curve.

Listing 93: 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 94. 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 GlobalAccuracy 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 calibration seed. Random calibration seeds are currently only supported for the NelsonSiegel interpolation method.

Listing 94: 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">EUR6M</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>
```

Bond Yield Shifted

When the node name is `BondYieldShifted`, the `Type` node has the only allowable value *Bond Yield Shifted*, and the node has the structure shown in Listing 95. This segment is used to build yield curves which are adjusted by liquid bond yields. The adjustment is derived as an average of the spreads between the bond's yields-to-maturity and the reference curve level at the tenor points corresponding the bond durations.

Compared to the fitted bond segment the shifted curve can be built with only one liquid bond. This approach is useful in cases of limited number of liquid comparable bonds and hence unstable fitting of Nelson Siegel. The average spread at the average duration point may be considered as a sensitivity point of a corresponding zero coupon bond.

The segment has the following elements:

- Quotes: a list of bond price quotes, for each security in the list, reference data must be available

- ReferenceCurve: the curve which will be used to calculate the bond spread. This curve will also be shifted by the resulting spread
- 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.

Listing 95: Bond Yield Shifted curve segment

```
        <YieldCurve>
        <CurveId>USD.Benchmark.Curve_Shifted</CurveId>
        <CurveDescription>Curve shifted with a bond's spreads at the bond duration tenors</CurveDesc
        <Currency>USD</Currency>
        <DiscountCurve/>
        <Segments>
          <BondYieldShifted>
            <Type>Bond Yield Shifted</Type>
            <ReferenceCurve>USD1D</ReferenceCurve>
            <Quotes>
              <Quote>BOND/PRICE/EJ7706660</Quote>
              <Quote>BOND/PRICE/ZR5330686</Quote>
              <Quote>BOND/PRICE/AS0644417</Quote>
            </Quotes>
            <Conventions>BOND_CONVENTIONS</Conventions>
            <ExtrapolateFlat>true</ExtrapolateFlat>
            <IborIndexCurves>
              <IborIndexCurve iborIndex="USD-LIBOR-3M">USD3M</IborIndexCurve>
            </IborIndexCurves>
          </BondYieldShifted>
        </Segments>
        <InterpolationVariable>Discount</InterpolationVariable>
        <InterpolationMethod>Linear</InterpolationMethod>
        <YieldCurveDayCounter>A365</YieldCurveDayCounter>
        <Tolerance> </Tolerance>
        <Extrapolation>true</Extrapolation>
        <BootstrapConfig> </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 96. 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 5.7.3.

- 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 96: 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 97. 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 97: 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 98. This segment is used to build a projection curve for an Ibor index based on a risk free rate and a spread.

Listing 98: 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 spread are read from the ibor
           fallback configuration -->
      <RfrIndex>USD-SOFR</RfrIndex>
      <Spread>0.0026161</Spread>
    </IborFallback>
  </Segments>
</YieldCurve>
```

Discount Ratio Segment

When the node name is `DiscountRatio`, the `Type` node has the only allowable value *Discount Ratio* and the node has the structure shown in Listing 99. This segment is used to build a curve with discount factors $P(0,t)$ from three input curves with discount factors $P_b(0,t)$, $P_n(0,t)$ and $P_d(0,t)$ (“base”, “numerator”, “denominator” curves) following the equation

$$P(0,t) = P_b(0,t)\frac{P_n(0,t)}{P_d(0,t)} \tag{3}$$

The main use case of this segment is to build a discount curve “CCY1-IN-CCY2” for cashflows in CCY1 collateralized in CCY2 when curves “CCY1-IN-BASE” and “CCY2-IN-BASE” are known for a common base currency BASE:

For a maturity t denote the zero rate on a curve “X” by $r_X(t)$ and the correpsonding discount factor by $P_X(0,t)$. Furthermore, write “CCY” as shorthand for “CCY-IN-CCY”, i.e. for the discount curve for cashflows in the same currency as the collateral currency “CCY”. We write the desired zero rate as

$$\begin{aligned} r_{\text{CCY1-IN-CCY2}} = r_{\text{CCY2}}+(&r_{\text{BASE-IN-CCY2}} - r_{\text{CCY2}})+ \\ &(r_{\text{CCY1-IN-CCY2}} - r_{\text{BASE-IN-CCY2}}) \end{aligned} \tag{4}$$

We now assume that these two rate differentials stay the same when we switch from collateral currency “CCY2” to “BASE”, i.e.

$$r_{\text{BASE-IN-CCY2}} - r_{\text{CCY2}} \approx r_{\text{BASE}} - r_{\text{CCY2-IN-BASE}} \tag{5}$$

$$r_{\text{CCY1-IN-CCY2}} - r_{\text{BASE-IN-CCY2}} \approx r_{\text{CCY1-IN-BASE}} - r_{\text{BASE}} \tag{6}$$

In less technical terms we assume that FX Forward Quotes CCY2 / BASE and CCY1 / BASE stay constant when the collateral currency changes, which seems reasonable, if no further market information is available.

The discount factors associated to the RHS of 5 and 6 can be written

$$P_{\text{BASE}}(0,t)/P_{\text{CCY2-IN-BASE}}(0,t) \tag{7}$$

$$P_{\text{CCY1-IN-BASE}}(0,t)/P_{\text{BASE}}(0,t) \tag{8}$$

and so 3 can be written

$$P_{\text{CCY1-IN-CCY2}}(0,t) = \frac{P_{\text{CCY2}}(0,t)P_{\text{CCY1-IN-BASE}}(0,t)}{P_{\text{CCY2-IN-BASE}}(0,t)} \tag{9}$$

so the following choice of curves will result in the desired “CCY1-IN-CCY2” curve:

- base curve = “CCY2-IN-CCY2”

- numerator curve = "CCY1-IN-BASE"
- denominator curve = "CCY2-IN-BASE"

Listing 99: Discount Ratio segment

```
<YieldCurve>
  <CurveId>GBP-IN-EUR</CurveId>
  <CurveDescription>GBP collateralized in EUR discount curve</CurveDescription>
  <Currency>GBP</Currency>
  <DiscountCurve/>
  <Segments>
    <DiscountRatio>
      <Type>Discount Ratio</Type>
      <BaseCurve currency="EUR">EUR1D</BaseCurve>
      <NumeratorCurve currency="GBP">GBP-IN-USD</NumeratorCurve>
      <DenominatorCurve currency="EUR">EUR-IN-USD</DenominatorCurve>
    </DiscountRatio>
  </Segments>
</YieldCurve>
```

5.7.2 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 9.12 and 9.13 respectively. The bootstrap also requires a market recovery rate quote and this is documented in Section 9.14.

Listing 100 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. For index term curves a suffix `_5Y` should be appended to the name indicating the index term, since this is the preferred name looked up by index cds and index cds option pricers. If such a curve is not found, the pricers will fall back to the specified credit curve id without suffix, i.e. following this naming convention is not mandatory, but recommended.
- `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 or `ConvSpreadCDS` if the `Quotes` reference Conventional CDS spread quotes. Note that if ConvSpreadCDS or Price is used, the model will be IsdaCdsEngine. Else, MidPointCdsEngine.
- `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 24.
- `RecoveryRate`: A valid recovery rate quote name as documented in Section 9.14.
- `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.
- `RunningSpread` [Optional]: The `RunningSpread` is optional and is used for
  - stripping cds curves from upfront quotes. Alternatively the upfront quote labels can contain the running spread.
  - the calculation of the ATM level in cds and index cds volatility surfaces that are strike dependent

  The value should be set whenever one of these use cases applies.
- `IndexTerm` [Optional]: The `IndexTerm` is optional and is used to set up index cds curves for a specific term. If several quotes are specified explicitly or via wildcards, the quote matching the specified term is used to build a flat curve. If no quote is available for the specified term, an interpolated term quote will be built using the adjacent terms of the provided quotes.
- `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 9.12 and if `Type` is `Price`, the quotes should be CDS upfront price quote strings as documented in Section 9.13 and If the `Type` is `ConvSpreadCDS`, the quotes should be Conv CDS spread quote strings as documented in Section 9.12. 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 5.11.23, 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 35. 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 35. 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 5.7.20. If omitted, this node's default values described in section 5.7.20 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).

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

Listing 100: Default curve configuration based on CDS quotes

5.7.3 Benchmark Default Curve

Default curves can be set up as a difference curve of two yield curves as shown in listing 101. 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

$$S(t) = \left(P_S(0,t)/P_B(0,t)\right)^{1/(1-R)} \tag{10}$$

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

$$S(0,t) = P_S(0,t)/P_B(0,t) \tag{11}$$

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 101: Benchmark default curve

5.7.4 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 102.

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

$$S(t) = \left[\left(\frac{P_{S,n}(t)}{P_{S,n}(t_n)} \right)^{(1-R_n)} \Pi_{i=0}^{n-1} \left(\frac{P_{S,i}(t_{i+1})}{P_{S,i}(t_i)} \right)^{(1-R_i)} \right]^{\frac{1}{1-R}} \tag{12}$$

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 102: Multi-Section default curve

5.7.5 Swaption Volatility Structures

Listing 103 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>
    <Interpolation>Hagan2002NormalZeroBeta</Interpolation>
    <ParametricSmileConfiguration>
      <Parameters>
        <Parameter>
          <Name>alpha</Name>
          <InitialValue>0.0050</InitialValue>
          <Calibration>Implied</Calibration>
        </Parameter>
        <Parameter>
          <Name>beta</Name>
          <InitialValue>0.0</InitialValue>
          <Calibration>Fixed</Calibration>
        </Parameter>
        <Parameter>
          <Name>nu</Name>
          <InitialValue>0.30</InitialValue>
          <Calibration>Calibrated</Calibration>
        </Parameter>
        <Parameter>
          <Name>rho</Name>
          <InitialValue>0.0</InitialValue>
          <Calibration>Calibrated</Calibration>
        </Parameter>
      </Parameters>
      <Calibration>
        <MaxCalibrationAttempts>10</MaxCalibrationAttempts>
        <ExitEarlyErrorThreshold>0.005</ExitEarlyErrorThreshold>
        <MaxAcceptableError>0.05</MaxAcceptableError>
      </Calibration>
    </ParametricSmileConfiguration>
    <Extrapolation>Flat</Extrapolation>
    <OutputVolatilityType>Normal</OutputVolatilityType>
    <OutputShift>0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0</OutputShift>
    <ModelShift>0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0</ModelShift>
    <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>
    <QuoteTag/>
  </SwaptionVolatility>
  ...
</SwaptionVolatilities>
```

Listing 103: Swaption volatility configuration

The meaning of each of the elements in Listing 103 is given below.

- CurveId: Unique identifier of the swaption volatility structure
- CurveDescription [Optional]: 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.
- Interpolation: Optional. Possible values: Linear, Hagan2002Lognormal, Hagan2002Normal, Hagan2002NormalZeroBeta, Antonov2015FreeBoundaryNormal, KienitzLawsonSwaynePde, FlochKennedy[6]. If not given, defaults to Linear.
- ParametricSmileConfiguration: Optional. Applies to SABR only. If not given, default values are used. Allows to specify initial values for calibrated parameters, to exclude single parameters from calibration and to set calibration parameters.

[6]FlochKennedy shows unstable results and is considered experimental. More work is needed.

See Example 59 for how to configure single value and termstructures of sabr parameters for swaption and cap curve configs.

- Extrapolation: Specifies the extrapolation behaviour in all dimensions. Allowable values: `Linear, Flat, None`
- OutputVolatilityType: Optional, defaults to input volatility type. Possible values: Normal, Lognormal (alias for ShiftedLognormal, shift is taken from OutputShift if given, or input market data shift. Input market quotes will be converted to output volatility type and shift before building the vol surfaces, except for parametric models (SABR), where the model is calibrated to the input market data directly (without conversion) and the output volatility type and shift is handled by the calibrated parametric model. An early conversion of the input market data would not be an advantage in this case, we could only loose information.
- OutputShift: Optional, defaults to input market data shift. Specifies the shift if OutputVolatilityType is Lognormal / ShiftedLognormal
- ModelShift: Optional, defaults to input market data shift. Specifies the shift used for SABR model if applicable
- 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. Note that trailing 0s are not ignored.
- QuoteTag [Optional]: If non-empty, a tag will be included in the market datum labels. This can be used to set up underlying specific volatility date. For

example, if the quote tag is set to EUR-EURIBOR-3M, the market datum labels will be `SWAPTION/RATE_LNVOL/EUR/EUR-EURIBOR-3M/5Y/10Y/ATM` instead of `SWAPTION/RATE_LNVOL/EUR/5Y/10Y/ATM`. See section 9.20.

5.7.6 Cap Floor Volatility Structures

The cap volatility structure parameterisation allows the user to pick out term cap volatilities or optionlet volatilities in the market data.

If term cap volatilities are given, users can 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.

If optionlet volatilities are given, no bootstrapping will be performed on the input market data. The curve or surface will be constructed using the interpolation method defined by user. The parameterisation allows for three separate types of input optionlet volatilities structures:

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

The input volatilities may be normal, lognormal or shifted lognormal. The structure of the market quotes is provided in Table 57.

Whether the input market data are term cap volatilities or optionlet volatilities depends on the value of the `InputType` node. This node may be set to `TermVolatilities` for term cap volatilities or `OptionletVolatilities` for optionlet volatilities.

For term cap volatilities, 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`. This node will be ignored if the inputs are optionlet volatilities.

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 six XML configurations separately for clarity:

1. Term volatility ATM curve with interpolation on term volatilities.
2. Term volatility ATM curve with interpolation on optionlet volatilities.
3. Term volatility surface, possibly including an ATM column, with interpolation on term volatilities.
4. Term volatility surface, possibly including an ATM column, with interpolation on optionlet volatilities.
5. Optionlet volatility ATM curve.
6. Optionlet volatility surface.

Before describing the different configurations we summarize the usage of the various interpolation fields:

1. If InterpolateOn is set to TermVolatilities, InterpolationMethod is used to interpolate the term volatilities during the bootstrap and to interpolate the optionlet volatilities of the final optionlet surface. The fields TimeInterpolation and StrikeInterpolation are ignored in this case.
2. If InterpolateOn is set to OptionletVolatilities, TimeInterpolation and StrikeInterpolation are used to interpolate optionlet volatilities both during the bootstrap and on the final optionlet surface. InterpolationMethod is ignored in this case.

Listing 104 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.

- `OutputVolatilityType`: Specifies the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0

- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface (if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.

- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.

- `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 35. As we are describing an ATM curve here, this node should be set to `true` as shown in 104.

- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 24.

- `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 23.

- `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 19 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 whether market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are provided. If false, the curve will fail to build if any quotes are missing. This also applies 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 25.
- `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 104.
- `BootstrapConfig` [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 5.7.20. If omitted, this node's default values described in section 5.7.20 are used.
- `InputType` [Optional]: The type of the marketdata input. Allowable values are `TermVolatilities` or `OptionletVolatilities`. As we are describing term cap volatilities input, this should be set to `TermVolatilities` or omitted as shown in Listing 104. If omitted, the default value is `TermVolatilities`.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <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>
  <InputType>TermVolatilities</InputType>
</CapFloorVolatility>
```

Listing 104: ATM cap floor configuration with interpolation on term volatilities.

Listing 105 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.
- `OutputVolatilityType`: Specified the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0
- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface (if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.
- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.
- `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 35. As we are describing an ATM curve here, this node should be set to `true` as shown in 105.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 24.
- `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 23.

- `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 19 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 whether market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are provided. If false, the curve will fail to build if any quotes are missing. This also applies 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 25.
- `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 105.
- `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 5.7.20. If omitted, this node's default values described in section 5.7.20 are used.

- `InputType` [Optional]: The type of the marketdata input. Allowable values are `TermVolatilities` or `OptionletVolatilities`. As we are describing term cap volatilities input, this should be set to `TermVolatilities` or omitted as shown in Listing 105. If omitted, the default value is `TermVolatilities`.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <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>
  <InputType>TermVolatilities</InputType>
</CapFloorVolatility>
```

Listing 105: ATM cap floor configuration with interpolation on optionlet volatilities.

Listing 106 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.
- `OutputVolatilityType`: Specified the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0
- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface

(if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.

- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.
- `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 35. 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 24.
- `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 23.
- `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 19 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 whether market data quotes for all tenors are required. If true, we attempt to build the curve from whatever quotes are provided. If false, the curve will fail to build if any quotes are missing. This also applies 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 25.

- 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 106.
- BootstrapConfig [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 5.7.20. If omitted, this node's default values described in section 5.7.20 are used.
- InputType [Optional]: The type of the marketdata input. Allowable values are TermVolatilities or OptionletVolatilities. As we are describing term cap volatilities input, this should be set to TermVolatilities or omitted as shown in Listing 106. If omitted, the default value is TermVolatilities.

```xml
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <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>
  <InputType>TermVolatilities</InputType>
</CapFloorVolatility>
```

Listing 106: Cap floor surface with interpolation on term volatilities.

Listing 107 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.
- `OutputVolatilityType`: Specified the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0
- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface (if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.
- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.
- `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 35. 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 24.
- `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 23.
- `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 19 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 whether market data quotes for all tenors and strikes are required. If true, we attempt to build the curve from whatever quotes are provided. If false, the curve will fail to build if any quotes are missing. This also applies 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 25.
- `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 107.
- `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` or one of the SABR variants Hagan2002Lognormal, Hagan2002Normal, Hagan2002NormalZeroBeta, Antonov2015FreeBoundaryNormal, KienitzLawsonSwaynePde, FlochKennedy.

The SABR variants are only supported for InterpolateOn = OptionVolatilities (or if InputType = OptionletVolatilities). If not set, `LinearFlat` is assumed.

- ParametricSmileConfiguration: Optional. Applies to SABR only. If not given, default values are used. Allows to specify initial values for calibrated parameters, to exclude single parameters from calibration and to set calibration parameters. See listing 110. See Example 59 for how to configure single value and termstructures of sabr parameters for swaption and cap curve configs.

- `QuoteIncludesIndexName` [Optional]: If true, the quote labels that are looked up in the market data to build the surface include the index name as e.g. in `CAPFLOOR/RATE_NVOL/USD/USD-LIBOR-3M/1Y/3M/0/0/0.01`. If false, the index name is not include as in `CAPFLOOR/RATE_NVOL/USD/1Y/3M/0/0/0.01`. If the flag is not given, it defaults to false. Including the index name in the market quotes allows to build cap surfaces on different underlying indices with the same tenor. The flag also affects shift quotes as e.g. `CAPFLOOR/SHIFT/USD/USD-LIBOR-3M/5Y` (index included in quote) vs. `CAPFLOOR/SHIFT/USD/5Y` (index not included in quote).

- `BootstrapConfig` [Optional]: This node holds configuration details for the iterative bootstrap that are described in section 5.7.20. If omitted, this node's default values described in section 5.7.20 are used.

- `InputType` [Optional]: The type of the marketdata input. Allowable values are `TermVolatilities` or `OptionletVolatilities`. As we are describing term cap volatilities input, this should be set to `TermVolatilities` or omitted as shown in Listing 107. If omitted, the default value is `TermVolatilities`.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <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>
  <QuoteIncludesIndexName>...</QuoteIncludesIndexName>
  <BootstrapConfig>...</BootstrapConfig>
  <InputType>TermVolatilities</InputType>
</CapFloorVolatility>
```

Listing 107: Cap floor surface with interpolation on optionlet volatilities.

Listing 108 shows the layout for parameterising an ATM optionlet volatility curve. 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 optionlet curve. 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.
- `OutputVolatilityType`: Specified the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0
- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface (if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.
- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.
- `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 35. As we are describing an ATM curve here, this node should be set to `true` as shown in 108.
- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 24.
- `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 23.
- `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 19 under `Roll Convention`.
- `Tenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the tenor by strike surface. A single wildcard

character, *, can also be used for wildcard tenor. In this case, i.e. configuring a surface, they must be provided.

- OptionalQuotes [Optional]: A boolean flag to indicate whether market data quotes for all tenors are required. Optionlet volatilities do not support optional quotes, so this node should be false or omitted. 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 25.
- DiscountCurve: A reference to a valid discount curve specification that will be used to discount cashflows. 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. A single wildcard character, *, can also be used for wildcard tenor. In this case, all the tenors found in the market data input will be used to construct 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.
- TimeInterpolation [Optional]: Indicates the interpolation and extrapolation, if allowed by the Extrapolation node, in the time direction. 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.
- InputType: The type of the marketdata input. Allowable values are TermVolatilities or OptionletVolatilities. As we are describing ATM curve on optionlet volatilities, this should be set to OptionletVolatilities as shown in Listing 108.

```xml
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <Extrapolation>...</Extrapolation>
  <IncludeAtm>true</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>false</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <TimeInterpolation>...</TimeInterpolation>
  <BootstrapConfig>...</BootstrapConfig>
  <InputType>OptionletVolatilities</InputType>
</CapFloorVolatility>
```

Listing 108: ATM cap floor configuration with optionlet volatilities input.

Listing 109 shows the layout for parameterising an optionlet tenor by absolute optionlet strike volatility surface. This parameterisation also allows for the inclusion of an optionlet 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 ATM optionlet curve. 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.
- `OutputVolatilityType`: Specified the vol used for caplet bootstrap and result surface, one of `Normal`, `Lognormal` or `ShiftedLognormal`, defaults to Normal for backwards compatibility before release 1.83.0. Only available for SABR interpolation if input type is set to optionlet vols.
- `OutputShift`: Specifies the vol shift used for caplet bootstrap and result surface (if OutputVolatilityType is `Lognormal` or `ShiftedLognormal`), defaults to input market data shift.
- `ModelShift`: Specifies the vol shift used for SABR model (if applicable). Defaults to input market data shift.
- `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 in combination with the surface. Allowable boolean values are given in the Table 35. 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 sections above. The optionlet surface is amended by inserting the optionlet volatilities at the forecast fixings.

- `DayCounter`: The day counter used to convert from dates to times in the underlying structure. Allowable values are given in the Table 24.

- `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 23.

- `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 19 under `Roll Convention`.

- `Tenors` [Optional]: A comma separated list of valid tenor strings giving the cap floor maturity tenors to be used in the surface. A single wildcard character, `*`, can also be used for wildcard tenor. In this case, all the tenors found in the market data input will be used to construct 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 whether market data quotes for all tenors are required. Optionlet volatilities do not support optional quotes, so this node should be false or omitted. 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 25.

- `DiscountCurve`: A reference to a valid discount curve specification that will be used to discount cashflows. 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. A single wildcard character, `*`, can also be used for wildcard tenor. 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.

- `TimeInterpolation` [Optional]: Indicates the interpolation and extrapolation, if allowed by the `Extrapolation` node, in the time direction. 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` [Optional]: 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.

- QuoteIncludesIndexName [Optional]: If true, the quote labels that are looked up in the market data to build the surface include the index name as e.g. in CAPFLOOR/RATE_NVOL/USD/USD-LIBOR-3M/1Y/3M/0/0/0.01. If false, the index name is not include as in CAPFLOOR/RATE_NVOL/USD/1Y/3M/0/0/0.01. If the flag is not given, it defaults to false. Including the index name in the market quotes allows to build cap surfaces on different underlying indices with the same tenor. The flag also affects shift quotes as e.g. CAPFLOOR/SHIFT/USD/USD-LIBOR-3M/5Y (index included in quote) vs. CAPFLOOR/SHIFT/USD/5Y (index not included in quote).
- InputType: The type of the marketdata input. Allowable values are TermVolatilities or OptionletVolatilities. As we are describing ATM curve on optionlet volatilities, this should be set to OptionletVolatilities as shown in Listing 109.

```
<CapFloorVolatility>
  <CurveId>...</CurveId>
  <CurveDescription>...</CurveDescription>
  <VolatilityType>...</VolatilityType>
  <OutputVolatilityType>...</OutputVolatilityType>
  <OutputShift>...</OutputShift>
  <ModelShift>...</ModelShift>
  <Extrapolation>...</Extrapolation>
  <IncludeAtm>...</IncludeAtm>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <BusinessDayConvention>...</BusinessDayConvention>
  <Tenors>...</Tenors>
  <OptionalQuotes>false</OptionalQuotes>
  <IborIndex>...</IborIndex>
  <DiscountCurve>...</DiscountCurve>
  <AtmTenors>...</AtmTenors>
  <SettlementDays>...</SettlementDays>
  <TimeInterpolation>...</TimeInterpolation>
  <StrikeInterpolation>...</StrikeInterpolation>
  <QuoteIncludesIndexName>...</QuoteIncludesIndexName>
  <InputType>OptionletVolatilities</InputType>
</CapFloorVolatility>
```

Listing 109: Cap floor surface with optionlet volatilities input.

```
<ParametricSmileConfiguration>
  <Parameters>
    <Parameter>
      <Name>alpha</Name>
      <InitialValue>0.0050</InitialValue>
      <Calibration>Implied</Calibration>
    </Parameter>
    <Parameter>
      <Name>beta</Name>
      <InitialValue>0.0</InitialValue>
      <Calibration>Fixed</Calibration>
    </Parameter>
    <Parameter>
      <Name>nu</Name>
      <InitialValue>0.30</InitialValue>
      <Calibration>Calibrated</Calibration>
    </Parameter>
    <Parameter>
      <Name>rho</Name>
      <InitialValue>0.0</InitialValue>
      <Calibration>Calibrated</Calibration>
    </Parameter>
  </Parameters>
  <Calibration>
    <MaxCalibrationAttempts>10</MaxCalibrationAttempts>
    <ExitEarlyErrorThreshold>0.005</ExitEarlyErrorThreshold>
    <MaxAcceptableError>0.05</MaxAcceptableError>
  </Calibration>
</ParametricSmileConfiguration>
```

Listing 110: Cap floor surface with optionlet volatilities input.

5.7.7 Proxy Cap Floor Volatility Configuration

In addition to the standard cap floor volatility configurations described above, ORE supports proxy cap floor volatility surfaces. A proxy configuration allows you to create a cap floor volatility surface by referencing an existing cap floor volatility surface and adjusting it for different underlying indices and optionally applying a scaling factor.

Listing 111 shows the layout for parameterising a proxy cap floor volatility surface. 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.
- `ProxyConfig`: Contains the proxy configuration settings. This node is required for proxy configurations and should be omitted for direct cap floor volatility configurations.
- `Source` (within ProxyConfig): Specifies the source cap floor volatility surface to use as the base for the proxy.
- `CurveId` (within Source): The curve identifier of the source cap floor volatility surface that will be used as the base. This must reference an existing cap floor volatility curve configuration.
- `Index` (within Source): The interest rate index associated with the source cap floor volatility surface.
- `RateComputationPeriod` [Optional] (within Source): The rate computation period for the source index. This is only required for OIS or BMA/SIFMA indices and should be omitted for standard IBOR indices.
- `Target` (within ProxyConfig): Specifies the target configuration for the proxy surface.

- `Index` (within Target): The target interest rate index for which the proxy surface will provide volatilities. This allows you to create a volatility surface for one index based on the volatility surface of another similar index.
- `RateComputationPeriod` [Optional] (within Target): The rate computation period for the target index. This is only required for OIS or BMA/SIFMA indices and should be omitted for standard IBOR indices.
- `ONCapSettlementDays` [Optional] (within Target): Settlement days for overnight index cap structures. Only relevant when the target index is an overnight index.
- `ScalingFactor` [Optional] (within ProxyConfig): An optional scaling factor to multiply the proxied volatility surface. If omitted, defaults to 1.0 (no scaling). This allows you to scale the entire volatility surface up or down by a constant factor. For example, a value of 1.1 would increase all volatilities by 10%, while 0.9 would decrease them by 10%. The scaling factor must be positive.

The proxy mechanism works by:

1. Taking the base volatility surface from the specified source curve
2. Adjusting the volatilities for differences in forward rate levels between the source and target indices
3. Optionally applying the scaling factor to all volatilities
4. Producing an optionlet volatility structure suitable for pricing instruments on the target index

This feature is particularly useful when you have a well-calibrated volatility surface for one index (e.g., USD-LIBOR-3M) and want to create a similar surface for a related index (e.g., USD-SOFR) by applying appropriate adjustments and scaling.

```xml
<CapFloorVolatility>
  <CurveId>USD-SOFR</CurveId>
  <CurveDescription>USD SOFR Cap Floor Volatility via USD LIBOR Proxy</CurveDescription>
  <ProxyConfig>
    <Source>
      <CurveId>USD-LIBOR-3M</CurveId>
      <Index>USD-LIBOR-3M</Index>
    </Source>
    <Target>
      <Index>USD-SOFR</Index>
      <RateComputationPeriod>3M</RateComputationPeriod>
      <ONCapSettlementDays>2</ONCapSettlementDays>
    </Target>
    <ScalingFactor>0.95</ScalingFactor>
  </ProxyConfig>
</CapFloorVolatility>
```

Listing 111: Proxy cap floor volatility configuration with scaling factor.

5.7.8 FX Volatility Structures

Listings 112, 113, 114, 115, 116, 117 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/EUR-IN-USD</FXForeignCurveID>
  <FXDomesticCurveID>Yield/USD/USD1D</FXDomesticCurveID>
  <DayCounter>A365</DayCounter>
  <Calendar>US,TARGET</Calendar>
  <Conventions>EUR-USD-FXOPTION</Conventions>
</FXVolatility>
```

Listing 112: 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 113: 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>
  <SmileExtrapolation>UseInterpolator</SmileExtrapolation>
</FXVolatility>
```

Listing 114: FX option volatility configuration Smile / Delta

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

```
    <SmileType>BFRR</SmileType>
    <SmileInterpolation>Cubic</SmileInterpolation>
    <TimeInterpolation>V2T</TimeInterpolation>
    <TimeWeighting>USD-JPY-FXOPTION-TIMEWEIGHTING</TimeWeighting>
    <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>
    <ButterflyErrorTolerance>0.01</ButterflyErrorTolerance>
  </FXVolatility>
```

Listing 115: FX option volatility configuration Smile / BFRR with 10 and 25 BF and RR

```
  <FXVolatility>
    <CurveId>USDJPY</CurveId>
    <CurveDescription />
    <Dimension>Smile</Dimension>
    <SmileType>Absolute</SmileType>
    <SmileInterpolation>Cubic</SmileInterpolation>
    <Expiries>1M,3M,6M,1Y,2Y,3Y,10Y</Expiries>
    <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 116: FX option volatility configuration Smile / Absolute vols

```
  <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 117: FX option volatility configuration ATM Triangulated

The meaning of each of the elements in Listings 112, 113, 114, 115, 116, 117 is given below.

- CurveId: Unique identifier of the FX volatility structure
- CurveDescription [Optional]: 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`, `Absolute`, 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 and Absolute if left blank

- TimeInterpolation [Optional]: specifies whether time interpolation is done in volatility (`V`, default) or variance (`V2T`)

- ButterflyErrorTolerance [Optional]: max allowed error for broker butterfly match, defaults to 0.01 relative rmse of broker butterfly market premiums vs theoretical premiums

- TimeWeighting [Optional]: if given, a time weighting scheme is applied for time interpolation, see the documentation of FxOptionTimeWeighting convention for more details

- 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` or `Absolute` curves.

- Deltas [Optional]: Strike grid, required in case of SmileType `Delta`
Allowable values: `ATM, *P, *C`, see example in Listing 114

- SmileDelta [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 115

- 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 5.11.19 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
- SmileExtrapolation [Optional]: Applicable only in case of SmileType Delta. Indicates the extrapolation in the smile direction. The allowable values are None, UseInterpolator (or Linear) and Flat. Both Flat and None give flat extrapolation. UseInterpolator indicates that the configured interpolation should be continued in the strike direction in order to extrapolate. Default if not provided value is Flat.

5.7.9 Equity Curve Structures

Listing 118 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 Table 18 for allowed interpolation methods -->
      <InterpolationMethod>Linear</InterpolationMethod>
    </DividendInterpolation>
    <DividendExtrapolation>False</DividendExtrapolation>
    <!-- Optional node, defaults to false -->
    <Extrapolation>False</Extrapolation>
    <DayCounter>A365</DayCounter>
  </EquityCurve>
  <EquityCurve>
    ...
  </EquityCurve>
</EquityCurves>
```

Listing 118: Equity curve configuration

The meaning of each of the elements is given below.

- CurveId: Unique identifier of the equity curve structure.
- CurveDescription [Optional]: A description of the equity curve structure, may be left blank.
- Currency: Currency of the equity.
- Calendar [Optional]: The term structure's calendar used in tenor to date conversions. Defaults to the calendar corresponding to `Currency`.
- ForecastingCurve: CurveId of the curve used for discounting equity fixing forecasts.
- Type: The quote types in `Quote` (e.g. option premium, forward equity price) and whether dividends are taken into account. Allowable values: `DividendYield, ForwardPrice, ForwardDividendPrice, OptionPremium, NoDividends`
- ExerciseStyle [Optional]: Exercise type of the underlying option quotes. Only required if `Type` is *OptionPremium*. Allowable values: `American, European`
- SpotQuote: Market datum ID/name of the current spot rate for the equity underlying.
- Quotes [Optional]: Market datum IDs/names to be used in building the curve structure.
- DayCounter [Optional]: The term structure's day counter used in date to time conversions. Defaults to `A365F`.
- DividendInterpolation [Optional]: This node contains an `InterpolationVariable` and `InterpolationMethod` sub-node, which define the variable on which the interpolation is performed and the interpolation method for the dividend curve, respectively. The allowable values are found in Table 17 and Table 18, respectively. This should not be provided if `Type` is `NoDividends`.
- DividendExtrapolation [Optional]: Boolean flag indicating whether extrapolation in the dividend curve is allowed. If True the dividend curve is extrapolated forward at the risk free rate beyond the last date of the curve - this is only considered when `Extrapolation` is False. Defaults to `False`.
- Extrapolation [Optional]: Boolean flag indicating whether extrapolation in the forward price is allowed. Defaults to `False`.

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.

5.7.10 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 [Optional]: A description of the curve. This field may be left blank.
- EquityId: [Optional] Identifies the underlying equity name, this is used in the construction of the `Quote` strings. If omitted the `CurveId` is used instead.
- 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 5.7.21. 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 119 shows the configuration of equity volatility structures with constant volatility. A node `Constant` takes one `Quote`, as described in Section 9.24, which is held constant for all strikes and expiries.

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

Listing 119: Equity option volatility configuration - constant

Secondly, the volatility curve configuration layout is given in Listing 120. 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 9.24. 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.

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

Listing 120: Equity option volatility configuration - curve

The volatility strike surface configuration layout is given in Listing 121. 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>
    <EquityId>RIC:.SP5</EquityId>
    <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 121: Equity option volatility configuration - strike surface

A volatility surface can also be given in terms of moneyness levels as shown in listing 122. 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 9.24 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 9.24 for details of the market data.

```
<EquityVolatilities>
  <EquityVolatility>
```

```
    <CurveId>SP5</CurveId>
    <CurveDescription>Lognormal option implied vols for SP 500</CurveDescription>
    <EquityId>RIC:.SP5</EquityId>
    <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 122: Equity option volatility configuration - moneyness surface

Finally, the volatility proxy surface configuration layout is given in Listing 123. 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 `EquityVolatilityCurve` field, which must match the `CurveId` of another configuration. `FXVolatilityCurve` and `CorrelationCurve` must be provided if the currency of the proxy surface is different to that of current surface, that can be omitted otherwise. The proxy surface looks up the volatility in the reference surface based on moneyness.

```
<EquityVolatilities>
  <EquityVolatility>
    <CurveId>ABC</CurveId>
    <CurveDescription>Lognormal option implied vols for APC - proxied from SP5</CurveDescription>
    <EquityId>RIC:.SP5</EquityId>
    <Currency>USD</Currency>
    <DayCounter>Actual/365 (Fixed)</DayCounter>
    <ProxySurface>
      <EquityVolatilityCurve>RIC:.SPX</EquityVolatilityCurve>
      <FXVolatilityCurve>GBPUSD</FXVolatilityCurve>
      <CorrelationCurve>FX-GENERIC-GBP-USD&amp;EQ-RIC:VOD.L</CorrelationCurve>
    </ProxySurface>
  </EquityVolatility>
  <EquityVolatility>
    ...
  </EquityVolatility>
</EquityVolatilities>
```

Listing 123: Equity option volatility configuration - proxy surface

5.7.11 Inflation Curves

Listing 124 shows the configuration of an inflation curve. The inflation curve specific elements are the following:

```xml
<InflationCurves>
    <InflationCurve>
        <CurveId>USCPI_ZC_Swaps</CurveId>
        <CurveDescription>Estimation Curve for USCPI</CurveDescription>
        <NominalTermStructure>Yield/USD/USD1D</NominalTermStructure>
        <Type>ZC</Type>
        <Segments>
            <Segment>
                <Conventions>USCPI_INFLATIONSWAP_2M</Conventions>
                <Quotes>
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/3M</Quote>
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/4M</Quote>
                    <!-- ...more quotes... -->
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/24M</Quote>
                </Quotes>
            </Segment>
            <Segment>
                <Conventions>USCPI_INFLATIONSWAP</Conventions>
                <Quotes>
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/3Y</Quote>
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/4Y</Quote>
                    <!-- ...more quotes... -->
                    <Quote>ZC_INFLATIONSWAP/RATE/USCPI/30Y</Quote>
                </Quotes>
            </Segment>
        </Segments>
        <Extrapolation>true</Extrapolation>
        <Calendar>US</Calendar>
        <DayCounter>A365</DayCounter>
        <Lag>3M</Lag>
        <Frequency>Monthly</Frequency>
        <BaseRate/>
        <Tolerance>0.000000000001</Tolerance>
        <InterpolationVariable>ZeroRate</InterpolationVariable>
        <InterpolationMethod>Linear</InterpolationMethod>
        <UseLastFixingDate>false</UseLastFixingDate>
        <Seasonality>
            <BaseDate>20160101</BaseDate>
            <Frequency>Monthly</Frequency>
            <Factors>
                <Factor>SEASONALITY/RATE/MULT/USCPI/JAN</Factor>
                <Factor>SEASONALITY/RATE/MULT/USCPI/FEB</Factor>
                <!-- ...more quotes... -->
                <Factor>SEASONALITY/RATE/MULT/USCPI/DEC</Factor>
            </Factors>
        </Seasonality>
    </InflationCurve>
</InflationCurves>
```

Listing 124: Inflation Curve Configuration

5.7.11.1 Legacy Schema

The legacy schema (still supported) uses a single block for quotes and conventions:

```
<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>
        <!-- ...more quotes... -->
        <Quote>ZC_INFLATIONSWAP/RATE/USCPI/40Y</Quote>
    </Quotes>
    <Conventions>USCPI_INFLATIONSWAP</Conventions>
    <!-- ... -->
</InflationCurve>
```

- `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.
- `Segments`: Allows multiple instrument/convention blocks.
- `Quotes`: Legacy (use instead segments). The instruments' market quotes from which to bootstrap the curve.
- `Conventions`: Legacy (use instead segments). 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.
- `InterpolationVariable`: Optional, the variable on which the interpolation of the zero inflation curve is performed. The allowable values *ZeroRate* and *PriceIndex*. If not provided, the default value is *ZeroRate*. Ignored for year on year inflation curves.
- `InterpolationMethod`: Only relevant if InterpolationVariable is PriceIndex, allowed is *Linear* and *LogLinear*. Defaults to *Linear*.

The segment block defines instruments to use for curve building, with their quotes and swap conventions.

5.7.11.2 Segment Schema

- `Conventions`: The conventions applicable to the segment's instruments.
- `Quotes`: The instruments' market quotes for the segment.

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 seasonality 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.

5.7.11.3 Notes

- All swaps used in segments must reference the same inflation index name for curve building, but can differ in their swap conventions (e.g., lag, interpolation).
- 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.

5.7.12 Inflation Cap/Floor Volatility Surfaces

Listing 125 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 125: 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 indicate 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.

5.7.13 CDS Volatilities

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

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

Firstly, the constant volatility configuration layout is given in Listing 126. The single volatility quote ID, `constant_quote_id`, in the `Quote` node should be a CDS option volatility quote as described in Section 9.15. 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 126: Constant CDS volatility configuration

Secondly, the volatility curve configuration layout is given in Listing 127. 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 9.15. 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>
  <Terms>
    <Term>
      <Label>...</Label>
      <Curve>...</Curve>
    </Term>
  </Terms>
  <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 127: CDS volatility curve configuration

For backwards compatibility, the volatility curve configuration can also be given using a single Expiries node as shown in Listing 128. Note that this configuration is deprecated and the configuration in 127 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 is populated with the curve id or with the `[QuoteName]` if that is specified. The rules for including market quotes into the volatility surface construction are as follows:

- All quotes explicitly specified with their full name are loaded (applies to configs of type constant or curve without wildcards)
- If a quote does not contain a term, we only load it if at most one term is specified in the vol curve config. The quote gets the unique term specified in the vol curve configs assigned or 5Y if the config does not specify any terms.
- If a quote contains a term if this matches one of the configured terms in the curve configuration or if the curve configuration does not specify any terms.

The `[Terms]` node specifies a list of term labels "5Y", "7Y", ... and associated credit curve spec names representing the curve suitable to estimate the ATM level for that term.

If only one expiry is provided in the list, there is only one quote and a constant volatility structure is configured as in Listing 126. If more than one expiry is provided, a curve is configured as in 127. The interpolation is `Linear`, the extrapolation is `Flat` and `EnforceMontoneVariance` is `true`.

```
<CDSVolatility>
  <CurveId>..<CurveId>
  <CurveDescription>...</CurveDescription>
  <Terms>
    <Term>
      <Label>...</Label>
      <Curve>...</Curve>
    </Term>
  </Terms>
  <Expiries>...</Expiries>
  <DayCounter>...</DayCounter>
  <Calendar>...</Calendar>
  <QuoteName>...</QuoteName>
</CDSVolatility>
```

Listing 128: Legacy deprecated CDS volatility curve configuration

Thirdly, the volatility surface configuration layout is given in Listing 129. 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 defaults 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 deprecated CDS volatility curve configuration.
- `StrikeFactor`: this node is optional and the allowable value is any positive 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.

When the CDS volatility surface is configured as in Listing 129, 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 deprecated 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, `*`.

```
<CDSVolatility>
  <CurveId/>
  <CurveDescription/>
  <Terms>
    <Term>
      <Label>...</Label>
      <Curve>...</Curve>
    </Term>
  </Terms>
  <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 129: CDS volatility surface configuration

5.7.14 Base Correlations

Base correlations can be configured using either quoted base correlations directly or by implying them from quoted upfront quotes. When using quoted base correlations, the market quotes are directly used to construct the base correlation curve. Alternatively, base correlations can be implied from upfront quotes of tranches, which involves a calibration process to derive the base correlation values that best fit the observed market prices of the tranches.

Listing 130 shows the configuration of a Base Correlation curve from quoted correlations.

```
  <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 130: Base Correlation Configuration

The meaning of each of the elements in Listing 130 is given below.

- CurveId: Unique identifier of the base correlation structure
- CurveDescription [Optional]: 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

Listing 131 shows the configuration of a Base Correlation curve from upfront quotes or base correlations if it fails to imply the base correlations.

```
<BaseCorrelation>
  <CurveId>RED:2I666VDI3</CurveId>
  <CurveDescription />
  <Terms>*</Terms>
  <DetachmentPoints>*</DetachmentPoints>
  <SettlementDays>0</SettlementDays>
  <Calendar>weekends only</Calendar>
  <BusinessDayConvention>Following</BusinessDayConvention>
  <DayCounter>Actual/365 (Fixed)</DayCounter>
  <Extrapolate>true</Extrapolate>
  <QuoteName>2I666VDI3</QuoteName>
  <StartDate>2023-09-20</StartDate>
  <Rule>CDS2015</Rule>
  <AdjustForLosses>true</AdjustForLosses>
  <QuoteTypes>
    <QuoteType>PRICE</QuoteType>
    <QuoteType>BASE_CORRELATION</QuoteType>
  <QuoteTypes>
  <IndexSpread>0.01</IndexSpread>
  <Currency>EUR</Currency>
  <CalibrateConstituentsToIndexSpread>true</CalibrateConstituentsToIndexSpread>
  <AdjustForLosses>true</AdjustForLosses>
  <UseAssumedRecovery>true</UseAssumedRecovery>
  <RecoveryGrid>
    <Grid seniority="SECDOM">0.4</Grid>
    <Grid seniority="SNRFOR">0.7, 0.40000000000000002, 0.10000000000000001</Grid>
    <Grid seniority="SNRLAC">0.4</Grid>
    <Grid seniority="SUBLT2">0.2</Grid>
  </RecoveryGrid>
  <RecoveryProbabilities>
    <Probabilities seniority="SECDOM">1</Probabilities>
    <Probabilities seniority="SNRFOR">0.35, 0.29999999999999999, 0.34999999999999998</Probabilities>
    <Probabilities seniority="SNRLAC">1</Probabilities>
    <Probabilities seniority="SUBLT2">1</Probabilities>
  </RecoveryProbabilities>
</BaseCorrelation>
```

Listing 131: Base Correlation Configuration from Upfront Quotes

The meaning of each of the elements in Listing 131 is given below.

- CurveId: Unique identifier of the base correlation structure.

- CurveDescription [Optional]: 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 the time direction. A wildcard (*) can be used to include all terms.
- DetachmentPoints: Comma-separated list of equity tranche detachment points, sorted in increasing order. A wildcard (*) can be used to include all detachment points.
- SettlementDays: The floating term structure's settlement days argument used in the reference date calculation.
- 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.
- DayCounter: The term structure's day counter used in date to time conversions.
- Extrapolate: Boolean to indicate whether the correlation curve shall be extrapolated or not.
- QuoteName: The name of the quote used for the base correlation.
- StartDate: The start date for the base correlation curve.
- Rule: The rule used for the base correlation curve, e.g., CDS2015.
- AdjustForLosses: Boolean to indicate whether to adjust for losses.
- QuoteTypes: A priority list of quote types to use for the base correlation curve. If it fails to build a base correlation from the first quote type it will try the next one.
- IndexSpread: The spread of the index.
- Currency: The currency of the base correlation curve.
- CalibrateConstituentsToIndexSpread: Relevant only when price quotes are used. Indicates if the constituents should be calibrated to the index spread before deriving the base correlations from the tranche prices.
- UseAssumedRecovery: Boolean flag to indicate whether to use specialized default curves bootstrapped with assumed recovery rates (mean of the stochastic recovery rate distribution). These curves have following the naming convention __CDO_CurveName_&_REC_0.30_&_ and are used to derive the base correlations.
- RecoveryGrid: Specifies the recovery rates for different seniorities of the constituent. Use wildcard * to apply one grid for all seniorities.
- RecoveryProbabilities: Specifies the probabilities of the recovery rates for different seniorities. Use wildcard * to apply one grid for all seniorities.

5.7.15 FXSpots

Listing 132 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 132: FXSpot Configuration

5.7.16 Securities

Listing 133 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.
- an optional ConversionFactor. In case of a bond future, a conversion factor might be applied. The application requires a clean price settlement. Otherwise the default value of 1.0 for the conversion factor will be used.
- an optional PriceQuote. If the bond spread imply analytics is available, this is used to imply a SpreadQuote to match a given bond price. Notice that if a SpreadQuote is given explicitly in the market data, this will override an implied spread.

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 sensitivity and stress analysis.

Note: in case of a forward bond, the convention for security specification requires the format *securityId_FWDEXP_expiryDate.*

```
<Securities>
  <Security>
```

```
    <CurveId>SECURITY_1</CurveId>
    <CurveDescription>Security</CurveDescription>
    <SpreadQuote>BOND/YIELD_SPREAD/SECURITY_1</SpreadQuote>
    <RecoveryRateQuote>RECOVERY_RATE/RATE/SECURITY_1</RecoveryRateQuote>
    <PriceQuote>BOND/PRICE/SECURITY_1</PriceQuote>
    <ConversionFactor>BOND/CONVERSION_FACTOR/SECURITY_1</ConversionFactor>
  </Security>
</Securities>
```

Listing 133: Security Configuration

5.7.17 Correlations

Listing 134 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 separated 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 134: Correlation Configuration

5.7.18 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 135 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 135: Commodity Curve Configuration for WTI Oil

Listing 136 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 136: 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, ForwardFlat. 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 137. 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 137: 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 138 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 139 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 138 and 139.
- 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 138: 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>
    <PriceAsHistoricalFixing>true</PriceAsHistoricalFixing>
  </BasisConfiguration>
  <Extrapolation>true</Extrapolation>
</CommodityCurve>
```

Listing 139: Commodity curve configuration for WTI Midland (Argus)

The meaning of the fields in the `BasisConfiguration` node in Listings 138 and 139 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.
- `PriceAsHistoricalFixing` [Optional]: This is a boolean flag where `true`, the default value, indicates that the historical fixings are prices of the underlying. If set to false, the fixings are basis spreads and ORE will convert them into prices by adding the corresponding base index fixings.

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 140.

```
<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 140: Commodity curve configuration for PJM Real Time Peak

The `BootstrapConfig` node is described in Section 5.7.20. The remaining nodes in Listing 140 have been described already in this section. The meaning of each of the fields in the `PriceSegment` node in Listing 140 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 5.11.22.
- `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 140 or a list of explicit COMMODITY_FWD PRICE quote strings.

5.7.19 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 141 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 141: 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 142 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 142: 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`. `Normal`, `ShiftedLognormal` and `Lognormal` are supported for constant and absolute strike surfaces. For all other types only `Lognormal` is supported. `Lognormal` 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 9.28. 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 9.28, 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 143 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 143: 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 9.28, 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. The allowable values for `TimeInterpolation` are `Linear` and `Cubic`. If neither `Strikes` nor `Expiries` use the single wildcard value `*`, `TimeInterpolation` and `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. If either `Strikes` or `Expiries` use the single wildcard value `*`, `TimeInterpolation` and `StrikeInterpolation` can be different. For backward compatibility, if either `Strikes` or `Expiries` use the single wildcard value `*`, `TimeInterpolation` and `StrikeInterpolation` will use Linear as default value or when input values are not in allowable list. 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`: Defines how to extrapolate in the time direction. Allowed values are:

  - `None`: No extrapolation.
  - `Flat`: Keeps the volatility constant beyond the known range.
  - `UseInterpolator`: Extends the configured interpolation (linear or cubic) into the extrapolated range.
  - `Linear`: Legacy identifier and falls back to `UseInterpolator`, but only for backward compatibility. Prefer `UseInterpolator` for clarity.

  Notes:

  - Variance extrapolation works with linear and cubic interpolation.
  - Volatility extrapolation only works with linear interpolation.

- `TimeExtrapolationVariance`: Specifies whether to extrapolate variance or volatility in the time direction. Allowed values are:

  - `True`: Extrapolates variance (default if omitted).
  - `False`: Extrapolates volatility.

  Notes:

  - Ignored if TimeExtrapolation is set to Flat.
  - Volatility extrapolation in time works only with linear interpolation.

- 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 143 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.

- 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 144 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 144: 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 must be 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`: Defines how to extrapolate in the time direction. Allowed values are:
  - `None`: No extrapolation.
  - `Flat`: Keeps the volatility constant beyond the known range.
  - `UseInterpolator`: Extends the configured interpolation (linear) into the extrapolated range.
  - `Linear`: Legacy identifier and falls back to `UseInterpolator`, but only for backward compatibility. Prefer `UseInterpolator` for clarity.
- `TimeExtrapolationVariance`: Specifies whether to extrapolate variance or volatility in the time direction. Allowed values are:
  - `True`: Extrapolates variance (default if omitted).
  - `False`: Extrapolates volatility.

  Notes:
  - Ignored if `TimeExtrapolation` is set to `Flat`.
  - Volatility extrapolation in time works only with linear interpolation.
- `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 145 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 145: 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. The delta should match exactly the quote i.e 0.1 != 0.10
- CallDeltas: A comma separated list of one or more call deltas to use in the volatility surface. The delta should match exactly the quote i.e 0.1 != 0.10
- Expiries: A comma separated list of one or more expiries (e.g. 1W, 1M) to load. Supports using the single wildcard value *.
- TimeInterpolation: Only Linear is currently supported here.
- StrikeInterpolation: Allowable values are Linear, NaturalCubic, FinancialCubic and CubicSpline.
- Extrapolation: A boolean value indicating if extrapolation is allowed.
- TimeExtrapolation: Defines how to extrapolate in the time direction. Allowed values are:
  - None: No extrapolation.
  - Flat: Keeps the volatility constant beyond the known range.
  - UseInterpolator: Extends the configured interpolation (linear) into the extrapolated range.
  - Linear: Legacy identifier and falls back to UseInterpolator, but only for backward compatibility. Prefer UseInterpolator for clarity.
- TimeExtrapolationVariance: Specifies whether to extrapolate variance or volatility in the time direction. Allowed values are:
  - True: Extrapolates variance (default if omitted).
  - False: Extrapolates volatility.

  Notes:
  - Ignored if TimeExtrapolation is set to Flat.
  - Volatility extrapolation in time works only with linear interpolation.
- 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 146 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 146 shows the configuration used in this context.

```
<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 146: 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.
- `TimeExtrapolation`: Defines how to extrapolate in the time direction. Allowed values are:
  - `None`: No extrapolation.
  - `Flat`: Keeps the volatility constant beyond the known range.
  - `UseInterpolator`: Extends the configured interpolation (linear) into the extrapolated range.
  - `Linear`: Legacy identifier and falls back to `UseInterpolator`, but only for backward compatibility. Prefer `UseInterpolator` for clarity.
- `TimeExtrapolationVariance`: Specifies whether to extrapolate variance or volatility in the time direction. Allowed values are:
  - `True`: Extrapolates variance (default if omitted).
  - `False`: Extrapolates volatility.

  Notes:
  - Ignored if `TimeExtrapolation` is set to `Flat`.
  - Volatility extrapolation in time works only with linear interpolation.
- `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.

5.7.20 Bootstrap Configuration

The `BootstrapConfig` node, outlined in listing 147, 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>
  <Global>...<Global>
</BootstrapConfig>
```

Listing 147: `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 (iterative bootstrap) or the accuracy of the global optimizer (global 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.
- `Global` [Optional]: Defaults to false. This nodes specifies whether the curve should use a global bootstrap over all instruments of the curve (true) or iterative bootstrap (false, default value). If global bootstrap is used, only the fields Accuracy and SmoothnessLambda of the BootstrapConfig are relevant. Accurcy specifies the accuracy of the global optimizer in this case. Global bootstrap is required to build curves that form a cycle in the dependency graph, in this case all members of the cycle must have Global set to true in their bootstrap configs. In this case a global optimization is run over all instrument of all members of the cycle simultaneously. Global bootstrap is also required if according to the PillarChoice configuration of each segment there are curve instrument without or with more than one associated curve pillar date. Finally, global bootstrap can be preferable over iterative bootstrap if the latter requires an outer convergence loop, i.e. if a pillar choice is not set to LastRelevatDate or if the interpolation is non-local.
- `SmoothnessLambda` [Optional]: Defaults to zero. Only applies if GLobal is set to true. If positive, penalty components are included in the global optimization used to build a curve. For each adjacent pair of discrete forward rates F_i, F_{i+1} between curve interpolation pillars, the term $\lambda \cdot (F_{i+1} - F_i)$ is added to the target

function vector of which the MSE is optimized.

5.7.21 One Dimensional Solver Configuration

The OneDimSolverConfig node, outlined in Listing 148, 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.

```
<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 148: 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`.

Method	**Description**
Linear	Linear interpolation
LogLinear	Linear interpolation on the natural log of the interpolation variable
NaturalCubic	Monotonic Kruger cubic interpolation with zero second derivative at left and right
FinancialCubic	Monotonic Kruger cubic interpolation with zero second derivative at left and zero first derivative at right
ConvexMonotone	Convex Monotone Interpolation (Hagan, West)
Quadratic	Quadratic interpolation
LogQuadratic	Quadratic interpolation on the natural log of the interpolation variable
LogNaturalCubic	Monotonic Kruger cubic interpolation with zero second derivative at left and right
LogFinancialCubic	Monotonic Kruger cubic interpolation with zero second derivative at left and zero first derivative at right
LogCubicSpline	Non-monotonic cubic spline interpolation with zero second derivative at left and right
MonotnicLogCubicSpline	Monotonic cubic spline interpolation with zero second derivative at left and right
Hermite	Hermite cubic spline interpolation
CubicSpline	Non-monotonic cubic spline interpolation with zero second derivative at left and right
DefaultLogMixedLinearCubic	Mixed interpolation, first linear, then monotonic Kruger cubic spline
MonotonicLogMixedLinearCubic	Mixed interpolation, first linear, then monotonic natural cubic spline
KrugerLogMixedLinearCubic	Mixed interpolation, first linear, then non-monotonic Kruger cubic spline
LogMixedLinearCubicNaturalSpline	Mixed interpolation, first linear, then non-monotonic natural cubic spline
BackwardFlat	Backward-flat interpolation (piecewise constant, right-continuous)
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 18: Allowable interpolation methods.

5.8 Reference Data `referencedata.xml`

Reference Data is used to ease the burden on portfolio xml representation (see [1]), 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 5.7.1.

The allowable types for ReferenceData is

1. **Bond** static data consists of the Leg data for a given bond.
2. `SubType` has been added for reporting purposes, to feed into the ISDA product taxonomy, without impact on pricing.
   Valid `SubTypes` are:
   - ABS, Corp(orate), Loans, Muni, Sovereign
   - ABX, CMBX, MBX, PrimeX, TRX, iBoxx (in case the Bond represents a Credit or Bond index)

   Note that the SubType field is currently optional and not covered by schema checks.

```
<ReferenceData>
<!-- Bond reference datum -->
<ReferenceDatum id="SECURITY_1">
  <Type>Bond</Type>
  <BondReferenceData>
    <SubType>Muni</SubType>
    <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>
```

5.9 Ibor Fallback Config: `iborFallbackConfig.xml`

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>
        ....
```

5.10 SIMM Calibration: `simmcalibration.xml`

The SIMM Calibration can be used to add or override SIMM versions by specifying the risk weights, correlations, concentration thresholds along with associated buckets/labels and currency groups (for risk class FX and IR).

See Example_44 for a full SIMM calibration file for SIMM 2.5A [21]. The official configuration files for each version (SIMM 2.2 and greater) can be found in folder `Configurations/SIMM/`

The file consists of a `<SIMMCalibrationData>` node, with `<SIMMCalibration>` subnodes that each define a given SIMM version, as in Listing 149 below.

Listing 149: SIMM Calibration data

```
<SIMMCalibrationData>
  <SIMMCalibration id="official">
    <VersionNames>
      <Name>2.6</Name>
      <Name>2.5.6</Name>
    </VersionNames>
    <AdditionalFields>
      <SIMM_EffectiveDate>2023-12-02</SIMM_EffectiveDate>
    </AdditionalFields>
    <InterestRate>
      <RiskWeights>
        ......
      </RiskWeights>
      <Correlations>
        ......
      </Correlations>
      <ConcentrationThresholds>
        ......
      </ConcentrationThresholds>
    </InterestRate>
    <CreditQualifying>
      ......
    </CreditQualifying>
    <CreditNonQualifying>
      ......
    </CreditNonQualifying>
    <Equity>
      ......
    </Equity>
    <Commodity>
      ......
    </Commodity>
    <FX>
      ......
    </FX>
    <RiskClassCorrelations>
      ......
    </RiskClassCorrelations>
  </SIMMCalibration>
  <SIMMCalibration id="next">
    ......
  </SIMMCalibration>
</SIMMCalibrationData>
```

5.10.1 SIMM Calibration

A SIMM Calibration is defined by a `<SIMMCalibration>` node that defines a particular SIMM version, i.e. it defines a single set of risk weights, correlations, concentration thresholds and currency groups. The `<SIMMCalibration>` has the following components:

1. Version names - `<VersionNames>`
   This may contain any number of `<Name>` sub-nodes, where each value will be associated with the given SIMM calibration. In order to use a given calibration,

one of its names must be specified in the “version” parameter of the SIMM analytic (see Listing 26). In the example listing 149 above, the SIMM calibration will override the original SIMM 2.5.6/2.6 defined in the source code.

2. Additional fields - `<AdditionalFields>`
   This node is used for purely descriptive purposes and can contain any subnode.
3. Risk-class-specific sub-nodes:
   - `<InterestRate>`
   - `<CreditQualifying>`
   - `<CreditNonQualifying>`
   - `<Equity>`
   - `<Commodity>`
   - `<FX>`
4. Risk class correlations - `<RiskClassCorrelations>`

The risk class correlations and its subcomponents are given in Listing 150:

Listing 150: SIMM Calibration: risk class correlations

```
<RiskClassCorrelations>
  <Correlation label1="InterestRate" label2="FX">0.14</Correlation>
  <Correlation label1="InterestRate" label2="Equity">0.29</Correlation>
  <Correlation label1="InterestRate" label2="CreditQualifying">0.27</Correlation>
  <Correlation label1="InterestRate" label2="CreditNonQualifying">0.26</Correlation>
  <Correlation label1="InterestRate" label2="Commodity">0.31</Correlation>
  <Correlation label1="FX" label2="InterestRate">0.14</Correlation>
  <Correlation label1="FX" label2="Equity">0.25</Correlation>
  <Correlation label1="FX" label2="CreditQualifying">0.25</Correlation>
  <Correlation label1="FX" label2="CreditNonQualifying">0.14</Correlation>
  <Correlation label1="FX" label2="Commodity">0.3</Correlation>
  <Correlation label1="Equity" label2="InterestRate">0.29</Correlation>
    .......
</RiskClassCorrelations>
```

Each correlation value is given by a `<Correlation>` node with attributes `label1` and `label2` to specify the risk classes to which it applies.

Since the risk-class-specific components have many sub-nodes in common, the following section will be a description of the general ‘base’ structure, and then a section will be given for each risk class to explain its specific XML structure along with any components unique to that risk class. We will also make reference to the corresponding sections in the ISDA SIMM Methodology [21].

5.10.2 General Structure

All risk class subnodes (i.e. `InterestRate`, `CreditQualifying`, etc.) in the SIMM calibration will contain the following three components:

1. `<RiskWeights>`

2. `<Correlations>`
3. `<ConcentrationThresholds>`

The general structure is given by Listing 151. Note that `<HistoricalVolatilityRatio>` only applies to InterestRate, Equity, Commodity and FX, and `<CurrencyLists>` only applies to InterestRate and FX.

Listing 151: SIMM Calibration - General XML Structure

```
<RiskWeights>
  <Delta mporDays="10">
      ....
  </Delta>
  <Vega mporDays="10">
      ....
  </Vega>
  <HistoricalVolatilityRatio mporDays="10">...</HistoricalVolatilityRatio>
</RiskWeights>
<Correlations>
  <IntraBucketCorrelation>
      ....
  </IntraBucketCorrelation>
  <InterBucketCorrelation>
      ....
  </InterBucketCorrelation>
</Correlations>
<ConcentrationThresholds>
  <Delta>
      ....
  </Delta>
  <Vega>
      ....
  </Vega>
  <CurrencyLists>
      ....
  </CurrencyLists>
</ConcentrationThresholds>
```

The structure of the `<RiskWeights>` node is given by Listing 152. Every top-level node in `<RiskWeights>` should have an `<mporDays>` attribute (which defaults to *"10"* when omitted).

Listing 152: SIMM Calibration - Risk Weights

```
<RiskWeights>
  <Delta mporDays="10">
    <!-- e.g. for IR -->
    <Weight bucket="1" label1="2w">109</Weight>
    <Weight bucket="1" label1="1m">105</Weight>

    <!-- e.g. for CreditQualifying/CreditNonQualifying/Equity/Commodity -->
    <Weight bucket="1">75</Weight>
    <Weight bucket="2">90</Weight>

    <!-- e.g. for FX -->
    <Weight label1="2" label2="2">7.4</Weight>
    <Weight label1="2" label2="1">14.7</Weight>
  </Delta>
  <HistoricalVolatilityRatio mporDays="10">0.47</HistoricalVolatilityRatio>
  <Vega mporDays="10">
    <!-- e.g. for IR/CreditQualifying/Commodity/FX -->
    <Weight>0.76</Weight>

    <!-- e.g. for CreditNonQualifying/Equity -->
    <Weight bucket="1">280</Weight>
    <Weight bucket="2">1300</Weight>
  </Vega>
</RiskWeights>
```

The structure of the <Correlations> node is given by Listing 153.

Listing 153: SIMM Calibration - Correlations

```
<Correlations>
  <IntraBucketCorrelation>
    <!-- e.g. for IR -->
    <Correlation label1="2w" label2="1m">0.77</Correlation>
    <Correlation label1="2w" label2="3m">0.67</Correlation>

    <!-- e.g. for CreditQualifying/CreditNonQualifying -->
    <Correlation label1="aggregate" label2="same">0.93</Correlation>
    <Correlation label1="aggregate" label2="different">0.46</Correlation>

    <!-- e.g. for Equity/Commodity -->
    <Correlation bucket="1">0.18</Correlation>
    <Correlation bucket="2">0.2</Correlation>

    <!-- e.g. for FX -->
    <Correlation bucket="2" label1="2" label2="2">0.5</Correlation>
    <Correlation bucket="2" label1="2" label2="1">0.25</Correlation>
  </IntraBucketCorrelation>
  <InterBucketCorrelation>
    <!-- e.g. for CreditQualifying/CreditNonQualifying/Equity/Commodity -->
    <Correlation label1="1" label2="2">0.38</Correlation>
    <Correlation label1="1" label2="3">0.38</Correlation>
  </InterBucketCorrelation>
</Correlations>
```

The structure of the `<ConcentrationThresholds>` node is given by Listing 154.

Listing 154: SIMM Calibration - Concentration Thresholds

```
<ConcentrationThresholds>
  <Delta>
    <!-- e.g. for IR/CreditQualifying/CreditNonQualifying/Equity/Commodity/FX -->
    <Threshold bucket="1">30</Threshold>
    <Threshold bucket="2">330</Threshold>
  </Delta>
  <Vega>
    <!-- e.g. for IR -->
    <Threshold bucket="1">74</Threshold>
    <Threshold bucket="2">4900</Threshold>

    <!-- e.g. for CreditQualifying/CreditNonQualifying/Equity/Commodity/FX -->
    <Threshold>360</Threshold>
  </Vega>
</ConcentrationThresholds>
```

5.10.3 Interest Rate

The structure for the `<InterestRate>` node is given by Listing 155.

Listing 155: SIMM Calibration - Interest Rate Risk

```
<InterestRate>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
    <HistoricalVolatilityRatio mporDays="10">0.47</HistoricalVolatilityRatio>
    <Inflation mporDays="10">61</Inflation>
    <XCcyBasis mporDays="10">21</XCcyBasis>
    <CurrencyLists>
      <Currency bucket="1">USD</Currency>
        ....
      <Currency bucket="3">Other</Currency>
    </CurrencyLists>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <SubCurves>0.993</SubCurves>
    <Inflation>0.24</Inflation>
    <XCcyBasis>0.04</XCcyBasis>
    <Outer>0.32</Outer>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
    <CurrencyLists>
      <Currency bucket="1">Other</Currency>
      <Currency bucket="2">USD</Currency>
        ....
    </CurrencyLists>
  </ConcentrationThresholds>
</InterestRate>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section D.1, 33.
  Each `<Weight>` node must have a `bucket` (defining the currency group) and `label1` (defining the tenor) attribute.
  Allowable `bucket` values: *"1"* for regular volatility currencies, *"2"* for low-volatility currencies, and *"3"* for high-volatility currencies.
  Allowable `label1` values: *"2w", "1m", "3m", "6m", "1y", "2y", "3y", "5y", "10y", "15y", "20y", "30y"*.
- Vega risk weight: Section D.1, 35.
  There should only be one `<Weight>` node, and no attributes are required.
- Historical volatility ratio (HVR): Section D.1 34.
  There should only be one `<HistoricalVolatilityRatio>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).
- Inflation risk weight: Section D.1, 33.
  There should only be one `<Weight>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).
- Risk weight for cross-currency basis swap spread (`<XCcyBasis>`): Section D.1, 33
  There should only be one `<Weight>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).
- Currency groups for risk weights (`<CurrencyLists>`): Section D.1 33(1) to (3).
  Each `<Currency>` must have a `bucket` attribute associating it with a currency volatility group.
- Intra-bucket correlations: Section D.2, 36.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.
- Correlation between sub-curves: Section D.2, 36.
- Correlation for inflation rate: Section D.2, 36.
- Correlation for Cross-currency basis swap spread (`XCcyBasis`): Section D.2, 36.
- Correlation across different currencies (`<Outer>`): Section D.2, 37.
- Delta concentration thresholds: Section J.1, 74
  Each `<Threshold>` must have a `bucket` attribute associating it with a currency group.
  Allowable `bucket` values: *"1"* for high volatility, *"2"* for regular volatility, well-traded, *"3"* for regular volatility, less-traded, and *"4"* for low volatility.
- Vega concentration thresholds: Section J.6, 81.
  Each `<Threshold>` must have a `bucket` attribute associating it with a currency group.
  Allowable `bucket` values: *"1"* for high volatility, *"2"* for regular volatility, well-traded, *"3"* for regular volatility, less-traded, and *"4"* for low volatility.

- Concentration threshold currency groups (`CurrencyLists`): Section J.1, 75.
  Each `<Currency>` must have a `bucket` attribute associating it with a currency group.
  Allowable `bucket` values: *"1"* for high volatility, *"2"* for regular volatility, well-traded, *"3"* for regular volatility, less-traded, and *"4"* for low volatility.

5.10.4 Credit Qualifying

The structure for the `<CreditQualifying>` node is given by Listing 156.

Listing 156: SIMM Calibration - Credit Qualifying Risk

```
<CreditQualifying>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
    <BaseCorrelation mporDays="10">10</BaseCorrelation>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <InterBucket>...</InterBucket>
    <BaseCorrelation>...</BaseCorrelation>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
  </ConcentrationThresholds>
</CreditQualifying>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section E.1, 39.
  Each `<Weight>` node must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "Residual"*, as defined in Section E.1, 38.

- Vega risk weight: Section E.1, 40.
  There should only be one `<Weight>` node, and no attributes are required.

- Base correlation risk: Section E.1 41.

- Intra-bucket correlations: Section E.2, 42.
  Each `<Correlation>` must have a `label1` and `label2` attribute.
  Allowable `label1` values: *aggregate* or *residual*.
  Allowable `label2` values: *same* or *different*.

- Inter-bucket correlations: Section E.2, 43.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.
  Allowable `label1`/`label2` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"*, as defined in Section E.1, 38.

- Correlation for Base correlation risks: Section E.2, 42.
- Delta concentration thresholds: Section J.2, 76
  Each `<Threshold>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "Residual"*, as defined in Section E.1, 38.
- Vega concentration threshold: Section J.7, 83.
  There should only be one `<Threshold>`, and no attributes are required.

5.10.5 Credit Non-Qualifying

The structure for the `<CreditNonQualifying>` node is given by Listing 157.

Listing 157: SIMM Calibration - Credit Non-Qualifying Risk

```
<CreditNonQualifying>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <InterBucket>...</InterBucket>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
  </ConcentrationThresholds>
</CreditNonQualifying>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section F.1, 46.
  Each `<Weight>` node must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "Residual"*, as defined in Section F.1, 45.
- Vega risk weight: Section F.1, 47.
  There should only be one `<Weight>` node, and no attributes are required.
- Intra-bucket correlations: Section F.2, 48.
  Each `<Correlation>` must have a `label1` and `label2` attribute.
  Allowable `label1` values: *aggregate* or *residual*.
  Allowable `label2` values: *same* or *different*.
- Inter-bucket correlations: Section F.2, 49.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.
  Allowable `label1`/`label2` values: *"1", "2"*, as defined in Section F.1, 45.
- Delta concentration thresholds: Section J.2, 76
  Each `<Threshold>` must have a `bucket` attribute.

Allowable `bucket` values: *"1"*, *"2"*, *"Residual"*, as defined in Section F.1, 45.

- Vega concentration threshold: Section J.7, 83.
  There should only be one `<Threshold>`, and no attributes are required.

5.10.6 Equity

The structure for the `<Equity>` node is given by Listing 158.

Listing 158: SIMM Calibration - Equity Risk

```
<Equity>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
    <HistoricalVolatilityRatio mporDays="10">0.6</HistoricalVolatilityRatio>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <InterBucket>...</InterBucket>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
  </ConcentrationThresholds>
</Equity>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section G.1, 56.
  Each `<Weight>` node must have a `bucket` attribute.
  Allowable `bucket` values: *"1"*, *"2"*, *"3"*, *"4"*, *"5"*, *"6"*, *"7"*, *"8"*, *"9"*, *"10"*, *"11"*, *"12"*, *"Residual"*, as defined in Section G.1, 50.

- Vega risk weight: Section G.1, 58.
  Each `<Weight>` node must have a `bucket` attribute.
  Allowable `bucket` values: *"1"*, *"2"*, *"3"*, *"4"*, *"5"*, *"6"*, *"7"*, *"8"*, *"9"*, *"10"*, *"11"*, *"12"*, *"Residual"*, as defined in Section G.1, 50.

- Historical volatility ratio (HVR): Section G.1 57.
  There should only be one `<HistoricalVolatilityRatio>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).

- Intra-bucket correlations: Section G.2, 59.
  Each `<Correlation>` must have a `bucket` attribute. Allowable `bucket` values: *"1"*, *"2"*, *"3"*, *"4"*, *"5"*, *"6"*, *"7"*, *"8"*, *"9"*, *"10"*, *"11"*, *"12"*, *"Residual"*, as defined in Section G.1, 50.

- Inter-bucket correlations: Section G.2, 60.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.

Allowable `label1`/`label2` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"*, as defined in Section G.1, 50.

- Delta concentration thresholds: Section J.3, 77
  Each `<Threshold>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "Residual"*, as defined in Section G.1, 50.
- Vega concentration thresholds: Section J.8, 84.
  Each `<Threshold>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "Residual"*, as defined in Section G.1, 50.

5.10.7 Commodity

The structure for the `<Commodity>` node is given by Listing 159.

Listing 159: SIMM Calibration - Commodity Risk

```
<Commodity>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
    <HistoricalVolatilityRatio mporDays="10">0.74</HistoricalVolatilityRatio>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <InterBucket>...</InterBucket>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
  </ConcentrationThresholds>
</Commodity>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section H.1, 61.
  Each `<Weight>` node must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17"*, as defined in Section H.1, 61.
- Vega risk weight: Section H.1, 63.
  There should only be one `<Weight>` node, and no attributes are required.
- Historical volatility ratio (HVR): Section H.1 62.
  There should only be one `<HistoricalVolatilityRatio>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).
- Intra-bucket correlations: Section H.2, 64.
  Each `<Correlation>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17"*, as defined in Section H.1, 61.

- Inter-bucket correlations: Section H.2, 65.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.
  Allowable `label1`/`label2` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17"*, as defined in Section H.1, 61.
- Delta concentration thresholds: Section J.4, 78.
  Each `<Threshold>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17"*, as defined in Section H.1, 61.
- Vega concentration thresholds: Section J.9, 85.
  Each `<Threshold>` must have a `bucket` attribute.
  Allowable `bucket` values: *"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17"*, as defined in Section H.1, 61.

5.10.8 FX

The structure for the `<FX>` node is given by Listing 160.

Listing 160: SIMM Calibration - FX Risk

```xml
<FX>
  <RiskWeights>
    <Delta mporDays="10">...</Delta>
    <Vega mporDays="10">...</Vega>
    <HistoricalVolatilityRatio mporDays="10">0.57</HistoricalVolatilityRatio>
    <CurrencyLists>
      <Currency bucket="2">Other</Currency>
      <Currency bucket="1">BRL</Currency>
      <Currency bucket="1">TRY</Currency>
      <Currency bucket="1">RUB</Currency>
    </CurrencyLists>
  </RiskWeights>
  <Correlations>
    <IntraBucket>...</IntraBucket>
    <Volatility>0.5</Volatility>
  </Correlations>
  <ConcentrationThresholds>
    <Delta>...</Delta>
    <Vega>...</Vega>
    <CurrencyLists>
      <Currency bucket="1">USD</Currency>
      <Currency bucket="1">EUR</Currency>
        ....
      <Currency bucket="3">Other</Currency>
    </CurrencyLists>
  </ConcentrationThresholds>
</FX>
```

The above values are found in the following sections of the ISDA SIMM Methodology [21]:

- Delta risk weights: Section I.1, 69.
  Each `<Weight>` node must have a `label1` (defining the first currency's volatility group) and `label2` (defining the second currency's volatility group) attribute.
  Allowable `label1`/`label2` values: *"1"* for high FX volatility currencies and *"2"* for regular FX volatility currencies.
- Vega risk weight: Section I.1, 71.
  There should only be one `<Weight>` node, and no attributes are required.
- Historical volatility ratio (HVR): Section I.1 70.
  There should only be one `<HistoricalVolatilityRatio>` node, and the only attributed needed is `mporDays` (which defaults to *"10"* if omitted).
- Currency groups for risk weights (`<CurrencyLists>`): Section I.1 67. and 68.
  Each `<Currency>` must have a `bucket` attribute associating it with a currency volatility group.
  Allowable `bucket` values: Allowable `label1`/`label2` values: *"1"* for high FX volatility currencies and *"2"* for regular FX volatility currencies.
- Intra-bucket correlations: Section I.2, 72.
  Each `<Correlation>` must have a `label1` and `label2` attribute. Note that although the correlations are symmetric, the correlation value for both directions must be provided.
  Allowable `bucket` values: *"1"* if the SIMM calculation currency is in the regular FX volatility group, and *"2"* if it is in the high FX volatility group.
  Allowable `label1`/`label2` values: *"1"* for high FX volatility currencies and *"2"* for regular FX volatility currencies.
- Correlation between FX volatility and curvature risk factors (`<Volatility>`): Section I.2. 73
- Delta concentration thresholds: Section J.5, 79
  Each `<Threshold>` must have a `bucket` attribute associating it with a currency risk group.
  Allowable `bucket` values: *"1", "2", "3"*.
- Vega concentration thresholds: Section J.10, 86.
  Each `<Threshold>` must have a `bucket` attribute associating it with a currency group.
  Allowable `bucket` values: *"1"* for "Category 1 - Category 1", *"2"* for "Category 1 - Category 2", *"3"* for "Category 1 - Category 3", *"4"* for "Category 2 - Category 2", *"5"* for "Category 2 - Category 3", and *"6"* for "Category 3 - Category 3".
- Concentration threshold currency groups (`CurrencyLists`): Section J.5, 80.
  Each `<Currency>` must have a `bucket` attribute associating it with a currency risk group.
  Allowable `bucket` values: *"1"* for "Category 1 - Significantly material", *"2"* for "Category 2 - Frequently traded", and *"3"* for "Category 3 - Others".

5.11 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.

5.11.1 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 161. The node for a tenor based zero rate is shown in Listing 162.

Listing 161: Zero conventions

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

Listing 162: 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 7.3)
- 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 7.3).
- 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.

5.11.2 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 163. Alternatively, all the necessary conventions can be given explicitly without reference to an index family. The structure of this node is shown in Listing 164.

Listing 163: Deposit conventions

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

Listing 164: 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.

5.11.3 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 165. 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, 'FirstDayOfMonth' when the start and end date are the first day of a month, or 'SecondThursday' when the expiry date is the second Thursday of the month. If not given this field defaults to 'IMM'.
  - For MM futures 'IMM' or 'SecondThursday' are allowed. The expiry date for 'IMM' is determined as the next 3rd Wednesday of the expiry month of a future. The expiry date for 'SecondThursday' is determined as the 2nd Thursday of the expiry month (e.g. used for AUD-BBSW-3M futures).
  - 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. For 'FirstDayOfMonth' the start and end date are adjusted to the next business day w.r.t. roll convention 'following' and the overnight index calendar - or if given the calendar given in the field 'Calendar'. This tenor is derived from the market quote, see 9.8.
- Calendar: Only applicable to overnight index futures with date generation rule 'FirstDayOfMonth'. Overwrites the calendar used to adjust the start and end date of the future. Optional, if not given, the overnight index publishing calendar will be used to adjust the future start and end date.

- 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 166, 167, 168, 169 show examples for Euribor-3M, AUD-BBSW-3M, SOFR-3M and SOFR-1M future conventions.

Listing 165: Future conventions

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

Listing 166: Euribor 3M MM Future conventions

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

Listing 167: AUD BBSW 3M MM Future conventions (Second Thursday expiry)

```
<Future>
  <Id>AUD-BBSW-3M-FUTURES-CONVENTIONS</Id>
  <Index>AUD-BBSW-3M</Index>
  <DateGenerationRule>SecondThursday</DateGenerationRule>
</Future>
```

Listing 168: 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 169: USD SOFR 1M OI Future conventions

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

5.11.4 FRA Conventions

A node with name *FRA* is used to store conventions for FRA quotes. The structure of this node is shown in Listing 170. 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 170: FRA conventions

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

5.11.5 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 171.

Listing 171: OIS conventions

```
<OIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <Index> </Index>
  <FixedDayCounter> </FixedDayCounter>
  <FixedCalendar> </FixedCalendar>
  <PaymentLag> </PaymentLag>
  <EOM> </EOM>
  <FixedFrequency> </FixedFrequency>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <Rule> </Rule>
  <PaymentCalendar> </PaymentCalendar>
  <RateCutoff> </RateCutoff>
</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.
- FixedCalendar [Optional]: The business day calendar on the fixed leg. Optional to retain backwards compatibility with older versions, if not given defaults to index fixing calendar.
- 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.
- 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 end of the interest period date. The last observed rate is applied for the remaining days in the period. This rate cut-off gives the number of days e.g. 1 for ESTR or SOFR. If not specified, this defaults to 0 days.

5.11.6 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 172.

Listing 172: 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".

5.11.7 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 173.

Listing 173: Average OIS conventions

```
<AverageOIS>
  <Id> </Id>
  <SpotLag> </SpotLag>
  <FixedTenor> </FixedTenor>
  <FixedDayCounter> </FixedDayCounter>
  <FixedCalendar> </FixedCalendar>
  <FixedConvention> </FixedConvention>
  <FixedPaymentConvention> </FixedPaymentConvention>
  <FixedFrequency> </FixedFrequency>
  <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.
- FixedFrequency [Optional]: The frequency of payments on the fixed leg. If not provided, this defaults to *Annual*.
- 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.

5.11.8 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 174.

Listing 174: Tenor basis swap conventions

```
<TenorBasisSwap>
  <Id> </Id>
  <PayIndex> </PayIndex>
  <PayFrequency> </PayFrequency>
  <ReceiveIndex> </ReceiveIndex>
  <ReceiveFrequency> </ReceiveFrequency>
  <SpreadOnRec> </SpreadOnRec>
  <IncludeSpread> </IncludeSpread>
  <SubPeriodsCouponType> </SubPeriodsCouponType>
</TenorBasisSwap>
```

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

- PayIndex: The name of Ibor/Overnight Index of the pay leg.
- PayFrequency [Optional]: The frequency of payments on the PayIndex leg. This is usually the same as the PayIndex's tenor. However, it can also be longer, e.g. overnight indexed vs overnight indexed basis swaps that may be quarterly on both legs. If not provided, this defaults to the PayIndex's tenor.
- ReceiveIndex: The name of Ibor/Overnight Index of the receive leg.
- ReceiveFrequency [Optional]: The frequency of payments on the ReceiveIndex leg. This is usually the same as the ReceiveIndex's tenor. However, it can also be longer, e.g. overnight indexed vs overnight indexed basis swaps that may be quarterly on both legs. If not provided, this defaults to the ReceiveIndex's tenor.
- SpreadOnRec [Optional]: *True* if the tenor basis swap quote has the spread on the pay 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 spread leg and *False* if not. If not provided, this defaults to *False*.
- SubPeriodsCouponType [Optional]: This field can have the value *Compounding* or *Averaging*. It applies to Ibor vs OI and Ibor vs Ibor basis swaps when the frequency of payments on the spread leg does not equal the spread leg index's tenor. If *Compounding* is specified, then the spread tenor Ibor index is compounded and paid on the frequency specified in the corresponding node. If *Averaging* is specified, then the short tenor Ibor index is averaged and paid on the frequency specified in the corresponding node.

5.11.9 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 175.

Listing 175: 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*.

5.11.10 BMA / SIFMA Basis Swap Conventions

A node with name *BMABasisSwap* is used to store conventions for BMA / SIFMA Basis Swap quotes. The structure of this node is shown in Listing 171.

Listing 176: BMA / SIFMA Basis conventions

```
<BMABasisSwap>
  <Id> </Id>
  <Index> </Index>
  <BMAIndex> </BMAIndex>
  <BMAPaymentCalendar> </BMAPaymentCalendar>
  <BMAPaymentConvention> </BMAPaymentConvention>
  <BMAPaymentLag> </BMAPaymentLag>
  <IndexPaymentCalendar> </IndexPaymentCalendar>
  <IndexPaymentConvention> </IndexPaymentConvention>
  <IndexPaymentLag> </IndexPaymentLag>
  <IndexSettlementDays> </IndexSettlementDays>
  <IndexPaymentPeriod> </IndexPaymentPeriod>
  <OvernightLockoutDays> </OvernightLockoutDays>
</BMABasisSwap>
```

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

- Index: The name of an ibor, term rate or overnight index, e.g. USD-LIBOR-3M, USD-SOFR-3M, USD-SOFR
- BMAIndex: The name of a BMA / SIFMA index, e.g. USD-SIFMA
- BmaPaymentCalendar [Optional]: The payment calendar of the BMA / SIFMA leg. If not specified, defaults to BMA index fixing calendar.
- BmaPaymentConvention [Optional]: The payment convention of the BMA / SIFMA leg. If not specified, defaults to Following.
- BmaPaymentLag [Optional]: The payment lag of the BMA / SIFMA leg. If not specified, defaults to 0.
- IndexPaymentCalendar [Optional]: The payment calendar of the reference leg. If not specified, defaults to index fixing calendar.
- IndexPaymentConvention [Optional]: The payment convention of the reference leg. If not specified, defaults to Following.
- IndexPaymentLag [Optional]: The payment lag of the reference leg. If not specified, defaults to 0.
- IndexSettlementDays [Optional]: The settlement days of the reference leg. If not specified, defaults to bma index fixing days.
- IndexPaymentPeriod [Optional]: The payment period of the reference leg. If not specified, defaults to bma index tenor if reference index is overnight, otherwise to reference index tenor.
- OvernightLockoutDays [Optional]: Lockout days (rate cutoff) for the reference leg, if index is overnight.

5.11.11 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 177.

Listing 177: FX conventions

```
<FX>
  <Id> </Id>
  <SpotDays> </SpotDays>
  <SourceCurrency> </SourceCurrency>
  <TargetCurrency> </TargetCurrency>
  <PointsFactor> </PointsFactor>
  <AdvanceCalendar> </AdvanceCalendar>
  <SpotRelative> </SpotRelative>
  <EOM> </EOM>
  <Convention> </Convention>
</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*.
- EOM [Optional]: A flag indicating whether the end of month roll convention is to be used for FX forward quotes. If not provided, it defaults to *False*.
- Convention [Optional]: The business day convention used when advancing dates. If not provided, it defaults to *Following*.

5.11.12 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 178.

Listing 178: Cross currency basis swap conventions

```
<CrossCurrencyBasis>
  <Id> </Id>
  <SettlementDays> </SettlementDays>
  <SettlementCalendar> </SettlementCalendar>
  <RollConvention> </RollConvention>
  <FlatIndex> </FlatIndex>
  <SpreadIndex> </SpreadIndex>
  <EOM> </EOM>
  <IsResettable> </IsResettable>
  <FlatIndexIsResettable> </FlatIndexIsResettable>>
  <PaymentLag> </PaymentLag>
  <FlatPaymentLag> </FlatPaymentLag>
  <!-- for OIS only -->
  <IncludeSpread> </IncludeSpread>
  <Lookback> </Lookback>
  <FixingDays> </FixingDays>
  <RateCutoff> </RateCutoff>
  <IsAveraged> </IsAveraged>
  <FlatIncludeSpread> </FlatIncludeSpread>
  <FlatLookback> </FlatLookback>
  <FlatFixingDays> </FlatFixingDays>
  <FlatRateCutoff> </FlatRateCutoff>
  <FlatIsAveraged> </FlatIsAveraged>
</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*.
- FlatTenor [Optional]: the flat leg period length (typical value is 3M), defaults to index tenor except for ON indices for which it defaults to 3M

- SpreadTenor [Optional]: the spread leg period length (typical value is 3M), defaults to index tenor except for ON indices for which it defaults to 3M
- SpreadPaymentLag [Optional]: the payment lag for the spread leg, allowable values are 0, 1, 2, ..., defaults to 0 if not given
- FlatPaymentLag [Optional]: the payment lag for the flat leg, allowable values are 0, 1, 2, ..., defaults to 0 if nove given
- SpreadIncludeSpread [Optional]: Only relevant if spread leg is OIS, allowable values are true, false, defaults to false if not given
- SpreadLookback [Optional]: Only relevant if spread leg is OIS, allowable values are 0D, 1D, ..., defaults to 0D if not given
- SpreadFixingDays [Optional]: Only relevant if spread leg is OIS, allowable values are 0, 1, 2, ..., defaults to 0 if not given
- SpreadRateCutoff [Optional]: Only relevant if spread leg is OIS, allowable values are 0, 1, 2, ..., defaults to 0 if not given
- SpreadIsAveraged [Optional]: Only relevant if spread leg is OIS, allowable values are true, false, defaults to false if not given
- FlatIncludeSpread [Optional]: Only relevant if spread leg is OIS, allowable values are true, false, defaults to false if not given
- FlatLookback [Optional]: Only relevant if spread leg is OIS, allowable values are 0D, 1D, ..., defaults to 0D if not given
- FlatFixingDays [Optional]: Only relevant if spread leg is OIS, allowable values are 0, 1, 2, ..., defaults to 0 if not given
- FlatRateCutoff [Optional]: Only relevant if spread leg is OIS, allowable values are 0, 1, 2, ..., defaults to 0 if not given
- FlatIsAveraged [Optional]: Only relevant if spread leg is OIS, allowable values are true, false, defaults to false if not given

5.11.13 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 179

Listing 179: 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>
  <StartDelay>2</StartDelay>
  <StartDelayConvention>Following</StartDelayConvention>
</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.
- `StartDelay`: [Optional] The inflation swap starts n business days after today, defaults to zero if omitted.
- `StartDelayConvention`: [Optional] BusinessDayConvention to adjust the start day, defaults to Following if omitted.
- `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 180.

- `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 5.11.25, i.e. they are `ScheduleData` elements. An example of the AU CPI conventions set up is given in Listing 180. 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 180: AU CPI inflation swap conventions

```
<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>
```

5.11.14 CMS Spread Option Conventions

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

Listing 181: 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.

5.11.15 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 182: 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.

5.11.16 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 183: Overnight index convention

```xml
<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.

5.11.17 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 definition will be used in the code for the given `Id`.

Listing 184: 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.

5.11.18 Swap Index Conventions

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

Listing 185: 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.

5.11.19 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 186.

Listing 186: 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 5.11.11). 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.

5.11.20 FX Option Time Weighting Scheme

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 186.

Listing 187: FX option time weighting scheme

```xml
<FxOptionTimeWeighting>
  <Id>USD-JPY-FXOPTION-TIMEWEIGHTING</Id>
  <!-- Priorities:
             1 Weekend (Saturday, Sunday)
             2 Event
             3 Trading Center (product of weights if several apply)
             4 Weekday (Monday to Friday) -->
  <WeekdayWeights>
    <Monday>1.0</Monday>
    <Tuesday>1.0</Tuesday>
    <Wednesday>1.0</Wednesday>
    <Thursday>1.0</Thursday>
    <Friday>1.0</Friday>
    <Saturday>0.3</Saturday>
    <Sunday>0.3</Sunday>
  </WeekdayWeights>
  <TradingCenters>
    <TradingCenter>
      <Name>JP,US</Name>
      <Calendar>NY,JPY</Calendar>
      <Weight>0.5</Weight>
    </TradingCenter>
  </TradingCenters>
  <Events>
    <Event>
      <Description>US Election</Description>
      <Date>2024-11-06</Date>
      <Weight>8.0</Weight>
    </Event>
  </Events>
</FxOptionTimeWeighting>
```

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

- WeekdayWeights: Specifies weights to be applied to weekdays
- TradingCenters: Zero, one or more trading centers specified by a calendar and a weight per trading center
- Events: Zero, one or more dates, on which special weights apply

To determine a weight for a given day exactly one of the following rules is applied, with priority from top to bottom:

- If a day falls on a weekend, the weight for Saturday, Sunday from WeekdayWeights is applied to the day.
- If a day is in the event list, the weight for that date is applied.
- If a day is a holiday in one or more trading centers, the product of the weights of these trading centers is applied.
- Otherwise, the weight for the weekday (Monday to Friday) is applied.

5.11.21 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 188.

Listing 188: 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 31.
- `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 23. 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 19. 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`.

5.11.22 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 189.

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 31.
- `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 or if the option contracts cycle is monthly 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 190. 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.

  - The `LastWeekday` [Optional]: This node is used to indicate a date in a given month in the form of the last named weekday of that month e.g. last Wednesday. The node takes a weekday in the form of the first three characters of the weekday with the first character capitalised.
  - The `BusinessDaysAfter` This node can contain any integer. If the number is positive the contract expiry is the n-th business day of the contract month. If the number is negative the contract expiry date is this number of business days before the first calendar day of the contract month.
  - The `WeeklyDayOfTheWeek` [Optional]: This node is used to indicate a date in a given week in the form of the named weekday, e.g. Wednesday. This node is mandatory for weekly contract frequencies and is not allowed with any other frequency. The 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`, but allowed values are `Daily`, `Weekly`, `Monthly`, `Quaterly` and `Annual`.
- `Calendar`: The business day trading calendar(s) applicable for the commodity future contract.
- `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 19. 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 189. 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 omitted, 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 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.
- `OptionContractFrequency` [Optional]: This node indicates the frequency of the commodity future options if it differs from the frequency of the underlying future contract. The value here is usually `Monthly`
- `OptionNthWeekday` [Optional]: This node has the elements shown in Listing 190. 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.
- `OptionBusinessDayConvention` [Optional]: The business day convention used to adjust the option expiry date to a good business day if `OptionExpiryDay` is used.
- `OptionExpiryLastWeekdayOfMonth` [Optional]: This node is used to indicate a date in a given month in the form of the last named weekday of that month e.g. last Wednesday. The node takes a weekday in the form of the first three characters of the weekday with the first character capitalised.
- `OptionExpiryWeeklyDayOfTheWeek` [Optional]: This node is used to indicate a date in a given week in the form of the named weekday, e.g. Wednesday. The node takes a weekday in the form of the first three characters of the weekday with the first character capitalised. This node is mandatory for weekly expiring options. The node is not allowed to use with any other option contract frequency.
- `OptionCalendarDaysBefore` [Optional]: This node can contain any non-negative integer. The option expiry date is this number of calendar days before the first calendar day of the contract month.
- `OptionMinBusinessDaysBefore` [Optional]: This node can contain any non-negative integer. The option expiry date is this at least this number of business days before the contract expiry. The option expiry date is calculated using the regular rule and if the option expiry date falls on a date before this minimum date, the minimum date will be taken instead.
- `OptionUnderlyingFutureConvention` [Optional]: Sometimes the next contract expiry, as specified in the convention, is not the correct option underlying. For example the base metals options expiries on the 1st Wednesday of the contract month, and during the first 3 months there are daily future contracts available. The option underlying is not the future contract which matures on the option expiry but the one which matures on the 3rd Wednesday of the month. This field is referencing to an commodity future convention which specifies the correct expiry date for the underlying contract.

- `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.

- `SavingsTime` [Optional]: For some derivatives, quantities are given as quantity per calendar day and hour. The monthly quantity is then scaled by the number of calendar days times hours per day (see above) plus or minus a daylight savings correction. To compute the daylight savings correction a convention is needed that describes the dates on which dates one hour is gained resp. lost. Currently supported conventions are US, Null. Default is US if no convention is given.

- `ValidContractMonths` [Optional]: For some commodities the contract frequency is almost monthly but for some calendar months there are no contracts listed. For example Corn Futures are only listed for the expiry months March, May, July, September and December. For those contracts the *ContractFrequency* need to be set to *Monthly* and the valid months have to be added to this node. This node is ignored for all other frequencies and if its omitted all calendar months are valid.

An example `CommodityFuture` node for the NYMEX WTI future contract, specified here, is provided in Listing 191.

The `AveragingData` node referenced above has the structure shown in Listing 192. 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*.

5.11.23 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 193.

Listing 193: CDS conventions

```
<CDS>
  <Id>...</Id>
  <SettlementDays>...</SettlementDays>
  <Calendar>...</Calendar>
  <Frequency>...</Frequency>
  <PaymentConvention>...</PaymentConvention>
  <Rule>...</Rule>
  <DayCounter>...</DayCounter>
  <LastPeriodDayCounter>...</LastPeriodDayCounter>
  <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 23.
- 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 19 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 24 is allowed.
- `LastPeriodDayCounter` [optional]: The day counter for the last fee leg payment on the CDS. The ISDA standard is `A360 (Incl Last)` but any valid day counter from Table 24 is allowed. If not given, the following fallback rule applies: If DayCounter is `A360`, LastPeriodDayCounter is set to `A360 (Incl Last)`, otherwise LastPeriodDayCounter is set to the same value as DayCounter.
- `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 35. 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`.

5.11.24 Bond Yield Conventions

A node with name `BondYield` is used to store conventions for the conversion of bond prices into bond yields. The structure of this node is shown in Listing 194.

Listing 194: Bond yield conventions

```
<BondYield>
  <Id>CMB-DE-BUND-10Y</Id>
  <Compounding>Compounded</Compounding>
  <Frequency>Annual</Frequency>
  <PriceType>Clean</PriceType>
  <Accuracy>1.0e-8</Accuracy>
  <MaxEvaluations>100</MaxEvaluations>
  <Guess>0.05</Guess>
</BondYield>
```

The meaning of the elements is as follows:

- Id: The constant maturity index index name. This must be of the form "CMB-FAMILY-TENOR" where FAMILY can consist of any number of tags separated by "-"
- Compounding: Compounding of the yield - Simple, Compounded, Continuous, SimpleThenCompounded
- Frequency: Frequency of the cash flows - Annual, Semiannual, Quarterly, Monthly etc.
- PriceType: Dirty or Clean
- Accuracy/MaxEvaluations/Guess: QuantLib parameters that control the convergence of the numerical price to yield conversion.

Listing 189: 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>
  <OptionContractFrequency>...</OptionContractFrequency>
  <OptionNthWeekday>
    <Nth>...</Nth>
    <Weekday>...</Weekday>
  </OptionNthWeekday>
  <OptionExpiryLastWeekdayOfMonth>...</OptionExpiryLastWeekdayOfMonth>
  <OptionExpiryWeeklyDayOfTheWeek>...</OptionExpiryWeeklyDayOfTheWeek>
  <OptionBusinessDayConvention>...</OptionBusinessDayConvention>
  <FutureContinuationMappings>
    <ContinuationMapping>
      <From>...</From>
      <To>...</To>
    </ContinuationMapping>
    ...
  </FutureContinuationMappings>
  <OptionContinuationMappings>
    <ContinuationMapping>
      <From>...</From>
      <To>...</To>
    </ContinuationMapping>
    ...
  </OptionContinuationMappings>
  <AveragingData>
    ...
  </AveragingData>
  <HoursPerDay>...</HoursPerDay>
  <SavingsTime>...<SavingsTime>
  <ValidContractMonths>
          <Month>...</Month>
  </ValidContractMonths>
  <OptionUnderlyingFutureConvention>...</OptionUnderlyingFutureConvention>
</CommodityFuture>
```

Listing 190: `NthWeekday` *node outline*

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

Listing 191: 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>
```

Listing 192: ***`AveragingData`*** *node structure*

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

5.11.25 Schedule Data (Rules, Dates and Derived)

The ScheduleData trade component node is used within the LegData trade component. The Schedule can be rules based (at least one Rules sub-node exists), dates based (at least one Dates sub-node exists, where the schedule is determined directly by Date child elements), or derived from another schedule in the same leg (at least one Derived sub-node exists). 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, dates and derived from a base schedule are shown in Listing 195, Listing 196, and Listing 197 respectively.

Listing 195: Schedule data, rules based

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

Listing 196: Schedule data, date based

```xml
<ScheduleData>
  <Dates>
    <Calendar>NYB</Calendar>
    <Convention>Following</Convention>
    <Tenor>3M</Tenor>
    <EndOfMonth>false</EndOfMonth>
    <EndOfMOnthConvention>Following</EndOfMOnthConvention>
    <IncludeDuplicateDates>false</IncludeDuplicateDates>
    <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>
```

Listing 197: Schedule data, derived

```
              <ScheduleData>
                <Derived>
                  <BaseSchedule>ScheduleData</BaseSchedule>
                  <Shift>3M</Shift>
                  <Calendar>GBP</Calendar>
                  <Convention>Following</Convention>
                </Derived>
              </ScheduleData>
```

The ScheduleData section can contain any number and combination of `<Dates>`, `<Rules>` and `<Derived>` 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`, and `Rule`. `EndDateConvention` is optional. If not existing, a `Dates` or `Derived` sub-node is required.
- `StartDate`: The schedule start date.

  Allowable values: See `Date` in Table 19.
- `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 19.
- `AdjustEndDateToPreviousMonthEnd` [Optional]: Only relevant for commodity legs. Allows for the `EndDate` to be on a date other than the end of the month. If set to *true* the given `EndDate` is restated to the end date of the previous month.

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

  Note that *0D* and *1T* are equivalent valid values, and both cause there to be no intermediate dates between `StartDate` and `EndDate`.
- `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 23 Calendar.

- `Convention`: Determines the adjustment of the schedule dates with regards to the selected calendar, i.e. the roll convention.

  Allowable values: See Table 20 Roll Convention.

- `TermConvention` [Optional]: Determines the adjustment of the final schedule date with regards to the selected calendar. If left blank or omitted, defaults to the value of `Convention`.

  Allowable values: See Table 20 Roll Convention.

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

  Allowable values and descriptions: See Table 22 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.

  If `EndOfMonth` is *true*, and `EndOfMonthConvention` is omitted:
  - the date is set to the last calendar day in a month if the roll convention is *Unadjusted*, and
  - the date is set to the last business day in a month if the roll convention is anything other than *Unadjusted*

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 35. 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*.

- `EndOfMonthConvention` [Optional]: Determines the adjustment of the end-of-month schedule dates with regards to the selected calendar. This field is only used when `EndOfMonth` is *true*. If left blank or omitted, then the default *Preceding* / *MF* convention is applied (i.e. end-of-month dates will never be adjusted over to the beginning of the next month)

  Allowable values: See Table 20 Roll Convention.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 35. 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, determining the end date of the first period. If omitted the first period will follow the date generation rule. Note that for date generation rules *CDS* and *CDS2015*, the FirstDate has no impact and the schedule is built from IMM dates.

  Allowable values: See `Date` in Table 19. The FirstDate cannot be before the StartDate of the Schedule, and cannot be after the EndDate of the Schedule.

- `LastDate` [Optional]: Date for final stub period, determining the start date of the last period. If omitted the last period will follow the date generation rule.

For date generation rules *CDS* and *CDS2015*, the LastDate has no impact and the schedule is built from IMM dates.

Allowable values: See `Date` in Table 19. The LastDate cannot be after the EndDate of the Schedule, and cannot be before the StartDate of the Schedule.

- `RemoveFirstDate` [Optional]: If true the first date will be removed from the schedule. Useful to define a payment schedule using the rules for a calculation schedule.

  Allowable values: true, false

- `RemoveLastDate` [Optional]: If true the last date will be removed from the schedule. Useful to define a fixing or reset schedule using the rules for a calculation schedule.

  Allowable values: true, false

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 23 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 Table 20 Roll Convention. 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 35. Defaults to *false* if left blank or omitted.

- `EndOfMonthConvention` [Optional]: Whenever the `EndOfMonth` logic is applied, this is used as the roll convention along with the `Calendar`for any date adjustments.

  Allowable values: See Table 20 Roll Convention. Defaults to *Preceding* if omitted.

- `IncludeDuplicateDates` [Optional]: If set to *false* the resulting schedule will have unique set of dates and all duplicates will be removed. Default to *false*.

- `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 19.

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

- `BaseSchedule`: The schedule from which the derived schedule will be deduced.

  Allowable values: Must be the node name of another schedule in a given leg data node.

- `Shift` [Optional]: The tenor/period offset to be applied to each date in the base schedule in order to obtain the derived schedule.

  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. If left blank or omitted, defaults to *0D*.

- `Calendar` [Optional]: The calendar adjustment to be applied to each date in the base schedule in order to obtain the derived schedule.

  Allowable values: See Table 23 Calendar. Defaults to *NullCalendar* if left blank or omitted, i.e. no holidays at all, not even on weekends.

- `Convention` [Optional]: The roll convention to be applied to each date in the base schedule in order to obtain the derived schedule.

  Allowable values: See Table 20 Roll Convention. Defaults to *Unadjusted* if left blank or omitted.

- `RemoveFirstDate` [Optional]: If true the first date will be removed from the schedule. Useful to define a payment schedule based on a calculation schedule.

  Allowable values: true, false

- `RemoveLastDate` [Optional]: If true the last date will be removed from the schedule. Useful to define a fixing or reset schedule based on a calculation schedule.

  Allowable values: true, false

Listing 198: Historical return configuration

```xml
<ReturnConfiguration>
    <Return key="DiscountCurve">
        <Type>Log</Type>
        <Displacement>0.0</Displacement>
    </Return>
    <Return key="EquitySpot">
        <Type>Relative</Type>
        <Displacement>0.0</Displacement>
    </Return>
    <Return key="EquitySpot/EQUITY_1">
        <Type>Absolute</Type>
        <Displacement>0.0</Displacement>
    </Return>
    <!-- ... more configurations ... -->
</ReturnConfiguration>
```

5.12 Historical Return Configuration

The `ReturnConfiguration` allows the user to specify, for each risk factor type, how historical returns are computed in historical scenario generation and backtesting. This configuration controls the return type (e.g., log, absolute, relative) and an optional displacement for each risk factor. Additionally, it is possible to override the default return configuration for specific names (e.g., for a particular equity or commodity).

The root element is `ReturnConfiguration`, which contains one or more `<Return>` blocks. Each `Return` must have a `key` attribute, either the risk factor key type for the default configuration (e.g., `DiscountCurve`, `FXSpot`, `EquitySpot`, etc.). or the risk factor key type plus the underlying name for a specific override (e.g `EquitySpot/EQUITY1`.

Elements and Attributes

- `Return key="..."`: Specifies the configuration for a risk factor type. The `key` attribute must match a valid risk factor key type or the specific key type and name.
    - `Type`: The return type. Allowed values are `Log`, `Absolute`, and `Relative`.
    - `Displacement`: The displacement value (floating point). Used to avoid division by zero or negative values in relative/log returns.

Notes

- If no override is specified for a particular name, the default return configuration for the risk factor type is used.
- The displacement should be set to a small positive value if the risk factor can approach zero, to avoid numerical issues.

6 Collateral Balances

The collateral balances file - `collateralbalances.xml` - contains the list of collateral balances (i.e. margin amounts and independent amount) under a Credit Support Annex.

The balances of each netting set are defined within their own `CollateralBalance` node. All of these `CollateralBalance` nodes are contained as children of a `CollateralBalances` node.

The collateral balances are given in the following XML template:

Listing 199: Collateral balance definition

```xml
<CollateralBalances>
    <CollateralBalance>
        <NettingSetId> </NettingSetId>
        <Currency>USD</Currency>
        <IndependentAmountHeld/>
        <InitialMargin> </InitialMargin>
        <VariationMargin> </VariationMargin>
    </CollateralBalance>
    <CollateralBalance>
        .......
    </CollateralBalance>
</CollateralBalances>
```

The meanings of the various elements of the `CollateralBalance` node are as follows (default input values for certain analytics are specified in their own respective sections, otherwise the defaults given below, if any, are applicable):

- `NettingSetId`: The unique identifier for the (collateralised) ISDA netting set.
  Allowable values: Any string.
- `Currency`: The currency that the collateral balance amounts are assumed to be denominated in.
  Allowable values: See Table 21.
- `IndependentAmountHeld` [Optional]: The netted sum of all independent amounts covered by the CSA.
  Allowable values: Any number. A negative number implies that the counterparty holds the independent amount. If provided, overrides the specified independent amount held (if any) in the corresponding netting set definitions file. Otherwise (if left blank or omitted), the independent amount in the netting set definitions file is used.
- `InitialMargin` [Optional]: The initial margin amount received.
  Allowable values: Any number. A negative number implies that the counterparty holds the initial margin.
- `VariationMargin` [Optional]: The variation margin amount received.

Allowable values: Any number. A negative number implies that the counterparty holds the variation margin.

7 Counterparty Information

The counterparty information file - `counterparty.xml` - contains a list of counterparty-level details. The file is written in XML format, with a top-level `CounterpartyInformation` node consisting of two children nodes: `Counterparties` and `Correlations`.

7.1 Counterparties

The `Counterparties` node is used to define inputs for each counterparty in the calculations. Each counterparty is then defined with its own `Counterparty` node, with the following XML template:

Listing 200: Counterparty definition

```
<Counterparties>
    <Counterparty>
        <CounterpartyId> </CounterpartyId>
        <CreditQuality>IG</CreditQuality>
        <BaCvaRiskWeight> </BaCvaRiskWeight>
        <SaCcrRiskWeight> </SaCcrRiskWeight>
        <SaCvaRiskBucket> </SaCvaRiskBucket>
    </Counterparty>
    <Counterparty>
        .......
    </Counterparty>
</Counterparties>
```

The meanings of the various elements of the `Counterparty` node are as follows (default input values for certain analytics are specified in their own respective sections, otherwise the defaults below are applicable):

- `CounterpartyId`: The unique identifier for the counterparty.
  Allowable values: Any string.
- `ClearingCounterparty` [Optional]: Whether the counterparty is a clearing counterparty.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 35. If left blank or omitted, defaults to *False*.
- `CreditQuality` [Optional]: Credit quality/rating.
  Allowable values: *HY* (high yield), *IG* (investment grade), *NR* (not rated). Defaults to *NR* if left blank or omitted.
- `BaCvaRiskWeight` [Optional]: BA-CVA supervisory risk weight based on sector and credit quality. This field is only used when calculating BA-CVA or SA-CVA.
  Allowable values: Any number. If left blank or omitted, defaults to zero.
- `SaCcrRiskWeight` [Optional]: SA-CCR supervisory risk weight based on sector and credit quality. This field is only used when calculating SA-CCR or BA-CVA (which itself calculates SA-CCR).
  Allowable values: Any number. If left blank or omitted, defaults to *1.0*.

- `SaCvaRiskBucket` [Optional]: SA-CVA delta risk buckets for counterparty credit spread.
  Allowable values: Any integer from *1* to *8* (inclusive).

7.2 Correlations

The `Correlations` node is used to define counterparty correlations which are used for calculating correlations between counterparty credit risk factor, with the following XML template:

Listing 201: Counterparty correlations definition

```
<Correlations>
    <Correlation cpty1="CPTY_A" cpty2="CPTY_B">0.5</Correlation>
    <Correlation cpty1="CPTY_B" cpty2="CPTY_C">0.5</Correlation>
    ....
</Correlations>
```

7.3 Allowable Values

`Date`	
Date Fields	**Allowable Values**

All Date fields:
`StartDate`
`EndDate`
`Date`
`ExerciseDate`
`PayDate`
`ValueDate`
`NearDate`
`FarDate`
etc
	Any of 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.

Table 19: Allowable Values for Date

`Convention`	
Roll Convention Fields	**Allowable Values**

All Convention fields:
`Convention`
`TermConvention`
`PaymentConvention`
etc
	F, Following, FOLLOWING
	MF, ModifiedFollowing, Modified Following, MODIFIEDF
	P, Preceding, PRECEDING
	MP, ModifiedPreceding, Modified Preceding, MODIFIEDP
	U, Unadjusted, INDIFF
	HMMF, HalfMonthModifiedFollowing, HalfMonthMF, Half Month Modij
	NEAREST (takes future date in case of equal distance)

Table 20: Allowable Values for Roll Conventions

`Currency`	
Category	**Allowable Values**
Fiat Currencies	*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,XAU,XCD,XOF, XPF,XSU,XUA,YER,ZAR,ZMW,ZWL*
Minor Currencies	*GBp, GBX* (for pennies of GBP) *ILa, ILX, ILs, ILA* (for agorot of ILS) *ZAc, ZAC, ZAX* (for cents of ZAR) Note: Minor Currency codes are only supported for equity products.
Precious Metals treated as Currencies	*XAG, XAU, XPD, XPT*
Cryptocurrencies	*BTC, XBT, ETH, ETC, BCH, XRP, LTC*
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.	

Table 21: Allowable Values for Currency

`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).

EveryThursday	If FirstDate is not given, all thursdays between start and end date. If FirstDate is given, FirstDate plus all thursdays between FirstDate and end date.

Table 22: 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
US-SOFR	US SOFR fixing 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
CY, CYP	Cyprus 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
GR, GRC	Greek 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
IE, IRL	Ireland 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
AE, ARE, AED	United Arab Emirates
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
XASX	Australian Securities Exchange 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
XDUB	Ireland 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
XPAR	Paris stock exchange
WMR	Thomson Reuters WM/Reuters Spot
IslamicWeekendsOnly	Islamic Weekends Only Calendar
WeekendsOnly	Weekends Only Calendar
IslamicWeekendsOnly	Islamic 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
NullCalendar, Null	Null Calendar, i.e. all days are business days

Table 23: 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)
Act/365 (Canadian Bond)	Actual 365 Fixed (Canadian Bond)
T360, 30/360, ACT/nACT, 30/360 US, 30/360 (US), 30U/360, 30US/360	Thirty 360 (US)
30/360 NASD	Thirty 360 (NASD)
30/360 (Bond Basis)	Thirty 360 (Bond Basis)
30E/360 (Eurobond Basis), 30E/360, 30/360 AIBD (Euro), 30E/360.ICMA, 30E/360 ICMA	Thirty 360 (European)
30E/360E, 30E/360 ISDA, 30E/360.ISDA, 30/360 German, 30/360 (German)	Thirty 360 (German)
30/360 Italian, 30/360 (Italian)	Thirty 360 (Italian)
ActActISDA, ACT/ACT.ISDA, Actual/Actual (ISDA), ActualActual (ISDA), ACT/ACT, 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), ACT29	Actual Actual (AFB)
BUS/252, Business/252	Brazilian Bus/252
1/1	1/1
Simple	Simple Day Counter
Year	Year Counter

Table 24: 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-EURIBOR365* *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* *RUB-MOSPRIME* *SKK-BRIBOR* *THB-THBFIX* *THB-THOR* *THB-BIBOR* *NZD-BKBM*
TENOR	An integer followed by *D, W, M or Y*

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 26: 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 27: 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 28: 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 configuration). 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 29: 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 configuration). Typically a RIC-code with the *RIC:* prefix Examples: *RIC:.SPX* (S&P 500 Index) *RIC:EEM.N* (iShares MSCI Emerging Markets ETF)

Table 30: 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 31: 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
`LIEN1`	First lien
`LIEN2`	Second lien
`LIEN3`	Third lien

Table 32: 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 33: 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 34: Allowable Values for Exchange

Boolean nodes	
Node Value	**Evaluates To**
`Y`, `YES`, `TRUE`, `true`, `1`	`true`
`N`, `NO`, `FALSE`, `false`, `0`	`false`

Table 35: Allowable values for boolean node

8 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.

8.1 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 202: Uncollateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <ActiveCSAFlag> </ActiveCSAFlag>
    <CSADetails></CSADetails>
</NettingSet>
```

The meanings of the various elements are as follows:

- `NettingSetId`: The unique identifier for the ISDA netting set.
  Allowable values: Any string
- `ActiveCSAFlag` [Optional]: Boolean indicating whether the netting set is covered by a Credit Support Annex. Allowable values: For uncollateralised netting sets this flag should be *False*. If left blank or omitted, defaults to *True*.
- `CSADetails` [Optional]: Node containing as children details of the governing Credit Support Annex. For uncollateralised netting sets, this node is not needed.

8.2 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 203: Collateralised netting set definition

```
<NettingSet>
    <NettingSetId> </NettingSetId>
    <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>
        <CalculateIMAmount>true</CalculateIMAmount>
        <CalculateVMAmount>true</CalculateVMAmount>
    </CSADetails>
</NettingSet>
```

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` [Optional]: 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.

Defaults to *Bilateral* if left blank or omitted.

- `CSACurrency` [Optional]: 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.
  Allowable values: Any currency. See Table 21.
- `Index` [Optional]: The index is used to derive the fixing which is used for compounding cash collateral in the master currency of the CSA.
  Allowable values: An alphanumeric string of the form CCY-INDEX-TENOR. CCY, INDEX and TENOR must be separated by dashes (-). CCY and INDEX must be among the supported currency and index combinations. TENOR must be an integer followed by *D*, *W*, *M* or *Y*, except for Overnight indices which do not require a TENOR. See Table 25.
- `ThresholdPay` [Optional]: A threshold amount above which the counterparty is entitled to request collateral to cover excess exposure.
  Allowable values: Any number.
- `ThresholdReceive` [Optional]: A threshold amount above which we are entitled to request collateral from the counterparty to cover excess exposure.
  Allowable values: Any number.
- `MinimumTransferAmountPay` [Optional]: 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.
  Allowable values: Any number.
- `MinimumTransferAmountReceive` [Optional]: 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.
  Allowable values: Any number.
- `IndependentAmount` [Optional]: This element contains two child nodes:
  - `IndependentAmountHeld`: The netted sum of all independent amounts covered by this ISDA agreement/CSA.
    Allowable values: Any number. 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.
    Allowable values: The only supported value here is *FIXED*.
- `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*).
    Allowable values:
  - `PostFrequency`: The frequency with which the counterparty is entitled to request additional margin from us.

Allowable values: Any period definition (e.g. *2D*, *1W*, *1M*, *1Y*).

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).

- `MarginPeriodOfRisk`: The length of time assumed necessary for closing out the portfolio position after a default event.
  Allowable values: Any period definition (e.g. *2D*, *1W*, *1M*, *1Y*).
- `CollateralCompoundingSpreadReceive`: The spread over the O/N interest accrual rate taken by the clearing house, when holding collateral.
  Allowable values: Any number.
- `CollateralCompoundingSpreadPay`: The spread over the O/N interest accrual rate taken by the clearing house, when collateral is held by the counterparty.
  Allowable values: Any number.
- `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.
  Allowable values: Any currency. See Table 21.
- `ApplyInitialMargin`: Apply (dynamic) initial Margin in addition to variation margin
  Allowable values: Boolean node, the set of allowable values is given in Table 35.
- `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.
- `CalculateIMAmount`: Boolean indicating whether to calculate initial margin from SIMM. For uncollateralised netting sets this flag will be ignored. This only applies to the SA-CCR calculations.
  Allowable values: Boolean node, the set of allowable values is given in Table 35.
- `CalculateVMAmount`: Boolean indicating whether to calculate variation margin from the netting set NPV. For uncollateralised netting sets this flag will be ignored. This only applies to the SA-CCR calculations.
  Allowable values: Boolean node, the set of allowable values is given in Table 35.

9 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 19.

- **Quote**: A generic description that contains Instrument Type and Quote Type, followed by instrument specific descriptions (see 9.1 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 36.

- **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, ASSUMED_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, CONV_CREDIT_SPREAD, YIELD_SPREAD, HAZARD_RATE, RATE, RATIO, PRICE, RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, BASE_CORRELATION, SHIFT*

Table 36: Allowable values for Instrument and Quote type market data.

An excerpt from a typical `market.txt` file is shown in Listing 204.

Listing 204: 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
```

9.1 Zero Rate

The instrument specific information to be captured for quotes representing Zero Rates is shown in Table 37.

Property	**Allowable values**	**Description**
Instrument Type	*ZERO*	
Quote Type	*RATE, YIELD_SPREAD*	
Currency	See `Currency` in Table 19	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 24	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 19	Either a Tenor for tenor based zero quotes, or an explicit maturity date (ZeroDate)

Table 37: Zero Rate

Examples with a Tenor and with a ZeroDate:

- ZERO/RATE/USD/USD6M/A365F/6M

- ZERO/RATE/USD/USD6M/A365F/12-05-2018

9.2 Discount Factor

The instrument specific information to be captured for quotes representing Discount Factors is shown in Table 38.

Property	**Allowable values**	**Description**
Instrument Type	*DISCOUNT*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 19	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 19	Either a Term is used to determine the maturity date, or an explicit maturity date (Discount Date) is given.

Table 38: Discount Rate

If a Term is given in the last element of the quote, it is converted to a maturity date using the calendar, specified in the conventions. Bear in mind, only zero conventions (see Listing 162) can be used for the discount factor instruments.

Examples with a Term and with a DiscountDate:

- DISCOUNT/RATE/EUR/EUR3M/3Y
- DISCOUNT/RATE/EUR/EUR3M/12-05-2018

9.3 FX Spot Rate

Property	**Allowable values**	**Description**
Instrument Type	*FX*	
Quote Type	*RATE*	
Unit currency	See `Currency` in Table 19	Unit/Source currency
Target currency	See `Currency` in Table 19	Target currency

Table 39: FX Spot Rate

Example:

- FX/RATE/EUR/USD

9.4 FX Forward Rate

An FX Forward quote is expected in either a “forward points” quotation, or an “outright” quotation.

The forward points convention is given by:

$$\text{Forward Points} = \frac{\text{FX Forward} - \text{FX Spot}}{\text{Conversion Factor}}$$

with conversion factor set to 1.

Property	**Allowable values**	**Description**
Instrument Type	*FX_FWD*	
Quote Type	*RATE*	
Unit currency	See `Currency` in Table 19	Unit/Source currency
Target currency	See `Currency` in Table 19	Target currency
Term	An integer followed by D, W, M or Y.	Period from today to maturity. Alternatively, the maturity date.

Table 40: FX Forward Rate

The forward outright is given by:

$$\text{Forward Outright} = \text{FX Spot} + \text{Forward Points} \times \text{Conversion Factor}$$

with conversion factor set to 1.

Property	**Allowable values**	**Description**
Instrument Type	*FX_FWD*	
Quote Type	*PRICE*	
Unit currency	See `Currency` in Table 19	Unit/Source currency
Target currency	See `Currency` in Table 19	Target currency
Term	An integer followed by D, W, M or Y.	Period from today to maturity. Alternatively, the maturity date.

Table 41: FX Forward Rate

Example:

- FXFWD/RATE/EUR/USD/1M
- FXFWD/PRICE/EUR/USD/3M
- FXFWD/PRICE/EUR/USD/2026-03-02

9.5 Deposit Rate

Property	**Allowable values**	**Description**
Instrument Type	*MM*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 19	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 42: 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 5.11, and is between 0D and 2D, i.e. "spot days" are between 0 and 2.

Example:

- MM/RATE/EUR/2D/3M

9.6 FRA Rate

Property	**Allowable values**	**Description**
Instrument Type	*FRA*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 19	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 43: 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 19	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 44: IMM FRA Rate

Example:

- FRA/RATE/EUR/9M/3M
- IMM_FRA/RATE/EUR/2/3

9.7 Money Market Futures Price

Property	**Allowable values**	**Description**
Instrument Type	*MM_FUTURE*	
Quote Type	*PRICE*	
Currency	See `Currency` in Table 19	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 45: 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 5.11.3.

Example:

- MM_FUTURE/PRICE/EUR/2018-06/LIF3ME/3M

9.8 Overnight Index Futures Price

Property	**Allowable values**	**Description**
Instrument Type	*OI_FUTURE*	
Quote Type	*PRICE*	
Currency	See `Currency` in Table 19	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 46: 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 5.11.3.

Example: Three Months SOFR Futures (DEC 2019):

- OI_FUTURE/PRICE/USD/2019-12/CME:SR3Z2019/3M

9.9 Swap Rate

Property	**Allowable values**	**Description**
Instrument Type	*IR_SWAP*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 19	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 47: Swap Rate

Property	**Allowable values**	**Description**
Instrument Type	*IR_SWAP*	
Quote Type	*RATE*	
Currency	See `Currency` in Table 19	Currency of the Swap rate
IndexName	Optional, any string	Generally used to differentiate swaps referencing different interest rate indices with the same tenor
Start Date	A valid date	
Tenor	An integer followed by D, W, M or Y	Underlying index period
End Date	A valid date	

Table 48: Swap Rate with Start and End Date

Forward start for the non-dated variant is usually not quoted, but needs to be derived from conventions.

Example:

- IR_SWAP/RATE/EUR/2D/6M/10Y
- IR_SWAP/RATE/GBP/20230921/1D/20231102

9.10 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 19	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 49: 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

9.11 Cross Currency Basis Swap Spread

Property	Allowable values	Description
Instrument Type	*CC_BASIS_SWAP*	
Quote Type	*BASIS_SPREAD*	
Flat currency	See `Currency` in Table 19	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 19	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 50: Cross Currency Basis Swap Spread

Example:

- CC_BASIS_SWAP/BASIS_SPREAD/USD/3M/JPY/6M/10Y

9.12 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 19	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 51: CDS spread quote

Currently, only par CDS spreads (quote type `CREDIT_SPREAD`) are supported. Conventional CDS spreads (quote type `CONV_CREDIT_SPREAD`) are not supported at this time. 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
- CDS/CONV_CREDIT_SPREAD/JPM/SNRFOR/USD/5Y
- CDS/CONV_CREDIT_SPREAD/JPM/SNRFOR/USD/5Y/100
- CDS/CONV_CREDIT_SPREAD/JPM/SNRFOR/USD/XR14/5Y
- CDS/CONV_CREDIT_SPREAD/JPM/SNRFOR/USD/XR14/5Y/100
- CDS/CONV_CREDIT_SPREAD/RBS/SUBLT2/EUR/MR14/10Y
- CDS/CONV_CREDIT_SPREAD/RBS/SUBLT2/EUR/MR14/10Y/500
- CDS/CONV_CREDIT_SPREAD/RBS/SUBLT2/EUR/1Y
- CDS/CONV_CREDIT_SPREAD/RBS/SUBLT2/EUR/1Y/500

9.13 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 19	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 52: 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

9.14 CDS Recovery Rate

Property	Allowable values	Description
Instrument Type	RECOVERY_RATE or ASSUMED_RECOVERY_RATE	
Quote Type	RATE	
Entity	String	The CDS reference entity name
Tier	String	The CDS tier
Currency	See Currency in Table 19	The CDS currency
DocClause	String	Optional, the CDS doc clause

Table 53: 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

9.15 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.

9.16 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* or *ASSUMED_RECOVERY_RATE*	
Quote Type	*RATE*	
ID	String	Security ID

Table 54: Security Recovery Rate

Currently, only par (real) recovery rates (quote type `RECOVERY_RATE`) are supported.

Example:

- RECOVERY_RATE/RATE/SECURITY_1
- ASSUMED_RECOVERY_RATE/RATE/SECURITY_1

9.17 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 19	Hazard rate currency
Term	An integer followed by D, W, M or Y	Generic period from start to maturity

Table 55: Hazard Rate

Example:

- HAZARD_RATE/RATE/CPTY_A/SR/USD/30Y
- HAZARD_RATE/RATE/CPTY_C/SR/EUR/0Y

9.18 FX Option Implied Volatility

Property	**Allowable values**	**Description**
Instrument Type	*FX_OPTION*	
Quote Type	*RATE_LNVOL*	
Unit currency	See `Currency` in Table 19	Unit/Source currency
Target currency	See `Currency` in Table 19	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 56: 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

9.19 Cap Floor Implied Volatility

Property	Allowable values	Description
Instrument Type	*CAPFLOOR*	
Quote Type	*RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, SHIFT, PRICE*	Lognormal quoted volatility, normal quoted volatility, shifted lognormal quoted volatility, shift quote, premium quote.
Currency	See `Currency` in Table 19	Currency of the cap floor quote.
Index Name	A string	(optional) An interest rate index name giving the index underlying the cap floor quotes. See Table 25.
Term	A valid tenor string	Period from start to cap or floor maturity.
Index Tenor	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.
Option Style	C or F	(optional) Valid for premium quotes only, indicates whether the datum is a cap or floor quote respectively.

Table 57: Cap floor implied volatility quote

An index name should be used where a currency has more than one index of a given tenor with an options surface. It must match `IborIndex` in its corresponding `CapFloorVolatility` configuration, see section 5.7.6.

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/Index Tenor` where the meaning of `Currency` and `Index Tenor` are given in Table 57.

We have the following examples of cap floor implied volatility, shift, and premium 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.
- `CAPFLOOR/RATE_SLNVOL/EUR/EURIBOR/5Y/6M/0/0/0.03`: 5Y 3% strike cap floor implied shifted lognormal volatility quote where the underlying rate is the EUR-EURIBOR-6M.

- `CAPFLOOR/SHIFT/EUR/EURIBOR/6M`: Strike shift convention corresponding to shifted lognormal implied capfloor volatility quotes where the underlying rate is the EUR-EURIBOR-6M.
- `CAPFLOOR/PRICE/EUR/EURIBOR/5Y/6M/0/0/0.03/C`: 5Y 3% strike cap floor premium quote where the underlying rate is the EUR-EURIBOR-6M.

9.20 Swaption Implied Volatility

Property	**Allowable values**	**Description**
Instrument Type	*SWAPTION*	
Quote Type	*RATE_LNVOL, RATE_NVOL, RATE_SLNVOL, SHIFT, PRICE*	Lognormal quoted volatility, Normal quoted volatility, shifted lognormal quoted volatility, shift, premium quote.
Currency	See `Currency` in Table 19	Currency of the Swaption volatility
Quote Tag	A string	(optional) A tag to differentiate different sets of swaption data in a currency. See note below.
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	(not required for ATM), deviation from the ATM strike. Note that trailing 0s are not ignored.
Option Style	P or R	(optional) Valid for premium quotes only, indicates whether the datum represents a payer or receiver swaption premium respectively.

Table 58: Swaption Implied Volatility

A quote tag should be used where a currency has more than one index with a swaption surface. It must match `QuoteTag` in its corresponding `SwaptionVolatility` configuration, see section 5.7.5.

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 58.

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)

9.21 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 19	Currency of the equity

Table 59: Equity Spot Price

9.22 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 19	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 60: Equity Forward Price

Examples:

- EQUITY_FWD/PRICE/SP5/USD/2016-06-16
- EQUITY_FWD/PRICE/SP5/USD/2Y

9.23 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 19	Currency of the equity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 61: Equity Dividend Yield Rate

Examples:

- EQUITY_DIVIDEND/RATE/SP5/USD/2016-06-16
- EQUITY_DIVIDEND/RATE/SP5/USD/2Y

9.24 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 19	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 62: 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

9.25 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 19	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 63: 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

9.26 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 19	Currency of the commodity

Table 64: Commodity Spot Price

Examples:

- COMMODITY/PRICE/PM:XAUUSD/USD

9.27 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 19	Currency of the commodity
Maturity	Date string, or integer followed by D, W, M or Y	Maturity of the forward quote

Table 65: Commodity Forward Price

Examples:

- COMMODITY_FWD/PRICE/NYMEX:CL/USD/2030-11-20

9.28 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 19	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 66: 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 67: 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 68: 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 69: 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

9.29 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 70: 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.

9.30 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 71: 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.

9.31 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 72: 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.

9.32 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 73: 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.

9.33 Bond Yield Spreads

Property	**Allowable values**	**Description**
Instrument Type	*BOND*	
Quote Type	*YIELD_SPREAD*	
Name	String	Identifying name of the bond

Table 74: Bond Yield Spreads

This quote provides the spread for a specified bond over the benchmark rate.

Examples:

- BOND/YIELD_SPREAD/SECURITY_1

9.34 Bond Prices

Property	**Allowable values**	**Description**
Instrument Type	*BOND*	
Quote Type	*PRICE*	
Name	String	Identifying name of the bond

Table 75: Bond Prices

This quote provides the price for a specified bond. The reference data of the bond specifies the further characteristics of the price, i.e. type (clean or dirty) and quote method (percentage of par or currency per unit).

Examples:

- BOND/PRICE/SECURITY_1

9.35 Bond Future Price

Property	**Allowable values**	**Description**
Instrument Type	*BOND_FUTURE*	
Quote Type	*PRICE*	
Name	String	Identifying name of the future contract

Table 76: Bond Future Prices

This quote provides the price for a specified bond future contract.

Examples:

- BOND_FUTURE/PRICE/TYH25

9.36 Bond Future Conversion Factor

Property	**Allowable values**	**Description**
Instrument Type	*BOND_FUTURE*	
Quote Type	*CONVERSION_FACTOR*	
Name	String	Identifying name of the bond
FutureContract	String	Identifying name of the future contract

Table 77: Bond Conversion Factor

This quote provides the conversion factor for a deliverable bond of a bond future contract.

Examples:

- BOND_FUTURE/CONVERSION_FACTOR/ISIN:US91282CNF40/TYU25

9.37 Base Correlations

Property	**Allowable values**	**Description**
Instrument Type	*CDS_INDEX*	
Quote Type	*BASE_CORRELATION*	
Index	String	CDS index name
Term	Period (e.g. 5Y)	Term on the base correlation curve, the curve is flat if quotes for only one term are provided
DetachmentPoint	Real in (0...1)	Detachment point of the equity tranche this quote refers to

Table 78: Base correlation quotes

This quote provides the base correlation for a CDS index' equity tranche with the specified detachment point. Example:

- CDS_INDEX/BASE_CORRELATION/2I65BYBD6/5Y/0.03

Typically there are several base correlation quotes per term for several detachment points such as 0.03, 0.07, 0.15.

9.38 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 79: 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

9.39 Conditional Prepayment Rates

Property	**Allowable values**	**Description**
Instrument Type	*CPR*	
Quote Type	*RATE*	
Name	String	Identifying name of the bond

Table 80: Conditional Prepayment Rates

This quote provides the spread for a specified bond over the benchmark rate.

Examples:

- CPR/RATE/SECURITY_1

10 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 19.

- Index Name: The name of the Index.

  Allowable values are given in Table 81.

- 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 205. Note that alternative index name formats are used (Table 81).

Listing 205: Excerpt of a fixings file

```
20150202 EUR-EONIA -0.00024
20150202 EUR-EURIBOR-1M 0.00003
20150202 EUR-EURIBOR-1W -0.00022
20150202 EUR-EURIBOR-2W -0.00017
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-3M 0.00055
20150202 EUR-EURIBOR-6M 0.00134
20150202 EUR-EURIBOR-6M 0.00271
20150202 GBP-LIBOR-12M 0.009565
20150202 GBP-LIBOR-1M 0.0050381
20150202 GBP-LIBOR-1W 0.0047938
20150202 GBP-LIBOR-3M 0.0056338
20150202 GBP-LIBOR-6M 0.006825
20150202 JPY-LIBOR-12M 0.0026471
20150202 JPY-LIBOR-1M 0.0007143
20150202 JPY-LIBOR-1W 0.0004357
20150202 JPY-LIBOR-3M 0.0010429
20150202 JPY-LIBOR-6M 0.0014357
20150202 USD-LIBOR-12M 0.006194
20150202 USD-LIBOR-1M 0.001695
20150202 USD-LIBOR-1W 0.00136
20150202 USD-CMS-10Y 0.01500
20150202 EUR-CMS-20Y 0.01700
20150202 FX-ECB-EUR-USD 1.0919
20150801 FRHICP 100.36
```

IR Index of form CCY-INDEX-TENOR:	
Index Component	**Allowable Values**
CCY-INDEX	*EUR-EONIA* *EUR-EURIBOR* *EUR-LIBOR* *USD-FedFunds* *USD-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* *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* *RUB-MOSPRIME* *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 81: 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 82: 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 83: 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 84: 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 85: Allowable values for generic indices.

11 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 19.

- 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 206.

Listing 206: 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
```

References

[1] ORE Product Catalogue, available at http://opensourcerisk.org and github.com/OpenSourceRisk/Engine

[2] ORE Methodology Guide, available at http://opensourcerisk.org and github.com/OpenSourceRisk/Engine

[3] http://www.opensourcerisk.org

[4] http://www.quantlib.org

[5] http://www.quaternion.com

[6] http://www.acadia.inc

[7] http://www.lseg.com

[8] http://quantlib.org/install/vc10.shtml

[9] https://git-scm.com/downloads

[10] https://sourceforge.net/projects/boost/files/boost-binaries

[11] http://www.boost.org

[12] http://jupyter.org

[13] https://docs.continuum.io/anaconda

[14] http://www.libreoffice.org

[15] 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

[16] Andersen, L., and Piterbarg, V. (2010): Interest Rate Modeling, Volume I-III

[17] 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.

[18] 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

[19] Leif B. G. Andersen, Michael Pykhtin, Alexander Sokol, *Rethinking Margin Period of Risk*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2719964, 2016

[20] 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

[21] ISDA SIMM Methodology, version 2.5A, (based on v2.5a) https://www.isda.org/a/FBLgE/ISDA-SIMM_v2.5A.pdf